Blockchain Made Wireless

by Minghui Xu

B.S. in Physics, July 2018, Beijing Normal University

A Dissertation submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial satisfaction of the requirements
for the degree of Doctor of Philosophy

May 16, 2021

Dissertation directed by

Xiuzhen (Susan) Cheng
Professor of Computer Science

The School of Engineering and Applied Science of The George Washington University certifies that Minghui Xu has passed the Final Examination for the degree of Doctor of Philosophy as of March 12, 2021. This is the final and approved form of the dissertation.

Blockchain Made Wireless

Minghui Xu

Dissertation Research Committee:

Xiuzhen (Susan) Cheng, Professor of Computer Science, Dissertation Director

Abdou Youssef, Professor of Computer Science, Committee Member

Hyeong-Ah Choi, Professor of Computer Science, Committee Member

Wei Cheng, Assistant Professor of Electrical & Computer Engineering, University of Washington, Committee Member

© Copyright 2021 by Minghui Xu
All rights reserved

Dedication

Handle the difficult while it is still easy. Cultivate the great while it is still small.

– The Tao Te Ching

Acknowledgments

First and foremost, I would like to thank my advisor Xiuzhen (Susan) Cheng for cultivating my research taste and materially affecting my ability to be an independent researcher. It is difficult to imagine a more supportive advisor with as much warmth and patience.

I also want to give special thanks to the many professors who have advised me along the way. Thanks to Dr. Yerukhimovich Arkady, who taught me how to think as a crypto guy. Thanks to Dr. Dongxiao Yu for teaching me a great deal about designing distributed algorithms in wireless networks. Thanks to Dr. Pablo Frank-Bolton, for introducing me to system programming. Thanks to Dr. Robert Pless, who, as a department chair, supported me during the pandemic. Prior to my time in GWU CS, I had the great fortune to work with Shengling Wang, who gave me advice on graduate study and helped me switch to a computer scientist from a physicist.

Thanks to my committee members not already listed: Abdou Youssef (committee chair), Hyeong-Ah Choi, and Wei Cheng. I am grateful for their insightful questions and recommendations on my research [1].

I have also had the great fortune of working with incomparable peers, who have offered their friendship and wisdom: Chunchi Liu, Qin Hu, Yinhao Xiao, Yifei Zou, Yuhan Ma, John Shepherd, Honglu Jiang, Hechuan Guo, and all the others who cheered me on along the way.

Finally, I would never be who I am today if it weren't for my family: Ling Kong, Hongjia Xu, and Nan Zhang.

[1] This study was supported by the US National Science Foundation (IIS-1741279)

Abstract

Blockchain Made Wireless

Known as a distributed ledger technology (DLT), blockchain has attracted much attention due to its properties such as decentralization, immutability, traceability, transparency, and its potential of becoming an infrastructure for various applications. Blockchain can empower wireless networks with identity management, data integrity, access control, and high-level security. However, previous studies on blockchain-enabled wireless networks mostly focus on proposing architectures or building systems with popular blockchain protocols. Nevertheless, existing protocols have obvious shortcomings when adopted in a wireless network where nodes have limited physical resources, favor low communication complexity, and encounter wireless-specific problems.

In this thesis, we first propose a *General Blockchain System Analytic Framework (GBSAF)* by which existing blockchain systems can be conveniently decoupled, broken down, and efficiently analyzed. Then we propose two complementary wireless blockchain protocols, namely BLOWN and *wChain*. BLOWN (BLOckchain protocol for Wireless Networks) contains a novel consensus protocol named Proof-of-Channel (PoC) leveraging the natural properties of wireless networks. BLOWN is specially designed for single-hop wireless networks under an adversarial SINR model. We formalize BLOWN with the universal composition framework and prove its security properties, namely persistence and liveness against adversarial jamming, double-spending, Sybil attacks, which are also demonstrated by extensive simulation studies. *wChain* is a blockchain protocol specifically designed for multihop wireless networks that deeply integrates wireless communication

properties and blockchain technologies under the realistic SINR model. We adopt a hierarchical spanner as the communication backbone to address medium contention and achieve fast data aggregation within $O(\log N \log \Gamma)$ slots where N is the network size and Γ refers to the ratio of the maximum distance to the minimum distance between any two nodes. Besides, *wChain* employs data aggregation and reaggregation, and node recovery mechanisms to ensure efficiency, fault tolerance, persistence, and liveness. The worst-case time complexity of *wChain* is upper bounded by $O(f \log N \log \Gamma)$, where $f = \lfloor \frac{N}{2} \rfloor$ is the upper bound of the number of faulty nodes. To validate our design, we conduct both theoretical analysis and simulation studies, and the results demonstrate the nice properties of *wChain*. To conclude, this thesis points to a vast new space for the exploration of blockchain protocols in wireless networks.

Table of Contents

Dedication iv

Acknowledgments v

Abstract vi

List of Figures x

List of Tables xi

Chapter 1: Introduction 1
- 1.1 Blockchain Basics . . . 1
  - 1.1.1 Basic Structure . . . 1
  - 1.1.2 Types and Properties . . . 4
  - 1.1.3 Philosophy of Blockchain . . . 7
- 1.2 Blockchain in Wired and Wireless Networks . . . 9
  - 1.2.1 Existing Blockchain Protocol . . . 9
  - 1.2.2 Blockchain in Wireless Network . . . 10
- 1.3 Distributed Algorithms in Wireless Networks . . . 12

Chapter 2: GBSAF: General Blockchain System Analytic Framework 15
- 2.1 Motivations . . . 15
- 2.2 GBSAF Overview . . . 16
- 2.3 Structured Blockchain Overlay Network . . . 19
  - 2.3.1 Flat Network . . . 20
  - 2.3.2 Partitioned Network . . . 21
  - 2.3.3 Hierarchical Network . . . 22
- 2.4 Consensus Algorithm . . . 22
  - 2.4.1 Synchrony Abstraction . . . 22
  - 2.4.2 Competitive and Cooperative Consensus Algorithms 24
- 2.5 Distributed Ledger . . . 26
  - 2.5.1 Chain . . . 26
  - 2.5.2 DAG (From One-Dimensional to Two-Dimensional) . 27

Chapter 3: BLOWN: A Blockchain Protocol for Wireless Networks under Adversarial SINR 32
- 3.1 Introduction . . . 32
- 3.2 Models and Assumptions . . . 36
- 3.3 The BLOWN protocol . . . 39
  - 3.3.1 Overview and Utilities of BLOWN . . . 40
  - 3.3.2 The BLOWN Protocol Specifications . . . 44
- 3.4 Protocol Analysis . . . 49

3.4.1 UC Composition of BLOWN 51
3.4.2 Persistence and liveness 54
3.5 Simulation Study 70
3.5.1 Correctness and Efficiency 71
3.5.2 Jamming Attacks and Sybil Attacks 74

Chapter 4: wChain: A Fast Fault-Tolerant Blockchain Protocol for Multihop Wireless Networks **76**
4.1 Introduction 76
4.2 Models and Preliminaries 79
4.3 The Protocol 83
4.3.1 Design Objectives 84
4.3.2 Technical Overview and Utilities 84
4.3.3 Data Aggregation and Reaggregation 87
4.3.4 Fast Fault-Tolerant Blockchain Protocol 93
4.4 Protocol Analysis 94
4.4.1 Efficiency of Data Aggregation and Reaggregation . . 94
4.4.2 Persistence and Liveness 97
4.5 Simulation Results 99
4.5.1 Impacts of Network Size and Γ 101
4.5.2 Impacts of the SINR model parameters 103

Chapter 5: Conclusion and Future Directions **104**

Bibliography **106**

Appendix A: Probabilistic Tools **118**
A.1 Chernoff Bound 118
A.2 Azuma–Hoeffding Inequality 118

List of Figures

1.1 Blockchain prototype 1

2.1 Architecture of Internet, IoT and Blockchain 17
2.2 General Blockchain System Analytic Framework (GBSAF) . . . 18
2.3 Flat, partitioned, hierarchical networks. Nodes in each gray and dashed square frame are fully connected 19
2.4 Two major types of ledger structure 26

3.1 Epoch-based execution. 37
3.2 BLOWN protocol overview. 40
3.3 The protocol (hybrid experiment) $\pi_{\mathrm{B}}[\mathcal{F}_{SIG}, \mathcal{F}_{SORT}]$. 50
3.4 The protocol (hybrid experiment) $\pi_{\mathrm{SORT}}[\mathcal{F}_{\mathrm{VRF}}]$. 54
3.5 The ideal functionality $\mathcal{F}_{\mathrm{SORT}}$ 55
3.6 A simple example to illustrate the concepts of string, Δ-fork, and divergence. Specifically, we have $s_6 = \{0,1,\perp,0,1,1\}$, $l(p_1) = 5$, and $l(p_2) = 4$. The tree presented here is a Δ-fork with $\Delta = 1$, and $|div(p_1,p_2)| = 4$. 65
3.7 Aggregated probability and throughput *vs.* number of rounds, to demonstrate the convergence behavior, where $density = 1, d = 10, N = 100$, and nodes are uniformly distributed. 71
3.8 The performance of BLOWN *vs.* network size (in a uniform or Gauss distribution) and density. 73
3.9 The performance of BLOWN when confronting jamming attacks (random jammers or bursty jammers) and Sybil attacks. 74

4.1 Construction of the first two layers of a spanner followed by an MIS; here hollow and solid circles represent V_0 and V_1, respectively. 81
4.2 Spanner visualization: A $(\log\Gamma+1)$-level full spanner where arrows stand for the MIS relationship: each MIS node at a certain level covers a set of nodes in the next lower level. 82
4.3 The performance of *wChain* *vs.* the network size N and Γ (under uniform, normal, and exponential distributions). 100
4.4 The performance of *wChain* with various α and β (under a uniform distribution). 101

List of Tables

2.1 Comparison of Recent Blockchain System Surveys 15

3.1 Summary of Notations (BLOWN) 39

4.1 Summary of Notations (*wChain*) 83

Chapter 1: Introduction

1.1 Blockchain Basics

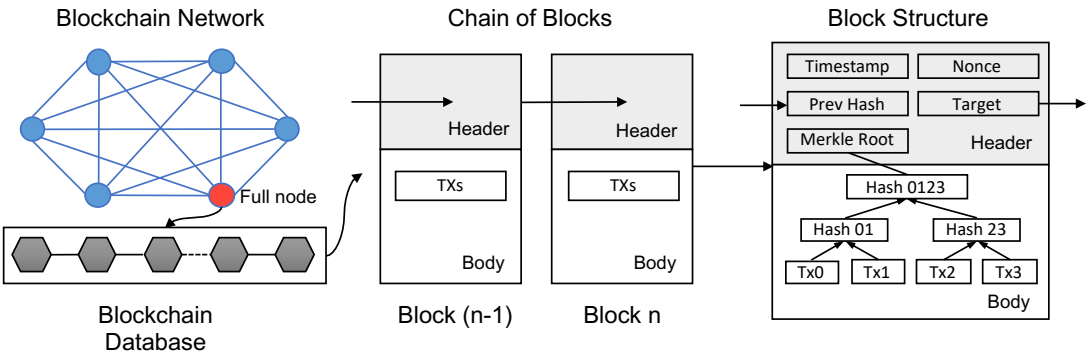


Figure 1.1: Blockchain prototype

1.1.1 Basic Structure

Figure 1.1 shows a blockchain prototype which is an abstraction of Bitcoin. A blockchain is a distributed ledger organized as a chain of blocks linked by hash and spread over all the blockchain nodes in a peer-to-peer network. A complete blockchain starts from a "genesis block", also known as block zero. A full node is a blockchain node that stores a complete blockchain in a local blockchain database. In contrast, a light node only stores a sequence of block headers and can validate the authenticity of blocks through Simplified Payment Verification (SPV). However, the light node is vulnerable because maintaining an incomplete blockchain leads to security concerns, e.g., double-spending attacks. If we look into a blockchain, a block is comprised of a header and a body. Each block, except for the genesis block, is chained to the previous block by referencing its hash. Generally,

a header consists of timestamp, nonce, previous hash, target, and Merkle root, while a body includes serialized transactions.

Additionally, three techniques involved in Figure 1.1 need further explanations: hash function, Merkle tree, and nonce. In analogy with fingerprints, a hash function is designed to map data of arbitrary length to a fixed value, denoted as $h := H(x) : \{1,0\}^* \rightarrow \{1,0\}^l$. The Secure Hash Algorithms (SHA) [1] is a family of hash functions standardized by National Institute of Standards and Technology (NIST) and considered collision-resistant so that it is hard for any probabilistic polynomial-time (PPT) algorithm to find a pair of x_1 and x_2, satisfying $H(x_1) = H(x_2)$ [49]. h as a fingerprint can be conveniently transferred (fixed and limited length) and securely verified ($H(x)$ is a one-way function). The Merkle tree, introduced by Ralph Merkle [71], is used to store multiple fingerprints using a binary tree. As the block body shown in Figure 1.1, the leaves of a Merkle tree are inputs, and each parent node is the hash of the values of its two children. With a Merkle tree, a block only needs to include a single Merkle root in its header, and the cost of one-time transaction verification is $O(\log N)$, where N is the number of inputs [16]. The cryptographic nonce is a one-time and uniform random number. In the context of Bitcoin, a miner can generate a proof-of-work when it successfully finds a nonce so that the double hash of the block header is smaller than the target. In other contexts, a nonce might not refer to a cryptographic nonce. For example, the Ethereum account uses a nonce as a transaction counter to prevent double-spending attacks.

For cryptocurrency, a transaction is a transfer of money and should be validated before organized into a block. A transaction model affects

[1] The most popular hash function used in the blockchain is SHA-256, which is considered secure, for now at least even though polynomial-time quantum algorithms for factoring and discrete logarithms have theoretically shown that it can menace the security of existing hash functions.

the transaction structure, transaction validation methods, and balance calculation approach. Currently, there are two popular transaction models: Unspent Transaction Output (UTXO) model and the Account/Balance model. In the UTXO model, each transaction spends money in UTXOs obtained from previous transactions and creates new UTXOs that can be spent in the future. Therefore, a legitimate UTXO should specify its owner and amount to be verifiable. The balance on a node's account is the total amount in all of its UTXOs. In contrast, the Account/Balance model is similar to a traditional bank. Each account is associated with a balance. A transaction can spend some coins in one account by crediting the amount to another account. Each model has its advantages and disadvantages in terms of security, scalability, simplicity, efficiency, and storage cost.

- *Security:* In the UTXO model, a valid transaction should contain valid UTXOs, whose total value should exceed the total amount of outputs. Each UTXO can only be spent once to avoid double-spending issues. In the Account/Balance model, a trivial method of avoiding double-spending attacks is to embody a counter in each transaction. Confronting two conflicted transactions, the validator would choose the one with a lower counter value. Both transaction models can add chain identity and nonce to mitigate replay attacks.

- *Scalability:* UTXOs support parallel processing of transactions since each UTXO is atomic, and transactions can spend different UTXOs simultaneously. In contrast, transactions should be processed serially in the Account/Balance model.

- *Efficiency:* The UTXO model is logically complicated, thus may leave the blockchain systems inefficient if not properly implemented. There

are some methods of improving the efficiency of the UTXO model. For example, each full node can maintain a reduced UTXO set, a subset of all UTXOs in the blockchain network. The UTXO set can be constantly updated according to newly proposed transactions. The transaction validation is reduced to check that UTXOs in a transaction are legal. Besides, we can store metadata to improve performance at the cost of storage [2].

- *Storage Cost:* In the UTXO model, an account needs more space to store its UTXOs even though UTXOs can be organized carefully. Also, a full node has to store the UTXO set containing all the UTXOs for validating transactions. For example, Bitcoin has roughly 65 million UTXOs at the time of writing, so that each full node stores a UTXO set of 3.65 GiB. More elaborate design can reduce storage cost – Bulletproofs can potentially improve range proof data with factor 10 reduction in size. [19].

1.1.2 Types and Properties

1.1.2.1 Blockchain Types

Typically, there are three types of blockchains based on scale and openness: public blockchain, private blockchain, and consortium blockchain. Public blockchain is also termed as permissionless blockchain. Private blockchain and consortium blockchain are permissioned.

- *Public Blockchain:* Public blockchains are fully decentralized, permissionless, and self-governed blockchains, e.g., Bitcoin, Ethereum, Litecoin. Anyone can feel free to join or leave the public blockchain network,

[2] Bitcoin core version 0.15.0 improved the performance of validating blocks at the cost of 10% increasing UTXO database by 15%.

and there is no access restriction to read or write, which brings about openness and liveness. Each node can issue transactions or contribute to the consensus process as a miner, validator, or voter. Incentive mechanism is always applied in the form of transaction fees, mining rewards, etc., to encourage nodes to maintain the self-governed nature. However, public blockchains confront various challenges from malicious nodes, such as token stealing [9], selfish mining [37, 97], Sybil attack [34], eclipse attack [101]. Thus the openness leads to more tight security requirements. Besides, distributed ledgers can be publicly accessed, arising the privacy and anonymity concerns [61]. Power consumption is also a problem to public blockchains, which introduce large-scale or even global mining.

- *Private Blockchain:* Private blockchain, as the name suggests, is the blockchain used by a single party. Instead of joining or leaving at will, blockchain nodes have to access a private blockchain through a verified invitation on an intranet. Thus, private blockchains are not permissionless and open but permissioned and closed. Private blockchains are efficient, safe, privacy-preserving, and easy-to-manage but sacrifice decentralization and openness. The owner or operator of a private blockchain can override, rollback, delete and edit ledgers, undermining blockchains' trustlessness. The private blockchain is a top choice for a single entity that intends to benefit from blockchain technology (e.g., inner data sharing, identity management).

- *Consortium Blockchain:* Consortium blockchains as permissioned and partially decentralized lie between public and private blockchains. Typical consortium blockchains include Hyperledger Fabric [7], Ripple

[98], Corda [18]. Unlike private blockchains, consortium ones are not granted to a single entity but manipulated by more than one organization. Blockchain nodes from different organizations are selected to participate in the consensus process. Consortium blockchains are perfect for building trustless applications among different organizations. For example, blockchain-based energy trading commonly involves energy sellers, buyers, banks, etc., among whom active cooperation and secure trading can be achieved [62].

1.1.2.2 Blockchain Properties

Blockchain technology exhibits superior properties that put itself to the front.

- *Immutability:* Briefly, immutability means that blockchains are permanent and unalterable over time. In other words, no one can modify a blockchain from the onset of its acceptance by the blockchain network. To achieve immutability, blockchain technology adopts a chain structure to realize a immutable ledger (space), a consensus algorithm to decide the next agreed and authenticated state (time). Immutability prevents disputes, preserves data integrity, and contributes to the efficiency of auditing. However, the other side of the coin is that private and sensitive information can not be hidden once being publicly recorded since history can never be altered.

- *Decentralization:* Based on the previous study by Paul Baran in 1962 [14], networks can factor into two components: centralized (star) and distributed (grid or mesh). Decentralized topology refers to a set of stars being connected, which precisely describes our internet. Compared to the centralized network, decentralization avoids the reliance

on a single point of control, and no centralized authority can manipulate the network. Technically, public blockchains are designed to be as "distributed" as possible so that a ledger can be maintained by massively distributed nodes. Consortium and private blockchains are decentralized with fewer stars than public blockchains, or even centralized.

- *Openness:* Openness is made available to the public (or permissionless) blockchains, which are fully open to the public. Each node can freely join or leave, issue transactions, or participate in consensus. Different from a traditional bank or service provider, openness grants blockchain users the right to hold a copy of historical operations on the blockchain. Besides, some blockchain implementations are open-source to acquire public trust and rapid development. A blockchain without a clear threat model, security guarantee, and proper implementation would be suspected as insecure and inefficient. Thus, the openness makes blockchain transparent and auditable and largely contributes to the liveness and prosperity of the blockchain community.

1.1.3 Philosophy of Blockchain

In this part, we mathematically articulate the philosophy underlying blockchain technology. The salient properties of blockchain essentially come from two procedures: generation of honesty and preservation of honesty.

1.1.3.1 Generation of Honesty

Assume a set of blockchain nodes d, we design a Honesty Generation Function (HGF) to generate a honesty distribution from d, which is denoted

as $D = HGF(d)$. Finally, sampling $S(D)$ gives the result that D satisfies the major honesty with a possibility P, that is $P = S(D)$.

There are two factors that affect P: d and S. d depends on the registration policy of blockchains, and S depends on the blockchain protocol. Simple designs are to set S as a random sampling so that P is entirely based on d, which requires d should fulfill the major honesty requirement. However, sophisticated designs can relax d, by designing sophisticated S.

1.1.3.2 Preservation of Honesty

After generating honesty, it is necessary to preserve honesty, both temporally and spatially. Temporal security emphasizes the temporal order of behaviors. It correctly records the events according to their real-time or occurrence. The ground-truth record after the i-th event e_i is $R_{i+1} = g(e_1, e_2, \ldots, e_i)$, where $g(\cdot)$ denotes a temporal-secured event logger. Or we can rewrite R in a recurrence form: $R_{i+1} = g(R_i, e_i)$. The latter-happened event cannot timely surpass the former-happened event, which solves the atomicity of events. For verification procedure, one should efficiently verify the sequential correctness of R.

However, one blockchain copy alone, cannot guarantee the security of itself (R). It can be modified into wrongness without distributed preserved by multiple nodes. So it requires spacial security to protect R. Distributed ledger is borrowed from the distributed database, to protect temporal security. $h(\cdot)$ is the storage distribution function, where $(R^{(1)}, R^{(2)}, \ldots, R^{(n)}) = h(R)$. $R^{(1)}, R^{(2)}, \ldots, R^{(n)}$ are data stored by blockchain nodes. Besides, we denote $h^{-1}(\cdot)$ as the truth collection function which can recover $R = h^{-1}(R^{(1)}, R^{(2)}, \ldots, R^{(n)})$. Spreading the exact duplication of R over all nodes is a straightforward design, namely $R^{(1)} = R^{(2)} = \cdots = R^{(n)} = h(R) = R$, and thus causes low

efficiency and high storage overhead. Sophisticated designs can make $h(\cdot)$ more complex, but $R^{(1)}, R^{(2)}, \ldots, R^{(n)}$ more storage-space-friendly.

1.2 Blockchain in Wired and Wireless Networks

1.2.1 Existing Blockchain Protocol

We survey blockchain consensus protocols of three categories: proof of physical resources, proof of virtual resources, or message passing. Proof of physical resources means that users compete for proposing blocks by demonstrating their utilization of physical resources. Proof-of-Work (PoW) is of the most use in blockchain protocols. The most popular example of PoW is the Bitcoin proposed in 2008, which selects leaders by computational powers (Proof-of-Work) [78]. Ethereum provides the Turning-complete Ethereum Virtual Machine (EVM) and adopts a modified PoW (with Ethash) [114].

Alternatives to PoW include Proof-of-Space [36], Proof-of-Burn (PoB) [48], Proof-of-Elapsed Time (PoET) [83], and Proof-of-Space Time (PoST) [76]. The Proof-of-Space, also known as Proof-of-Capacity or Proof-of-Storage, refers to competing by occupied memories or disk spaces, PoB means that a node can destroy coins to virtually earn mining rights, and PoET, proposed by Intel, leverages trusted hardware (e.g., SGX) to determine how long a node has to wait before it is allowed to generate a block.

In contrast to the proof of physical resources, the proof of virtual resources aims to show the utilization of virtual resources such as reputation, stake, or elaborately defined weight. For example, Proof of Stake (PoS) was developed to address the power consumption issue of PoW and it resorts to stakes as voting rights rather than computational powers. Algorand uses a cryptographic sortition algorithm to randomly select a verifiable committee

according to stakes [42]. IOHK created the Ouroboros family in recent years, which adopts PoS and G.O.D coin tossing to randomly choose a leader according to the stake distribution [51]. Snow White utilizes an epoch-based committee which embodies successful block miners in a specific time period so that all nodes have an identical view of the committee [17]. In Proof-of-Reputation (PoR) [72], each node is assigned a reputation, and a node can write blocks only when its reputation meets certain requirements. Thus PoR always comes with incentive mechanisms or economic penalty strategies.

In message-passing blockchain protocols, nodes can perform local computations and send messages to each other to reach consensus. This method provides blockchain the robustness to Byzantine failures while ensuring liveness and safety. In Ripple, a transaction that receives more than 80% votes from UNL-recorded servers can step into the next round, and transactions surviving through the whole RPCA process can be added to the blockchain [98]. ELASTICO partitions nodes by their unique identities and a consensus is reached in each shard based on byzantine agreement protocols [69]. Stellar creates overlapped shards, also known as quorum slices, leveraging Federated Byzantine Agreement (FBA) to reach consensus [67]. Omniledger uses lottery-like RandHound and VRF-based leader election algorithms to assign validators to each shard [55]. Other message-passing based protocols utilized in blockchain include PBFT [23], HoneyBadgerBFT [73], Tendermint [59], and Hotstuff [117].

1.2.2 Blockchain in Wireless Network

In this section, we summarize the state-of-the-art of blockchains that have been used in IoT (closely related to but not a pure wireless network)

and wireless networks.

1.2.2.1 Blockchain for Internet of Things

IoT always encompasses wireless devices. Blockchain technology has also been studied for its applications in IoT, including access management, security enhancement, privacy protection, etc. Oscar Novo proposed a new architecture for access management in IoT based on blockchain technology [82]. The architecture consists of six components, which include simplified wireless sensor networks. Dorri *et al.* optimized blockchain for IoT by introducing the distributed trust model. New blocks proposed by the user with high trust can be free from complete transaction validation, so this scheme decreases transaction processing overhead.

1.2.2.2 Blockchain for Wireless Networks

Blockchain technology has been studied for wireless applications such as mobile edge computing (MEC) [38], intelligent 5G [28], vehicular networking [70], secure localization [54], and Wireless D2D Transcoding [63]. Feng *et al.* [38] considered the joint optimization of blockchain and MEC through a radio and computational resource allocation framework. Dai *et al.* [28] proposed a secure and intelligent architecture for next-generation wireless networks by integrating blockchain and AI technologies. In vehicular ad hoc networks, Malik *et al.* [70] utilized blockchain for secure key management. A blockchain-based trust management model was proposed to ensure secure localization in wireless sensor networks [54]. A blockchain-enabled Device-to-Device (D2D) transcoding system was developed to provide trustworthy wireless transcoding services in [63]. Onireti *et al.* [85] provided an analytic modeling framework to obtain a viable area for wireless PBFT-

based blockchain networks. In [91], a trustless mechanism with PoW-based blockchain was established to incentivize nodes to store data. Liu *et al.* [64] realized computation offloading and content caching with mobile edge nodes in wireless blockchain networks. Sun *et al.* Sun *et al.* [107] proposed an analytic framework to explore how the performance and security of wireless blockchain systems are affected by wireless communication factors such as communication range and SINR. This research group analyzed how the distribution of the SINR ratio affects the performance of the blockchain system in another work [106].

Notwithstanding these studies on applying blockchain to wireless networks, we are still in dire need of blockchain protocols that are specifically designed for wireless networks, which drives us to investigate and summarize existing distributed leader election and consensus algorithms for wireless networks in the sequel.

1.3 Distributed Algorithms in Wireless Networks

Consensus and leader election have been extensively explored in various wireless contexts, and the corresponding solutions can guide us to design wireless blockchains in a proper way. Most existing studies on consensus and leader election for wireless networks assume a high-level wireless network abstraction [75] [79] [80] [33] [111] [90] or a realistic model grappling with issues in physical and link layers [26] [99] [11] [93] [43].

Moniz *et al.* [75] proposed a BFT consensus protocol with runtime bounded by $O(N^2)$ among $k > \lfloor \frac{N}{2} \rfloor$ nodes in wireless ad hoc networks. They hid physical layer information but let nodes directly use high-level communication primitives. Leveraging the elegance of absMAC, Newport provided upper and lower bounds for distributed consensus in wireless networks

[79]. Subsequently, Newport and Robinson proposed a fault-tolerant consensus algorithm that terminates within $O(N^3 \log N)$ with unknown network size [80]. A fully distributed leader election scheme was proposed to make election value unforgeable and be resistant to jamming attacks in wireless sensor networks. A secure extrema finding algorithm was proposed as a lock-step leader election algorithm that can be transformed to a secure preference-based leader election algorithm with a utility function depicting nodes' preference in wireless ad hoc networks [111]. A top k-leader election algorithm with faulty nodes was presented in [90].

Chockler *et al.* [26] investigated the relationship between collision detection and fault-tolerant consensus under a graph-based model. Assuming a graph model with message delays, Scutari and Sergio [99] proposed a consensus algorithm in wireless sensor networks with multi-path fading. Aysal *et al.* [11] studied the average consensus problem with probabilistic broadcasts under a graph-based model. Richa *et al.* [93] focused on self-stabilization of leader election for single-hop wireless networks to mitigate jamming attacks by adaptively adjusting the transmission probability at the MAC layer. Gołębiewski and Klonowski [43] proposed a fair leader election scheme in ad-hoc single-hop radio sensor networks, enabling resistance to adversaries who can transmit continuously to block channels or try to forge identities.

In contrast, we observe that designing a blockchain protocol is more challenging than consensus and leader election in wireless networks. On the one hand, blockchain introduces new data structures such as a chain of blocks, and thus the corresponding computation, communication, and storage overheads should be carefully addressed. On the other hand, it is harder to balance performance and security in blockchain since tradi-

tional consensus and leader election offer more straightforward services without requiring a strong security guarantee. They can not be migrated to blockchain system with only minor changes since blockchain requires extra design elements (e.g., randomness, transaction and block verification, blockchain update) to guarantee strict persistence and liveness properties.

Chapter 2: GBSAF: General Blockchain System Analytic Framework

2.1 Motivations

There is always a loud voice that blockchain is hard for beginners who need to comprehend computer networking, cryptography, distributed systems, economics, etc. We found that the hardship of understanding blockchain systems may result from the following reasons: 1) A blockchain system is elaborately designed with various techniques in separate fields and lots of components interacting with each other. However, the existing analytic framework or architecture is unable to clearly and precisely articulate blockchain systems. In other words, blockchain systems are rarely illustrated in a well-understood manner; 2) Terminologies and concepts are vague and have ambiguity in technical reports and even peer-reviewed papers. For example, what is the difference between partitioning and sharding? Does "nonce" have the same meaning in Bitcoin core and Ethereum? What is the difference of persistence, consistency, safety, and correctness properties of a blockchain?

Table 2.1: Comparison of Recent Blockchain System Surveys

Perspective	**Recent papers**	**Covered by GBSAF**
Analytic framework or architecture	[45][109][5][15][40] [113][126][94][68][32]	✓
Blockchain network	-	✓
Consensus algorithm	[77][74][122][115][113] [110][126][125][24][96] [32][123]	✓
Incentive mechanism	[122][66]	✓
Distributed ledger	-	✓

Prior studies that have noted the importance of understanding blockchain appropriately are listed in Table 2.1. We compare these works and find that existing works neither cover sufficient components of blockchains nor generally and wholly modeling blockchain systems. Therefore, we need a new framework enabling component-wise analysis of increasingly complex blockchain systems. This framework should achieve the following design goals:

1. Precisely define the core concepts used in blockchain such as blockchain node, blockchain network, consensus node, consensus algorithm, leader node, committee, etc.

2. Generalize blockchain systems and completely capture key building blocks and workflow.

3. Precisely and clearly define major components and show the interaction of them, supporting component-wise analysis of complex blockchain systems.

2.2 GBSAF Overview

To show the relationship between blockchain systems and the architectures of Internet and IoT, we provide Figure 2.1. The Internet architecture (or Internet protocol stack) consists of five layers: physical layer, link layer, network layer, transport layer, and application layer [58]. For IoT, there is no consensus on its architecture, so we adopt the most basic one, the three-layer architecture [100]. The blockchain only operates at the application layer while the underlying layers always remain unchanged. Remarkably, blockchain technology brings about a new infrastructure at the application level. It can provide excellent features such as traceability, decentralization,

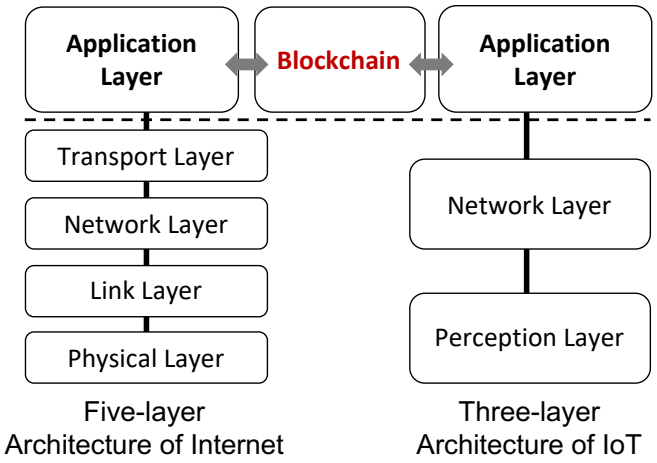


Figure 2.1: Architecture of Internet, IoT and Blockchain

immutability, which we can not simultaneously realize by traditional applications and application-layer protocols (HTTP, SMTP, FTP, etc.). In the past few years, numerous blockchain systems have emerged in academia and industry, each claiming its advantages over others. However, it is still a challenging task to conduct a comparable analysis on those systems. This section provides the General Blockchain System Analytic Framework (GBSAF), by which existing blockchain systems can be conveniently decoupled, broken down, and analyzed. This framework is also capable of providing insightful ideas about designing new blockchain systems. In the following, we first provide the definitions of four elementary concepts widely used by blockchain systems.

Definition 1. ***(Blockchain Node).*** *A blockchain node is a node which runs a blockchain system and connects to a blockchain network. We denote* (N_0) *as the set of all blockchain nodes.*

Definition 2. ***(Structured Blockchain Overlay Network).*** *A structured blockchain overlay network is a structured network which consists of blockchain nodes.*

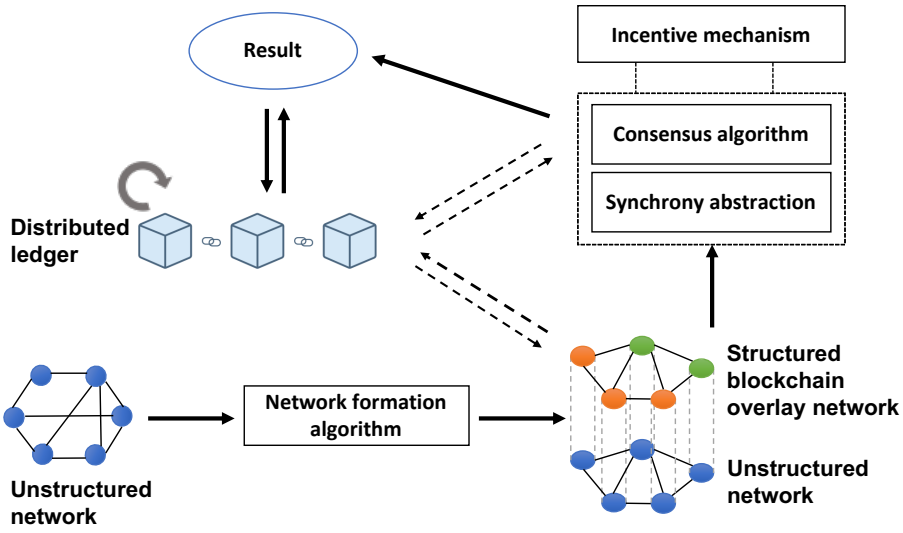


Figure 2.2: General Blockchain System Analytic Framework (GBSAF)

Definition 3. ***(Consensus Algorithm).*** *A consensus algorithm of a blockchain refers to an algorithm of achieving agreement on the next state of a blockchain system.*

Definition 4. ***(Consensus Node).*** *A consensus node is a blockchain node which executes a consensus algorithm. We denote N_1 as the set of consensus nodes, and $N_1 \subseteq N_0$.*

Based on the above definitions, we describe the workflow of GBSAF as shown in Figure 2.2 [1]. The network formation algorithm first takes as input an unstructured network and outputs a structured blockchain overlay network, depending on functionalities of blockchain nodes. Then a blockchain system should execute the synchrony abstraction and consensus algorithm. The synchrony abstraction helps model and verify the time synchronization of a blockchain system, which could be synchronous, partially synchronous, or asynchronous. A consensus algorithm in blockchain is

[1]The old version of this figure has previously published in the doctoral thesis of Chunchi Liu, who is the collaborator of mine. It is to claim that we have equal contributions to this framework.

a algorithm used to achieve agreement on data (e.g., transaction, block) among consensus nodes. The outcome of executing a consensus algorithm is stored in the distributed ledger. In addition, the incentive mechanism aims to reward or penalize blockchain nodes to maintain a stable blockchain system. In the following sections, we deficit major components of GBSAF, including the structured blockchain overlay network, consensus algorithm, and distributed ledger.

2.3 Structured Blockchain Overlay Network

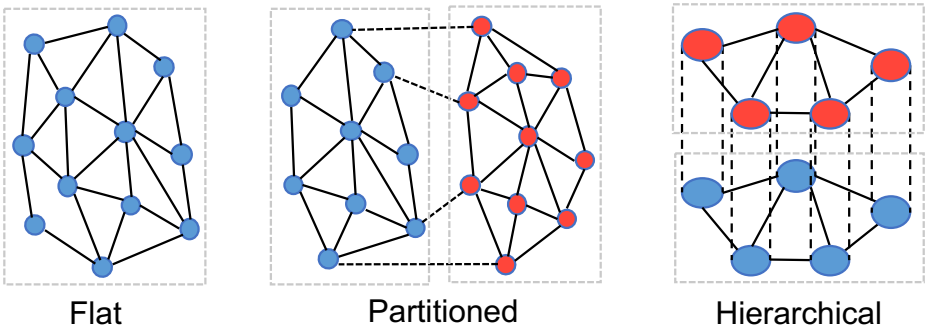


Figure 2.3: Flat, partitioned, hierarchical networks. Nodes in each gray and dashed square frame are fully connected

The structured blockchain overlay network should be first established ahead of the synchrony abstraction and consensus process. According to Definition 2 and 4, blockchain nodes form a structured blockchain overlay network. Using a topology-based taxonomy, the structured blockchain overlay network can be categorized into ***Flat***, ***Partitioned*** or ***Hierarchical*** as shown in Figure 2.3. Before elaborate on these concepts, we introduce the following two concepts: leader node and committee.

Definition 5. ***(Leader Node).*** *A leader node is an N_1 node which has been selected to finalize the next result (e.g., a miner who is selected to add the*

next new block to the blockchain)

Definition 6. ***(Committee).*** *A committee is a subset of N_1 nodes, each having a chance of being selected as the next bookkeeper during a specific time slot.*

Then we deficit the three categories of structured blockchain overlay networks. In the reminder of this chapter, we adopt the notion of "networ" for simplicity, representing the "structured blockchain overlay network".

2.3.1 Flat Network

A flat network means that all consensus nodes with identical functionality are (virtually) fully connected, constituting a complete graph. Based on Definition 5, a flat network only consists of N_1 nodes, among which a leader node is directly selected for block finalization. The flat network design appeared in some early blockchain systems, including Bitcoin, Ethereum, Namecoin, Litecoin, IOTA, etc.

In particular, a flat network can be organized through a registration process. Concerning Bitcoin, miners are N_1 nodes, each competing for being a leader node to win an opportunity of finalizing the next block. Even though the fork may happen when more than one block proposals appear simultaneously, Bitcoin sets the longest chain rule to avoid this from happening and guarantee that only one leader can be selected for each round. It takes four steps to become an eligible miner: prepare Bitcoin mining hardware, set up a Bitcoin wallet, download mining software, and choose a mining pool. The leader election in Bitcoin is exactly executing the PoW consensus algorithm, which is inefficient but largely prevents Sybil attacks. Ethereum and other altcoins that leverage flat networks have similar registration processes but vary in consensus algorithm. The

flat network is decentralized, easy-to-implement, and has a simple design rationale.

2.3.2 Partitioned Network

A partitioned network indicates that the graph of N_1 nodes is not fully connected. The partition in blockchain refers to partitioning blockchain networks spatially, indicating that nodes are divided into shards, each producing intra-shard consensus. In contrast, partitioning blockchain network temporarily means that some nodes can disconnect from the blockchain and maintain consistency within a limited time. Therefore, temporal partitioning is usually recognized as partially synchronous or asynchronous problems.

Partitioned networks of blockchains can be further categorized into *horizontally partitioned networks* and *vertically partitioned networks*. The horizontal partition means dividing a blockchain network into multiple, separate, asynchronous subnetworks with identical functionality and still support a global consensus. Some existing systems have adopted horizontally partitioned networks, e.g., Ripple, ELASTICO [69], Chainspace [4], Omniledger [55], Monoxide [112]. On the other hand, the vertical partition involves multiple different blockchain systems that can interoperate, such as Atomic Cross-chain Swap [44], Pegged Sidechain [12], Proof-of-Stake Sidechains [53], and Proof-of-Work Sidechains [52]. In general, blockchain utilizes sharding techniques and sidechain techniques to realize horizontally partitioned networks and vertically partitioned networks, respectively. Partitioned networks can be used to develop scalable and efficient blockchain systems. Still, it is criticized for introducing more cross-chain communication complexity and being hard to achieve consensus among subnetworks.

2.3.3 Hierarchical Network

Lastly, a hierarchical network consists of two types: *committee-based network* and *hierarchical multi-blockchain network*. The committee-based network contains at least one committee (as Definition 6) that is a group of nodes selected randomly according to certain features such as identity, stake, or pure randomness. The committee-based network can be organized by a committee election process, which is used to randomly select a subset of consensus nodes to form a committee. Blockchain systems have widely adopted the committee-based network. E.g., Algorand, ELASTICO, ByzCoin, and Snow White.

On the other hand, a hierarchical multi-blockchain network, in contrast, contains at least one beacon blockchain and multiple different blockchain systems. They work in a hierarchical manner. For example, the beacon blockchain coordinates communication and consensus processes among low-level heterogeneous blockchains. Different from partitioned networks that can interoperate fairly, a hierarchical multi-blockchain network establish a hierarchy. A notable blockchain that has adopted a hierarchical multi-blockchain network is the Hyperservice [65]. In a nutshell, hierarchical networks can alleviate the competition and reduce the communication overhead. However, it is criticized for sacrificing decentralization.

2.4 Consensus Algorithm

2.4.1 Synchrony Abstraction

Before analyzing blockchain consensus algorithms, we should figure out the synchrony abstraction, which can model and verify the time synchronization of blockchain systems and specify whether a blockchain network is

synchronous, partially synchronous, or asynchronous. Formally speaking,

- *Synchronous (fully synchronous and semi-synchronous):* In synchronous settings, blockchain systems execute under a synchronized clock, and message transmission delay is Δ-bounded, where $\Delta \in [0, 1]$ is a known fixed value. Some blockchain systems even make a strong assumption that network channels are *fully synchronous* without a time delay, indicating $\Delta = 0$. In a *semi-synchronous* model, $\Delta \in (0, 1]$. A synchronous model in blockchain can be achieved using a universal clock or by relying on a good network connection. Moreover, a synchronous model is convenient for theoretical analysis in that it leads to deterministic time synchronization, but real-world cases are at best partially synchronous and generally asynchronous.

- *Partially synchronous:* The partially synchronous model lies between synchronous and asynchronous models [35]. There are two situations we can say a blockchain is partially synchronous: a) an upper bound of message transmission delay exists as Δ, but unknown, which is also known as *a-priori* bound Δ; b) The Δ exists and known to us, but the blockchain system is unreliable, and the Δ partially takes effect.

- *Asynchronous:* In this circumstance, message transmission delays are finite and unbounded. Consensus can not be reached in the fail-stop case that terminates within a bounded time, even with only one crash failure according to the FLP impossibility result (named after Fischer, Lynch and Patterson) [39]. However, we can circumvent this problem by introducing non-determinism into blockchains.

Design a blockchain system in asynchronous networks is a tricky problem. The consistency of Bitcoin consensus (a.k.a. Nakamoto consensus)

algorithm has been analyzed in both synchronous [41] and partially synchronous [78, 31, 104, 87] settings. Garay *et al.* [41] formalized the Bitcoin consensus within a synchronous network in which messages sent by honest nodes arrive in the next round without any delay. Decker *et al.* [31] demonstrated that the primary cause for folks is the message delay, which is common in the partially synchronous model.

Synchrony abstraction has been underestimated but is of great importance since it serves as an invisible bridge between the structured blockchain overlay network and the consensus algorithm. A consensus algorithm that can be run in synchronous networks may fail to function when the system is directly transplanted to an asynchronous network. This case is not rare when we intend to implement blockchain in a more heterogeneous or dynamic network, e.g., IoT networks comprised of assorted devices, an unmanned aerial vehicle network, wireless networks considered in the following two chapters. Figuring out the function of synchrony abstraction helps extend blockchain to a vaster area of applications.

2.4.2 Competitive and Cooperative Consensus Algorithms

A consensus algorithm terminates when the majority of honest users reach an agreement on the next block. For example, in the context of Bitcoin, the consensus algorithm is PoW, and in Algorand, the consensus algorithm is BA*. Leveraging the GBSAF, we are the first to divide existing consensus algorithms into two categories: *competitive consensus algorithms* and *cooperative consensus algorithms*.

- *Competitive Consensus:* A process in which each consensus node competes with the others to unilaterally maximize its probability of winning the right to propose the next block. E.g., PoW, PoS.

- *Cooperative Consensus:* A process in which consensus nodes collaboratively maximize the probability of the next best bookkeeper to win the right to propose the next block. E.g., PBFT, Tendermint.

The major difference is that the competitive consensus algorithm requires each consensus node to compete for proposing the next block based on their resources. In contrast, the cooperative consensus algorithm organizes the blockchain nodes to minimize fault and produce an honest result collaboratively.

Concretely, competitive consensus algorithms, to be straightforward, are the ones that need competition. They ensure correctness based on three-step reasoning: 1) Assume the honest party owns the majority of the resources; 2) Randomly choose the creator of the next block based on the distribution of resources; 3) The random leader node being selected should be honest. We can see that this is a way of generating honesty according to our blockchain philosophy. We can then argue that there are two types of competitive consensus based on proof of physical resources or proof of virtual resources. Proof of physical resources means that users compete for proposing blocks by demonstrating their utilization of physical resources such as mining hardware, random-access memory, trusted hardware, hard drive. Proof of virtual resources aims to show the utilization of virtual resources such as reputation, stake, random number, or elaborately defined weight.

On the other hand, the cooperative consensus algorithms resort to delicate designs that manage the nodes to eliminate fault and create correctness collaboratively. Any consensus algorithm that follows this principle falls into this category. The cooperative consensus algorithms are mostly based on two models: message-passing and shared-memory model. These two

models differ based on how consensus nodes communicate. The message passing indicates that nodes reach consensus by sending and receiving messages, while shared memory means that nodes reach consensus by reading and writing a shared memory. A recent work by Aguilera *et al.* shows that these models can be integrated, namely, message passing while sharing memory model (M&M) [2].

2.5 Distributed Ledger

How a distributed ledger can operate depends on what types of ledger structure a blockchain leverages. The ledger structure has alternatives as chain and Directed Acyclic Graph (DAG) from a topological perspective.

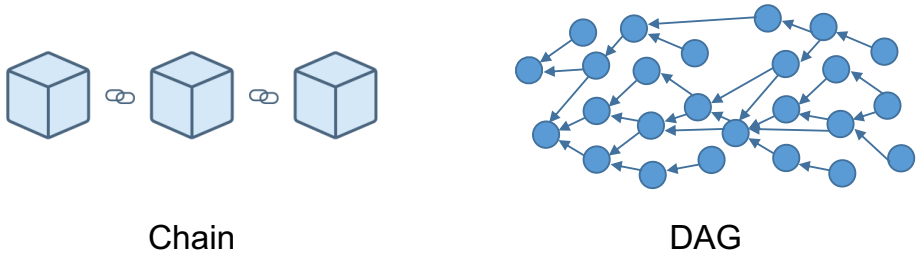


Figure 2.4: Two major types of ledger structure

2.5.1 Chain

The chain structure is of most use for blockchain and has been widely studied and implemented. A chain contains a sequence of blocks, each embodying organized data and points to a previous block using a hash. It is especially welcomed due to its linear ledger structure, which contributes to simple design and qualified tamper-proofing. It is convenient to verify transactions by checking the latest blockchain. An adversary can only tamper with the blockchain by cutting off some hash pointers since such

malicious behavior is easily detected and avoided.

However, it does not remain easy to maintain a chain structure rigorously in real-world systems. Bitcoin first adopts the longest-chain rule to let honest nodes choose the longest branch to avoid forks. GHOST [104] is an alternative to the longest-chain rule and designed to improve chain structure. A variant of GHOST has been implemented in Ethereum and has shown its correctness and efficiency. The GHOST protocol allows collecting all mined blocks in a tree structure. When confronting fork problems, GHOST requires nodes to select the Greedy Heaviest-Observed Sub-Tree instead of choosing the longest chain. Therefore blocks not in the main chain still matter in the sub-tree computation. Compared to the longest chain rule, GHOST increases computational power utilization by counting into more miners' efforts. GHOST also disrupts the relation between the longest-chain rule and the 51% attack so that attackers with competitive computational power cannot launch double spending through creating a fork.

A fundamental question about blockchain is whether we can use other ledger data structures to improve the performance while preserving immutability and robustness. This question drives researchers to explore a more complex ledger structure.

2.5.2 DAG (From One-Dimensional to Two-Dimensional)

Directed Acyclic Graph (DAG) is an intuitive attempt to increase ledger capacity and efficiency by extending the one-dimensional chain structure to a two-dimensional graph. The difference between a DAG and a tree is that a node in DAG can have multiple predecessors but only one predecessor in tree. Studies on DAG can be categorized into two classes. The first one is what we commonly recognize as a pure DAG-based ledger, which is

flexible, free of transaction fees, and has weaker requirements on global consensus and deterministic result. The second one is to partly adopt the idea of DAG, improve the blockchain by allowing blocks to have multiple parents, and managed to preserve good qualities like global consensus and high-confidence transaction finality.

Typical examples of the pure DAG-based class include the IOTA Tangle [89], Byteball [27], and Swirlds Hashgraph [13]. The most popular project that develops DAG-based blockchain is the IOTA Tangle. In Tangle, when a transaction wishes to be processed, its owner must validate two previous transactions. If neither of the previous transactions conflicts against the Tangle history, the new transaction will be attached to them as approval. To combat spam and Sybil attack, the concatenation is further hardened by mining a PoW nonce similar to Bitcoin. The new transaction is now waiting for incoming transactions' approval. The transaction itself is the smallest unit and does not form a block.

Byteball DAG (also called Obyte) is similar to the IOTA Tangle. The "transaction" in Tangle becomes a "storage unit" in Byteball. A storage unit includes the data to be stored, its creator's signature, and hash references to the parents. Byteball financially encourages a unit to include as many recent parents as possible. Byteball's solution against double-spending is to create a manually constructed universal time, the post-assigned index of a unit assigned by the main chain. When two conflicting transactions are both submitted to DAG, they are both accepted and extended. Byteball proposes a mechanism that can choose the main chain. Childless units greedily choose the best parent all the way to the genesis, then index all nodes in the main chain reversely. After both conflicting units being indexed by the main chain, Byteball requires newer units to choose the earlier unit

(unit with smaller index) to extend. Byteball further introduces the concept of witness nodes to enforce the main chain.

Swirlds Hashgraph consists of multiple parallel chains that are interconnected to each other. Each parallel chain is manipulated by a node, and the interconnection between chains represents the gossip history. This is called the "gossip through gossip" protocol, where all nodes repeat the process of randomly choosing another node, gossiping about their events, and constructing a hashgraph. The global consistency lies in the expected convergence of the hashgraph through gossiping. By executing deterministic functions, nodes can obtain a converged hashgraph composed of confirmed events. But nodes might not have a consistent view of new transactions.

In contrast, the GHOST-DAG family studied by Yonatan Sompolinsky, and Aviv Zohar does not emphasize a pure DAG-based ledger structure. Rather, it emphasizes leveraging the structural properties (multiple predecessors) of DAGs to mitigate the current blockchain's orphan rate problem. The GHOST-DAG family includes the Inclusive Block Chain Protocol [60], SPECTRE [103], GHOSTDAG [102] and PHANTOM [102] consecutively. The main difference is that the Byteball, IOTA Tangle, and Swirlds Hashgraph contain no blocks, while DAGs in the GHOST-DAG family is composed of blocks.

Inclusive Blockchain Protocol first formalizes the blockDAG model and presents an inclusive protocol. The word "inclusive" means to include off-chain transactions in the selected main chain, provided the transactions do not conflict with previously included ones. The main chain is selected by rule F, which together defines the inclusive-F protocol. Formally, F is a mapping from blockDAGs to a blockchain, such that the mapped blockchain from any subgraph in the blockDAG is always the longest. SPECTRE develops a voting algorithm regarding the order between each pair of blocks in the DAG.

In other words, SPECTRE develops a full order among all transactions, such that any two blocks are comparable to each other. Formally, fix two blocks x and y $\in$ G. The goal is to decide if $x \prec y$ or $y \prec x$. This theoretically allows deterministic solution over transaction conflicts and is essential to DAG consistency. Another important trait of SPECTRE is that the blocks vote for verification according to its position in DAG. PHANTOM made a further step to establishing a robust total order of all transactions. This is done by proving that the set of blocks created by honest miners is well-connected, and malicious nodes can't create a larger well-connected set of blocks given the assumption that attackers have less than 50% of total computational power. With this well-connected set (called the Maximum k-cluster SubDAG), PHANTOM can distinguish blocks mined properly by honest nodes and malicious nodes. However, the computation of the Maximum k-cluster SubDAG is NP-hard. So PHANTOM implements the solution using a greedy algorithm, which is named GHOSTDAG.

The advantage of DAG is that it can add transactions or blocks to a ledger efficiently due to its two-dimensional structure. However, it has imperfections that cannot be ignored. The DAG-based blockchains lack formal descriptions and rigorous security guarantees, mostly because it is hard or impossible to achieve. Note that DAG networks are always asynchronous. The timestamps of transactions cannot follow a universal clock, so the time-sequential correctness can not be guaranteed. Every node sees the whole DAG differently and intrinsically and cannot keep a consistent, universal view within a certain time boundary. As a result, each transaction has no deterministic time upper bound to be reliably finalized and can only be probabilistically accepted within infinite finalization time. This is fatal to the blockchain properties like immutability, transaction atomicity, and

unforgeability. A method to address this problem is to mandate all nodes to store the whole DAG and constantly update it. But this immediately weakens the proposed advantage of DAG being efficient, lightweight, and friendly to micro-payments.

Chapter 3: BLOWN: A Blockchain Protocol for Wireless Networks under Adversarial SINR

3.1 Introduction

Distributed Ledger Technology (DLT) refers to share, replicate, and synchronize a digital ledger across a distributed network without centralized data storage. As a widely used DLT, blockchain technologies intend to organize a digital ledger as a chain of blocks to enable remarkable properties such as decentralization, immutability, and traceability. Since Bitcoin has emerged as the first open cryptocurrency, blockchain has been envisioned as a promising technology that can be used in various practical applications such as Finance [108], Internet of Things (IoT) [6], supply chain [56], and security services [95]. In recent years, the popularity of 5G and IoT has arisen more problems of managing devices, sharing information, and carrying on computing tasks among wireless nodes [105]. Such problems become even intractable in a wireless network with small-world and super-dense features [25]. To overcome these challenges, effort has been made to build secure and trusted computing environments such as mobile edge computing (MEC) enabled blockchain [116] and the blockchain empowered 5G [29] in wireless networks taking advantage of blockchain technologies.

However, previous studies on blockchain-enabled wireless networks mostly focus on proposing architectures or building systems on top of popular blockchain protocols that are previously employed on the Internet. From the perspective of resource utilization, existing popular blockchain protocols can be divided into three categories, namely proof of physical resources, proof of virtual resources, and message passing. Proof of physical

resources requires the users to compete for proposing blocks by demonstrating their utilization of physical resources. For example, PoW-based blockchain protocols adopt the concept of proof of work, in which miners can generate blocks by solving a hash puzzle. The idea behind PoW is to randomly select a miner for block construction based on computational power. The ingenious design of PoW ensures that the leader is selected at a moderate rate, thus preventing the blockchain from double-spending and Sybil attacks. Many protocols utilize PoW, including Bitcoin [78], Ethereum [114], Byzcoin, to name a few. However, the principal concern about PoW is the large amount of power consumption[1]. Thus alternatives appear as substitutes for PoW, which include Proof-of-Space [36], Proof-of-Burn (PoB) [48], Proof-of-Elapsed Time (PoET) [83], and Proof-of-Space Time (PoST) [76]. A common drawback of such blockchain protocols lies in their prohibitively large demands of physical resources such as high computational power, storage, energy, or specific hardware, of which devices in wireless networks are limited.

As the second category, proof of virtual resources intends to show the utilization of virtual resources such as reputation, stake, and predefined weight. For example, Proof-of-Stake (PoS) aims to address the intensive power consumption of PoW and resorts to stakes as voting rights rather than computational powers. Several state-of-the-art blockchains have adopted this idea as a part of their protocols such as Algorand [42], Ouroboros [51], Casper [20]. Another example is Proof-of-Authority (PoA) proposed by Gavin Wood [8], in which blocks are proposed by approved authorities, also known as validators. Proof-of-Reputation (PoR) is a subset of PoA but is more secure and reliable since each node is assigned a more objective

[1]In November 2019, it is estimated that Bitcoin consumes more than 26000 MW of power, compatible with 26 nuclear power station units (1000 MW on average)

reputation [72], and a node can propose blocks only when its reputation meets certain requirements. In the design of our BLOWN protocol, we are partially enlightened by PoS, but BLOWN is more suitable for wireless networks because of the proposed PoC consensus protocol. In addition, BLOWN completely avoids the centralization problem caused by introducing over-powerful validators or authorities.

The third category is based on message passing, which includes Ripple [98], ELASTICO [69], Algorand [42], PBFT [23], HoneyBadgerBFT [73], Tendermint [59], and Hotstuff [117]. These brilliant schemes provide a blockchain system with liveness, persistence, security, and even robustness to Byzantine failures. However, the following two issues need to be addressed when adopting them for wireless networks. First, message passing could lead to non-trivial communication complexity, which is at least $O(N)$; and some protocols such as PBFT and Tendermint even have complexity of $O(N^2)$, where N is the network size. Second, N should be publicly known for protocol initialization as it determines how many Byzantine nodes the protocol can tolerate. Nevertheless, in wireless networks that typically have dynamic topologies, the exact N is hardly available.

To design an efficient and secure blockchain protocol for wireless networks, we confront three major challenges: 1) achieving high performance considering limited physical resources (computational power, storage, energy, specific requirements on hardware, etc.); 2) having low communication complexity even confronting interference or jamming attacks; and 3) being secure against various attacks such as double-spending and Sybil.

To simultaneously overcome these challenges, we propose the BLOWN (BLOckchain protocol for Wireless Networks) protocol. BLOWN is a two-phase protocol that does not require proof of physical resources but adopts

a new concept, Proof-of-Channel (PoC). The idea behind PoC is to let nodes compete for communication channels to win the rights of proposing blocks. Since channel competition is a natural property of wireless networks, we brace it as a part of our PoC to reduce the extra communication cost caused by the consensus process.

In this chapter, we consider that an adversary can make adversarial jamming on the nodes but controls no more than 50% wealth of the network. The wealth of the BLOWN network is the total number of coins held by all users. BLOWN is a provably secure system that satisfies two formal security properties: persistence and liveness. Persistence means that if an honest node proclaims a transaction as stable, other honest nodes, if queried, either report the same result or report error messages. Liveness, on the other hand, states that the transactions originated from the honest nodes can eventually be added to the blockchain. To prove BLOWN's properties, we formally model it with a universally composable (UC) framework and analyze it accordingly. Note that it is worthy of emphasizing that PoC can adapt to multi-hop wireless networks if combined with existing methods such as distributed spanner construction [119] or supported by an adequate routing layer [10].

Our main contributions are summarized as follows.

1. To the best of our knowledge, BLOWN is the first provably secure protocol that is particularly designed for wireless networks under a realistic adversarial SINR model.

2. The communication features of wireless networks are utilized to develop a novel, general Proof-of-Channel (PoC) consensus protocol, which leverages the natural properties of wireless networks, including broadcast communication and channel competition.

3. We develop a UC-style protocol for BLOWN and formally prove BLOWN's persistence and liveness properties by showing that it satisfies concrete chain growth, common prefix, chain quality properties. Finally, simulations are conducted to demonstrate our theoretical analysis.

3.2 Models and Assumptions

Network Model. In this chapter, we consider a network with a set V of N nodes arbitrarily deployed in a communication space. Such a network could contain a group of manipulated Unmanned Aerial Vehicles (UAVs) or intelligent robots in realistic scenarios. A node is equipped with a half-duplex transceiver that can transmit or receive a message, or sense the channel, but cannot transmit and receive or transmit and sense simultaneously. Let $d(u,v)$ be the Euclidean distance between nodes u and v, $D_R(v)$ denote the disk centered at v with a radius R, and $N_R(v)$ denote the set of nodes within $D_R(v)$ including v.

We assume that each node knows the identities, locations and public keys of all other nodes. We further assume that each node can generate key pairs and has access to a secure EUF-CMA digital signature scheme (details of cryptographical tools employed in this chapter are presented in protocol analysis and simulation sections). Each node maintains a hash-chain of blocks, and each block contains multiple transactions. We denote frequently-used notations of transaction, block, blokchain, chain of block headers by tx, B, BC, and BH, respectively, and use super/subscript to attach more specific information. A transaction is modeled as a coin exchange process. We adopt the notion of the unspent transaction outputs (UTXOs) accounting method. A UTXO defines an output of a blockchain transaction that has not been spent. An UTXO-based account stores coins in a set of

UTXOs. The UTXO model provides salient properties such as high scalability and strong security. Since UTXOs can be processed simultaneously, they support multi-thread transaction processing and therefore UTXO-based blockchains can be easily scaled through sharding [86]. Additionally, it is convenient to authenticate all transaction sequences using UTXOs, limiting the risk of double-spending attacks.

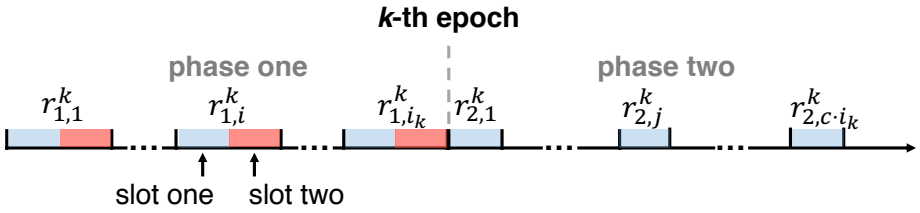


Figure 3.1: Epoch-based execution.

Interference Model. We adopt the Signal-to-Interference-plus-Noise-Ratio (SINR) wireless network model, which captures the network interference in a more realistic and precise way than a graph-based one [121]. A standard SINR model can be formulated as follows, which states that a message sent by u is correctly received by v if and only if

$$SINR = \frac{\mathscr{S}}{\mathscr{I} + \mathscr{N}} \geq \beta \tag{3.1}$$

where $\mathscr{S} = P \cdot d(u,v)^{-\alpha}$ is the received signal power of node v from u, P is the uniform transmit power, $\mathscr{I} = \sum_{w \in W \setminus \{u\}} P \cdot d(w,v)^{-\alpha}$ is the interference at v and W is the set of nodes that transmit in the current round, $\mathscr{N}$ is the ambient noise, $\alpha \in (2,6]$ is the path-loss exponent, and threshold $\beta > 1$ is determined by hardware. To capture a fine-grained noise, we define $\mathscr{N} = \mathscr{ADV}(v)$, where $\mathscr{ADV}(v)$ is the composite noise generated by the environment and adversaries. Assume each node uses a common noise threshold θ, and the distance between any two nodes is bounded by $R_0 = (P/\beta\theta)^{1/\alpha}$. We further

assume that each node can perform physical carrier sensing. If at least one neighboring node u of v broadcasts a message, v would either receive a message or sense a busy channel. At each slot, a node v may either (a) sense an idle channel (the measured signal power is lower than θ); (b) receive a message (the measured signal power exceeds θ and $SINR \geq \beta$); or (c) sense a busy channel (the measured signal power exceeds θ but $SINR < \beta$). Let $RSS = \mathscr{S} + \mathscr{I} + \mathscr{N}$ be the total received signal power at a node. Then when the node receives a message, the interference plus noise can be calculated by $\mathscr{I} + \mathscr{N} = RSS - \mathscr{S}$ with a known $\mathscr{S}$ [46, 124, 84, 47, 1].

Epoch-based Execution. As shown in Fig. 3.1, the BLOWN protocol is executed in disjoint and consecutive time intervals called *epochs*, and at each epoch no more than one block can be generated. Each epoch e_k consists of two phases with each containing multiple rounds. In e_k, we denote $r^k_{1,i}$ as the i-th round in phase one and $r^k_{2,j}$ as the j-th round in phase two, with $r^k_{1,i}$ consisting of two slots and $r^k_{2,j}$ having only one slot. Besides, i_k is the length of phase one and $c \cdot i_k$ is the length of phase two, where c is a variable constant determined later.

Adversary. We assume that there exists a group of adversaries who can freely join or leave the network, create identities, or make noises to interfere with any honest node at any time. For simplicity, the group of adversaries can be regarded as a powerful adversary $\mathscr{A}$ who controls less than 50% wealth of the entire network. $\mathscr{A}$ can launch jamming attacks. To leave a chance for an honest node to communicate, $\mathscr{A}$ is $((1-\varepsilon), T)$-bounded at any time interval of length T rounds, where $T \in \mathbb{N}$ and $0 < \varepsilon \leq 1$, indicating that the super adversary can jam nonuniformly at most $(1-\varepsilon)T$ rounds within T. Each node v maintains a variable T_v, which is the estimate of T by v.

In this chapter, we say that event E occurs with high probability (w.h.p.)

Table 3.1: Summary of Notations (BLOWN)

Symbol	Description
B_v^k	block generated by node v in the k-th epoch
BC_v^k	blockchain locally stored at node v
BH_v^k	block headers read from BC_v^k
$P_1(P_2)$	phase one (phase two)
V	set of all nodes
N	network size
c	constant used to determine the length of phase two
c_v	counter variable used to record round information
i_k	length of P_1 of the k-th epoch
tx	a transaction
txp_v	temporary transaction stack of node v
p_v	v's probability of sending a message
p_V	aggregated probability of all nodes
$\hat{p}$	an upper bound of p_v
$r_{1,i}^k(r_{2,j}^k)$	i-th (j-th) round of $P_1(P_2)$ in the k-th epoch
l_v	$l_v \in \{0,1,\cdots,w_v\}$ as the leader counter of v
l_v^0	initial value of l_v generated by *Sortition*()
T	time window of the adversary
T_v	estimate of T by node v
w_v	deposit of node v
π_v	proof created by *Sortition*()
ε	proportion of non-jammed rounds
τ	the parameter to determine the hardness of Sortition

if for any $c \geq 1$, E occurs with probability at least $1-1/N^c$, and with moderate probability (w.m.p.) if for any $c \geq 1$, E occurs with probability at least $1-1/\log^c N$. A summary of all important notations (including the ones from the BLOWN protocol and the protocol analysis) and their semantic meanings is provided in Table 3.1.

3.3 The BLOWN protocol

In this section, we present the two-phase BLOWN protocol. We first summarize the BLOWN protocol by providing the overview of BLOWN and its construction primitives, and then detail the protocol itself.

3.3.1 Overview and Utilities of BLOWN

In this subsection, we present an overview on BLOWN, and describe the construction primitives/utilities to more precisely and concisely illustrate the BLOWN protocol.

3.3.1.1 An Overview on BLOWN

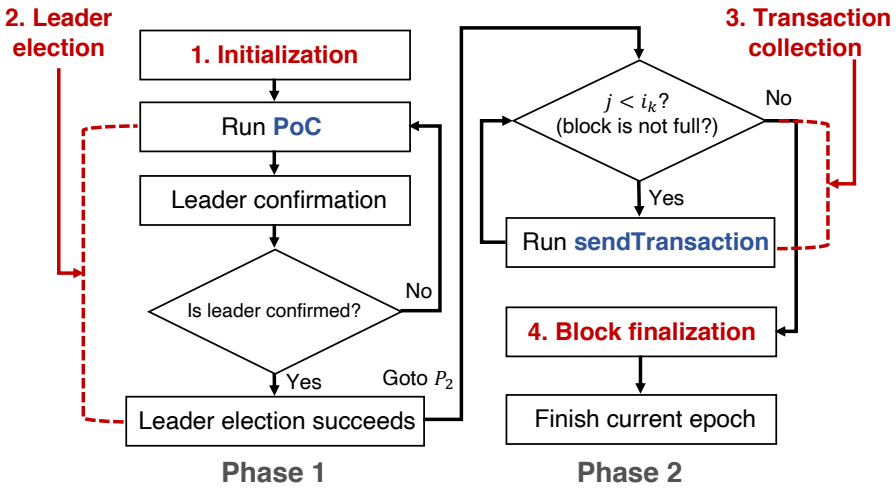


Figure 3.2: BLOWN protocol overview.

The BLOWN protocol proceeds in epochs, with each constructing no more than one block. Specifically, our protocol has two phases, denoted by P_1 and P_2 as shown in Fig. 3.2. P_1 is responsible for initialization and leader election while P_2 is for transaction collection and block finalization. In our design, nodes contend by broadcasting messages on a wireless channel. In response, we establish a robust jamming-resistant channel by introducing an adaptive transmission mechanism, confronting background noise and jamming simultaneously. Such a channel is realized by dynamically adjusting the transmission probability p_v of each node v according to its sensed contention in the network. Concretely, we first adopt the Sortition

Algorithm 1 Utilities for node v

function *SORTITION*(sk_v, seed, role, τ, w_v, W)
  h_v, π_v = VRF(sk_v, seed||role)
  $p = \tau/W$, l_v =*LeaderCounter*(role, w_v, h_v, p)
  return h_v, π_v, l_v
function *VERIFYSORTITION*(pk_v, h_v, π_v, seed, role, τ, w_v, W, l_v)
  if VerifyVRF(pk_v, h_v, π_v, seed||role) $= 0$ **then**
    return 0
  $p = \tau/W$, $\hat{l}_v$ =*LeaderCounter*(role, w_v, h_v, p)
  if $\hat{l}_v \neq l_v$ **then**
    return 0
  return 1
function *LEADERCOUNTER*(role, w_v, h_v, p)
  $l_v = 0$
  if role is `FOLLOWER` **then**
    return l_v
  if $\frac{h_v}{2^l} \in \left[0, \sum_{k=0}^{l_v} B(k; w_v, p)\right]$ **then**
    return l_v
  $l_v = 1$
  while $\frac{h_v}{2^l} \notin \left(\sum_{k=0}^{l_v-1} B(k; w_v, p), \sum_{k=0}^{l_v} B(k; w_v, p)\right]$ **do**
    $l_v = l_v + 1$
  return l_v
function *MSG*($r_{1,i}^k$, l_v)
  $m.r_{1,i}^k \leftarrow r_{1,i}^k$
  $m.l_v \leftarrow l_v$
  return m
function *MSGT*(tx, $r_{2,j}^k$, l_v)
  $m_T.tx \leftarrow tx$
  $m_T.r_{2,j}^k \leftarrow r_{2,j}^k$
  $m_T.l_v \leftarrow l_v$
  return m_T
function *MSGB*(BC_v^k, B_v^k, $r_{2,j}^k$, l_v, role, w_v, h_v, π_v, l_v')
  $m_B.BH_v^k \leftarrow$ read BH_v^k from BC_v^k
  $m_B.B_v^k \leftarrow B_v^k$
  $m_B.r_{2,j}^k \leftarrow r_{2,j}^k$
  $m_B.l_v \leftarrow l_v$
  $m_B.sort \leftarrow \{\text{role}, w_v, h_v, \pi_v, l_v'\}$
  return m_B
function *PACK*(txp_v) ▷ Pack txp to form a block
  return B_v^k
function *APPEND*(BC_v^{k-1}, B_u^k) ▷ append the new block
  return $BC_v^k \leftarrow BC_v^{k-1} + B_u^k$

algorithm to assign v a weight l_v based on its account balance. Sortition ensures that splitting coins to generate massive identities cannot break our protocol. After initialization, the protocol starts the process of leader election. We utilize the nature of contention in a wireless network to design our proof-of-channel consensus mechanism (PoC). To achieve usability and efficiency, PoC allows nodes to compete on the channel right-of-use to obtain opportunities of proposing blocks rather than rely on extra physical resources or introduce communication overhead. Concretely, upon receiving a message, l_v is decremented. The sole survivor with non-zero l_v at the end of P_1 is appointed as the leader. This essentially integrates leader election and channel contention into a single process, namely the phase one of BLOWN. In P_2, the leader is responsible for collecting and verifying transactions, assembling them into a new block, and then broadcasting the block to the whole network. If the new block is valid, it is admitted by all honest nodes.

3.3.1.2 Utilities

Algorithm 1 lists the following frequently used functions for any node v in BLOWN: *Sortition*(), *VerifySortition*(), *LeaderCounter*(), *MSG*(), *MSGT*(), *MSGB*(), *Pack*(), and *Append*(). It also presents the following data structures employed by the above functions: transaction tx, transaction stack txp_v, block B_v^k, block header BH_v^k, blockchain BC_v^k, basic message m, transaction message m_T, and block message m_B.

Concretely, we modify the crytographic Sortition algorithm proposed by Algorand[42] to make it suitable for our BLOWN protocol. The Sortition algorithm is based on a verifiable random function (VRF), which takes as inputs a private key sk_v, a random seed and a role, and outputs a hash h_v as well as its corresponding proof π_v. There are two types of roles: a `FOLLOWER`

who can only be a follower during an epoch and a LEADER who is a potential leader. Besides, W is the accumulated number of coins of all users in the network, w_v is the deposit of node v, $l_v \in \{0,1,\cdots,w_v\}$ is the leader counter of the node v, and $p = \frac{\tau}{W}$ is the probability based on which each coin is used to increment the counter value where τ determines the hardness. The probability of $l_v = k$ follows the binomial distribution $B(k;w_v,p) = \binom{w_v}{k} p^k (1-p)^{w_v-k}$ with $\sum_0^{w_v} B(k;w_v,p) = 1$. To determine l_v, the *LeaderCounter*(role, w_v, h_v, p) divides $[0,1]$ into consecutive intervals as $I(l_v) = \left[0, \sum_{k=0}^{l_v} B(k;w_v,p)\right]$ for $l_v = 0$ and $I(l_v) = \left(\sum_{k=0}^{l_v-1} B(k;w_v,p), \sum_{k=0}^{l_v} B(k;w_v,p)\right]$ for $I(l_v) \in \{1,\cdots,w_v\}$. If v's role is FOLLOWER, $I(l_v) = 0$; otherwise, if the normalized hash $\frac{h_v}{2^l}$ (l is the hash length) falls in the interval $I(l_v)$, l_v is returned as the value of the leader counter. The function *VerifySortition*() intends to check if h_v, π_v, l_v are valid by calling *VerifyVRF*() and recomputing *LeaderCounter*().

Three functions, namely *MSG*(), *MSGT*(), and *MSGB*(), generate messages that can respectively be used for leader election, transaction collection, and block finalization. Specifically, *MSG*() creates a basic message m for leader election in P_1, *MSGT*() produces a message m_T embodying a transaction, which is sent during the transaction collection process in P_2, and *MSGB*() outputs a message m_B which contains a B_v^k generated by the leader v, a BH_v^k read from BC_v^k, the current value l_v of the leader counter, and a string $\{\text{role}, w_v, h_v, \pi_v, l_v'\}$ used to verify Sortition where l_v' is the original value of the leader counter. To reduce communication cost, we send BH_v^k embodied in m_B for a simplified verification. Finally, *Pack*(txp_v) is adopted to validate and pack transactions to form a new block, and *Append*(BC_v^{k-1}, B_u^k) appends the new block B_u^k to the local blockchain BC_v^{k-1}.

3.3.2 The BLOWN Protocol Specifications

In a nutshell, BLOWN is a two-phase protocol. As shown respectively in Algorithm 3 and Algorihtm 5, phase P_1 is employed for initialization and leader election while phase P_2 is for transaction collection and block finalization.

3.3.2.1 Phase P_1

Algorithm 2 PoC subroutine

if v decides to send a message based on p_v **then**
  $m \leftarrow MSG(r^k_{1,i}, l_v)$, v broadcasts (m, σ)
else
  if channel is idle **then**
    $p_v = \min\{(1+\gamma)p_v, \hat{p}\}$
    $T_v = \max\{1, T_v - 1\}$
  else
    if v receives a message (m, σ) **then**
      $p_v = (1+\gamma)^{-1} p_v$
      $l_v = l_v - 1$
$c_v = c_v + 1$
if $c_v \geq T_v$ **then**
  $c_v = 1$
  if there is no idle rounds in the past T_v rounds **then**
    $p_v = (1+\gamma)^{-1} p_v$,
    $T_v = T_v + 2$

Let's examine the details of the BLOWN P_1 protocol. Lines 2-3 of Algorithm 3 constitute the initialization process. First, *Sortition*() takes as inputs sk_v, seed||role, τ, w_v and W (see Line 2), and outputs h_v, π_v and l^0_v, where h_v and π_v are respectively a hash and its corresponding proof, and $l^0_v \in \{0, 1, \cdots, w_v\}$ stands for the initial leader counter. Note that $l_v > 0$ indicates that v remains to be a potential leader while $l_v = 0$ indicates that v is a follower. Let $\hat{p}$ be the maximum transmission probability, which can be initialized to any small number in $(0, 1)$. Since the absence of followers

Algorithm 3 BLOWN P_1 protocol

▷ Initialization
$h_v, \pi_v, l_v^0 =$*Sortition*(sk_v, seed||role, τ, w_v, W)
$p_v = \hat{p}$, $c_v = 0$, $T_v = 1$, $i = 1, l_v = l_v^0$
▷ Leader election
while TRUE **do**
    if $l_v > 0$ **then** ▷ As a potential leader
        ▷ slot one of $r_{1,i}^k$
        run **PoC**
        ▷ slot two of $r_{1,i}^k$
        if v broadcasts a message in slot one **then**
            v listens on the channel
            if v senses an idle channel **then**
                Goto P_2 with $i_k = i$ ▷ run P_2 as a leader
        else
            $m \leftarrow MSG(r_{1,i}^k, l_v)$, and v broadcasts (m, σ)
    else ▷ As a follower
        ▷ slot one of $r_{1,i}^k$
        v listens on the channel to receive a message
        ▷ slot two of $r_{1,i}^k$
        if in slot one v receives (m, σ) from u and has $\mathscr{I} + \mathscr{N} < \theta$ **then**
            if v senses an idle channel **then**
                v recognize u as the leader
                Goto P_2 with $i_k = i$ ▷ run P_2 as a follower
        else
            $m \leftarrow MSG(r_{1,i}^k, l_v)$, and v broadcasts (m, σ)
    $i = i + 1$

might lead to a bad case in which all nodes are potential leaders and simultaneously broadcast messages in slot one[2], we prevent this from occurring by ensuring that there always exist at least one follower after initialization. A simple approach to achieving this goal is to artificially and randomly add followers (with a FOLLOWER rule) to the network. Second, we set $p_v = \hat{p}$, $c_v = 0$, $T_v = 1$, $i = 1, l_v = l_v^0$ (Line 3), where p_v is the probability on which node v decides to send a message, and is upper-bounded by $\hat{p}$, $c_v = 1$ is a counter variable used to record round information, T_v is the estimate of the time window of the adversary by node v, i is the round counter used in P_1, and l_v is the leader counter variable initialized to l_v^0. After initialization, P_1 proceeds round by round with each containing two slots, and a node v's activity at each slot depends on its role.

Before proceed, we need to explain the PoC subroutine described in Algorithm 2 to adjust l_v, p_v and T_v, the leader counter, transmission probability and adversary's time window estimate, according to the sensed channel condition at the first slot of each round in P_1. Specifically, v with $l_v > 0$ (a potential leader) performs the following actions: it either broadcasts a message (m, σ) with probability p_v (Lines 1-2), where σ is the signature of m, or senses the channel with probability $1 - p_v$ (Lines 3-10). One can see that v adapts its p_v in a multiplicative increase or decrease manner by a factor of $(1+\gamma)$, where $\gamma = O(1/(\log T + \log\log N))$ is a small number that is loosely determined by N and T. Particularly, p_v is multiplicatively increased (Line 5) when the channel is sensed idle or decreased (Line 9) when a message is received [3]. Such a mechanism ensures that honest nodes can cooperatively adjust their transmission probabilities to help reduce contention on the

[2]Such a bad case only occurs with a small probability, which is less than $\hat{p}^n$

[3]Receving a message indicates the message has a valid signature, and we do not explicitly present the signature verification process for conciseness.

channel. Meanwhile, we decrease T_v by 1 if the channel is idle (Line 6) as the estimate of adversary's time window seems to be too large when the channel is idle, and decrease l_v by one if a neighbor of v successfully broadcasts a message (Line 10) as the neighbor seems to have a better chance of being the leader. On the other hand, if the number of rounds in P_1 is no less than T_v (Line 12), the estimate of adversary's time window, we further check whether or not there is an idle round in the past T_v rounds (Line 14), and if not, p_v is decreased (Line 15) and T_v is increased (Line 16) to further adjust p_v and T_v. One can see that a successful broadcast causes the decrements of the l_v values of the receivers. When $l_v = 0$, v becomes a follower who can only sense the channel in slot one of the next round.

Now we are back to continue explaining Phase P_1 of the BLOWN protocol, which contains multiple rounds. At slot one of each round, if v is a potential leader, which means $l_v > 0$, v runs the PoC subroutine described in Algorithm 2 (Line 8); otherwise, v listens on the channel for message reception (Line 18). At slot two of each round, v behaves according to its actions in slot one. If v as a potential leader broadcasts a message in slot one and senses an idle channel in slot two, it can set itself as a leader and goto P_2 (Lines 10-13); otherwise it broadcasts a message in slot two. A follower v recognizes u as the leader only when v believes u is the only transmitter in slot one and senses an idle channel in slot two (Lines 20-23); otherwise v transmits in slot two (Line 25). In Theorem 4, we prove that slot two is capable of letting the leader and the followers mutually recognize each other.

At the end of P_1, there should be only one survivor with $l_v > 0$, who then becomes the leader. Note that i_k denotes the length of P_1, which is used to determine the length of P_2. We will prove in Theorem 4 of Section 3.4 that Algorithm 3 ensures a successful leader election.

Algorithm 4 sendTransaction subroutine

if v decides to send a message based on p_v **then**
  $m_T \leftarrow MSGT(tx, r^k_{2,j}, l_v)$, and v broadcasts (m_T, σ_T)
else
  if channel is idle **then**
    $p_v = \min\{(1+\gamma)p_v, \hat{p}\}$
    $T_v = \max\{1, T_v - 1\}$
  else
    if receives a message (m_T, σ_T) **then**
      $p_v = (1+\gamma)^{-1} p_v$
$c_v = c_v + 1$
if $c_v \geq T_v$ **then**
  $c_v = 1$
  if there is no idle round in the past T_v rounds **then**
    $p_v = (1+\gamma)^{-1} p_v$,
    $T_v = T_v + 2$

3.3.2.2 Phase P_2

Algorithm 5 BLOWN P_2 protocol

1: ▷ Transaction collection
2: **while** $j < c \cdot i_k$ **do**
3: **if** $l_v > 0$ **then** ▷ As a leader
4: v listens on the channel to receive a (m_T, σ_T)
5: **if** receives m_T.tx$\neq\perp$ **then**
6: $txp_v[j] = m_T.tx$
7: **else** ▷ As a follower
8: run **sendTransaction**
9: $j = j + 1$
10: ▷ Block finalization
11: **if** $j = c \cdot i_k$ **then**
12: **if** $l_v > 0$ **then** ▷ As a leader
13: $B^k_v \leftarrow Pack(txp_v)$
14: $BC^k_v \leftarrow Append(\text{BC}^{k-1}_v, B^k_v)$
15: $m_B \leftarrow MSGB(BC^k_v, B^k_v, r^k_{2,j}, l_v,$ role, $w_v, h_v, \pi_v, l^0_v)$, and broadcasts (m_B, σ_B)
16: **else** ▷ As a follower
17: **if** receives (m_B, σ_B) && $VerifySortition$ $(pk_v,$ seed, $\tau, W, m_B.sort) = 1$ **then**
18: $Append(\text{BC}^{k-1}_v, m_B.B^k_u)$

Phase P_2 of BLOWN performs transaction collection and block finalization, as shown in Algorithm 5. It proceeds by a fixed amount of $c \cdot i_k$ rounds where each round contains only one slot, and c is a constant to directly determine the length of P_2 and indirectly the maximum block size, which can be adjusted according to specific implementations. We refer to j as the round counter in P_2. If $j < c \cdot i_k$, a leader selected in P_1 should listen to the channel to receive signed transaction messages, which are recorded in the stack txp_v, while other nodes continuously broadcast signed transaction messages (Lines 2-8). After $c \cdot i_k$ rounds, the leader serializes all transactions to form a new block denoted by $B_v^k \leftarrow Pack(txp_v)$, and broadcasts the (m_B, σ_B) (Lines 12-15). Once receiving a (m_B, σ_B) from u, a node v should append the new block to its local blockchain only if σ_B is valid and *VerifySortition* (pk_v, seed, τ, W, $m_B.sort$) $= 1$ (Lines 17-18). Note that the sendTransaction subroutine presented in Algorithm 4 is employed by P_2 to broadcast transactions and the parameters p_v, c_v, T_v are utilized to ensure jamming-resistant communications as they function in the PoC subroutine shown in Algorithm 2.

3.4 Protocol Analysis

Proving security properties of a complex protocol such as BLOWN is very challenging. Thus we leverage the universally composable (UC) framework proposed by Canetti *et al.* [21]. The UC framework captures the security of a protocol via emulating an idealized protocol $\mathscr{F}$ (often referred to as an ideal functionality), which satisfies strong security properties. Then a real protocol π specifying concrete implementations is said to be secure if it is indistinguishable from $\mathscr{F}$. The main feature of the UC framework is the universal composability that allows one to perform analysis on a complex

Protocol $\pi_{\mathbf{B}}[\mathscr{F}_{\mathbf{SIG}}, \mathscr{F}_{\mathbf{SORT}}]$

π_{B} is a protocol run by all nodes interacting with the ideal functionalities $\mathscr{F}_{\mathrm{SIG}}$ and $\mathscr{F}_{\mathrm{SORT}}$.

Initialization: Send (sk_v, $seed$, $role$, τ, w_v, W) to $\mathscr{F}_{\mathrm{SORT}}$, which returns (h_v, π_v, l_v^0). Next initialize the remaining local parameters as $p_v = \hat{p}$, $c_v = 0$, $T_v = 1$, $i = 1, l_v = l_v^0$.

Leader election: For each round $r_{1,i}^k$ of P_1 during the k-th epoch, perform the following (1) or (2) according to the value of l_v:

(1) If $l_v > 0$, run PoC in slot one. If broadcasting a message in slot one, listen on the channel in slot two and if the channel is idle, goto P_2 with $i_k = i$ at the end of slot two; otherwise, send $m \leftarrow MSG(r_{1,i}^k, l_v)$ to $\mathscr{F}_{\mathrm{SIG}}$, which returns a signed message (m, σ), i.e., (m, σ) is obtained by querying $\mathscr{F}_{\mathrm{SIG}}$, then broadcast (m, σ) in slot two.

(2) If $l_v = 0$, listen on the channel in slot one. If receiving a valid (m, σ) from u with $\mathscr{I} + \mathscr{N} < \theta$ in slot one, and sensing an idle channel in slot two, recognize u as the leader and goto P_2 with $i_k = i$ at the end of slot two; otherwise, generate $m \leftarrow MSG(r_{1,i}^k, l_v)$, send m to $\mathscr{F}_{\mathrm{SIG}}$, which returns (m, σ), then broadcast (m, σ) in slot two. Note that a valid m holds when $\mathscr{F}_{\mathrm{SIG}}$ returns 1 upon being queried with (m, σ).

Transaction collection: At each round $r_{2,j}^k$, if $l_v > 0$, listen on the channel for a possible signed transaction message m_T, add the transaction to the local stack as $txp_v[j] = m_T.tx$ if receiving (m_T, σ_T) and $\mathscr{F}_{\mathrm{SIG}}$ returns 1 when being queried with (m_T, σ_T). If $l_v = 0$, send $m_T \leftarrow MSGT(tx, r_{2,j}^k, l_v)$ to $\mathscr{F}_{\mathrm{SIG}}$, which returns a signed message (m_T, σ_T), then broadcast (m_T, σ_T).

Block finalization: During the round $r_{2,c\cdot i_k}^k$, if $l_v > 0$, execute $B_v^k \leftarrow Pack(txp_v)$ and $BC_v^k \leftarrow Append(\mathrm{BC}_v^{k-1}, B_v^k)$; then generate $m_B \leftarrow MSGB(BC_v^k, B_v^k, r_{2,j}^k, l_v, \mathrm{role}, w_v, h_v, \pi_v, l_v^0)$ and send it to $\mathscr{F}_{\mathrm{SIG}}$, which returns (m_B, σ_B). If $l_v = 0$, listen on the channel for a possible block message (m_B, σ_B); if receiving a valid (m_B, σ_B), which means $\mathscr{F}_{\mathrm{SIG}}$ returns 1 upon being queried with (m_B, σ_B), and $\mathscr{F}_{\mathrm{SORT}}$ returning 1 upon being queried with (pk_v, seed, τ, W, $m_B.sort$), execute $Append(\mathrm{BC}_v^{k-1}, m_B.B_u^k)$.

Figure 3.3: The protocol (hybrid experiment) $\pi_{\mathrm{B}}[\mathscr{F}_{\mathrm{SIG}}, \mathscr{F}_{\mathrm{SORT}}]$.

protocol, whose security properties can be derived from the security of its components.

3.4.1 UC Composition of BLOWN

We formulate two UC-style protocols (or hybrid experiments), which are presented in Fig. 3.3 and Fig. 3.4. The $\pi_{\mathrm{B}}[\mathcal{F}_{\mathrm{SIG}}, \mathcal{F}_{\mathrm{SORT}}]$ conducts a hybrid experiment for BLOWN using an ideal hybrid functionality $[\mathcal{F}_{\mathrm{SIG}}, \mathcal{F}_{\mathrm{SORT}}]$ where $\mathcal{F}_{\mathrm{SIG}}$ is an ideal digital signature scheme and $\mathcal{F}_{\mathrm{SORT}}$ is an ideal functionality, performing three sortition-related functions as shown in Fig. 3.5. BLOWN is denoted as $\pi_{\mathrm{B}}[\pi_{\mathrm{SIG}}, \pi_{\mathrm{SORT}}]$, which implements real protocols π_{SIG} and π_{SORT}. Besides, $\pi_{\mathrm{SORT}}[\mathcal{F}_{\mathrm{VRF}}]$ is a protocol that realizes sortition-related functionalities, consisting of LeaderCounter, Sortition, and VerifySortition. These functionalities are consistent with the corresponding ones specified in Algorithm 1 except that $\pi_{\mathrm{SORT}}[\mathcal{F}_{\mathrm{VRF}}]$ uses an ideal functionality $\mathcal{F}_{\mathrm{VRF}}$ in Sortition and VerifySortition. In contrast, Algorithm 1 adopts a realistic VRF implementation. Let $\mathcal{A}, \mathcal{Z}, \mathcal{S}$ be respectively the adversary, environment, simulator, whose specific meanings should depend on the context. We first show that the following lemma 1 holds for $\pi_{\mathrm{SORT}}[\mathcal{F}_{\mathrm{VRF}}]$.

Lemma 1. *With the same security parameter λ, for each probabilistic polynomial-time (PPT) $\{\mathcal{A}, \mathcal{Z}\}$, it holds that the protocol $\pi_{\mathrm{SORT}}[\mathcal{F}_{\mathrm{VRF}}]$ securely realizes $\mathcal{F}_{\mathrm{SORT}}$ under the $\mathcal{F}_{\mathrm{VRF}}$-hybrid model.*

Proof. Let $\mathcal{A}$ be an adversary that interacts with the nodes running $\pi_{\mathrm{SORT}}[\mathcal{F}_{\mathrm{VRF}}]$ under the $\mathcal{F}_{\mathrm{VRF}}$-hybrid model. We need to construct an ideal simulator $\mathcal{S}$ such that the view of any environment $\mathcal{Z}$ of an interaction with $\mathcal{A}$ and $\pi_{\mathrm{SORT}}[\mathcal{F}_{\mathrm{VRF}}]$ is exactly the same as that of an interaction with $\mathcal{S}$ and $\mathcal{F}_{\mathrm{SORT}}$. In our construction, the simulator $\mathcal{S}$ runs $\mathcal{A}_{\mathcal{F}_{\mathrm{VRF}}}$ (under the name of $\mathcal{F}_{\mathrm{VRF}}$) and simulates other possibly involved nodes. Here, the $\mathcal{A}_{\mathcal{F}_{\mathrm{VRF}}}$ who

is attacking the VRF function is identically defined as the one attacking the ideal functionality $\mathcal{F}_{\text{VRF}}^{Praos}$ presented in [30]. $\mathcal{S}$ is responsible for forwarding messages from $\mathcal{Z}$ and $\mathcal{A}_{\mathcal{F}_{\text{VRF}}}$. Besides, $\mathcal{S}$ performs the following operations:

1. Simulating value and proof generation: When $\mathcal{S}$ receives a message (sk_v, seed||role) in the ideal process from $\mathcal{F}_{\text{SORT}}$, it simulates for $\mathcal{A}_{\mathcal{F}_{\text{VRF}}}$ (under the name of $\mathcal{F}_{\text{VRF}}$) the process of generating (Evaluated, s_{id}, h_v, π_v), where s_{id} represents a session id which is not explicitly presented in this chapter for simplicity. $\mathcal{S}$ then forwards (h_v, π_v) to $\mathcal{F}_{\text{SORT}}$.

2. Simulating verification: When $\mathcal{S}$ receives a message (pk_v, h_v, π_v, seed ||role) in the ideal process from $\mathcal{F}_{\text{SORT}}$ meaning a verificaiton query is received, it simulates for $\mathcal{A}_{\mathcal{F}_{\text{VRF}}}$ the process of VRF verification. Once receiving (Verified, s_{id}, h_v, π_v, f), $\mathcal{S}$ forwards (pk_v, h_v, π_v, seed||role, f) to $\mathcal{F}_{\text{SORT}}$.

It is straightforward to verify that $\mathcal{S}$ perfectly simulates the adversary and other components. That is, for any PPT $\{\mathcal{A}, \mathcal{Z}\}$, $\mathcal{Z}$ cannot distinguish between its interaction with $\mathcal{A}$ and $\pi_{\text{SORT}}[\mathcal{F}_{\text{VRF}}]$ or $\mathcal{S}$ and $\mathcal{F}_{\text{SORT}}$. Thus one can draw a conclusion that $\pi_{\text{SORT}}[\mathcal{F}_{\text{VRF}}]$ securely realizes $\mathcal{F}_{\text{SORT}}$ under the $\mathcal{F}_{\text{VRF}}$-hybrid model. □

In the setting of [30], the authors elegantly proved that there exists a realistic implementation of π_{VRF} that can securely realize the ideal $\mathcal{F}_{\text{VRF}}$ under the Computational Diffie-Hellman (CDH) assumption in the random oracle model. Therefore with such a secure real-world implementation, our protocol $\pi_{\text{SORT}}[\pi_{\text{VRF}}]$, abbreviated as π_{SORT}, is computationally indistinguishable from $\pi_{\text{SORT}}[\mathcal{F}_{\text{VRF}}]$, and thus securely realizes $\mathcal{F}_{\text{SORT}}$ according to Lemma 1. Then for the analysis of the complicated BLOWN protocol, one can get rid of the repeated reduction proofs by conducting a hybrid experiment

$\pi_{\mathrm{B}}[\mathcal{F}_{\mathrm{SIG}}, \mathcal{F}_{\mathrm{SORT}}]$, where $\mathcal{F}_{\mathrm{SORT}}$ is the ideal signature scheme presented in [22]. In Section 3.4.2, we report the salient features that can be realized by $\pi_{\mathrm{B}}[\mathcal{F}_{\mathrm{SIG}}, \mathcal{F}_{\mathrm{SORT}}]$ with the ideal combinatorial functionalities $[\mathcal{F}_{\mathrm{SIG}}, \mathcal{F}_{\mathrm{SORT}}]$. Thus we need to show that the real BLOWN protocol $\pi_{\mathrm{B}}[\pi_{\mathrm{SIG}}, \pi_{\mathrm{SORT}}]$ (π_{SIG} is a secure EUF-CMA digital signature scheme) and $\pi_{\mathrm{B}}[\mathcal{F}_{\mathrm{SIG}}, \mathcal{F}_{\mathrm{SORT}}]$ are computationally indistinguishable so that $\pi_{\mathrm{B}}[\pi_{\mathrm{SIG}}, \pi_{\mathrm{SORT}}]$ can inherit all features of $\pi_{\mathrm{B}}[\mathcal{F}_{\mathrm{SIG}}, \mathcal{F}_{\mathrm{SORT}}]$.

Theorem 1. *With same security parameter λ, for each PPT $\mathcal{A}, \mathcal{Z}$ it holds that there is a PPT $\mathcal{S}$ such that*

$$EXEC^{\mathcal{A},\mathcal{Z}}_{\pi_B[\pi_{SIG},\pi_{SORT}]} \approx EXEC^{\mathcal{S},\mathcal{Z}}_{\pi_B[\mathcal{F}_{SIG},\mathcal{F}_{SORT}]}, \tag{3.2}$$

where "$\approx$" means computationally indistinguishable.

Proof. With a real digital signature protocol π_{SIG}, we obtain $\pi_{\mathrm{B}}[\pi_{\mathrm{SIG}}, \mathcal{F}_{\mathrm{SORT}}]$, which is a protocol under the $\mathcal{F}_{\mathrm{SORT}}$-hybrid model. From Lemma 1, one can see that it holds for each PPT $\mathcal{A}$ and $\mathcal{Z}$, the protocol π_{SORT} securely realizes $\mathcal{F}_{\mathrm{SORT}}$. According to the universal composition theorem, it holds that for any adversary $\mathcal{A}_{\mathcal{F}_{\mathrm{VRF}}}$, there exists an adversary $\mathcal{A}_{\mathcal{F}_{\mathrm{SORT}}}$ such that for any environment $\mathcal{Z}$, we have

$$\mathrm{EXEC}^{\mathcal{A}_{\mathcal{F}_{\mathrm{VRF}}},\mathcal{Z}}_{\pi_{\mathrm{B}}[\pi_{\mathrm{SIG}},\pi_{\mathrm{SORT}}]} \approx \mathrm{EXEC}^{\mathcal{A}_{\mathcal{F}_{\mathrm{SORT}}},\mathcal{Z}}_{\pi_{\mathrm{B}}[\pi_{\mathrm{SIG}},\mathcal{F}_{\mathrm{SORT}}]}, \tag{3.3}$$

Let $\pi_{\mathrm{B}}[\mathcal{F}_{\mathrm{SIG}}, \mathcal{F}_{\mathrm{SORT}}]$ be a protocol under the $\mathcal{F}_{\mathrm{SIG}}$-hybrid model with a fixed $\mathcal{F}_{\mathrm{SORT}}$. Making use of an EUF-CMA digital signature scheme π_{SIG} that securely realizes $\mathcal{F}_{\mathrm{SIG}}$, we have

$$\mathrm{EXEC}^{\mathcal{A}_{\mathcal{F}_{\mathrm{SIG}}},\mathcal{Z}}_{\pi_{\mathrm{B}}[\pi_{\mathrm{SIG}},\mathcal{F}_{\mathrm{SORT}}]} \approx \mathrm{EXEC}^{\mathcal{A}_{\mathcal{F}_0},\mathcal{Z}}_{\pi_{\mathrm{B}}[\mathcal{F}_{\mathrm{SIG}},\mathcal{F}_{\mathrm{SORT}}]}, \tag{3.4}$$

where $\mathscr{A}_{\mathscr{F}_0}$ is a dumb adversary. Combining (3.3) and (3.4), one can construct the simulator $\mathscr{S}$ that can run $\mathscr{A}_{\mathscr{F}_{SORT}}, \mathscr{A}_{\mathscr{F}_{VRF}}, \mathscr{A}_{\mathscr{F}_0}$ and forward messages between the adversary and $\mathscr{Z}$ so that $\mathscr{Z}$ cannot distinguish the interactions with $\pi_B[\pi_{SIG}, \pi_{SORT}]$ from those with $\pi_B[\mathscr{F}_{SIG}, \mathscr{F}_{SORT}]$. □

Protocol $\pi_{\mathbf{SORT}}[\mathscr{F}_{\mathbf{VRF}}]$

LeaderCounter: When activated with input (role, w_v, h_v, p), first initialize $l_v = 0$. If role is `FOLLOWER`, output $l_v = 0$ and exit the LeaderCounter process; Otherwise, compute $\frac{h_v}{2^l}$. Then if $\frac{h_v}{2^l}$ falls in $\left[0, \sum_{k=0}^{l_v} B(k; w_v, p)\right]$, set $l_v = 1$. Increase l_v until it satisfies the requirement that $\frac{h_v}{2^l} \in \left(\sum_{k=0}^{l_v-1} B(k; w_v, p), \sum_{k=0}^{l_v} B(k; w_v, p)\right]$. Then sets l_v to v and output l_v.
Sortition: When activated with input (sk_v, seed, role, τ, w_v, W), first input (sk_v, seed||role) to $\mathscr{F}_{\text{VRF}}$, obtaining (h_v, π_v). Then compute $p = \tau/W$ and input (role, w_v, h_v, p) to the LeaderCounter, obtaining l_v. Finally, output (h_v, π_v, l_v).
VerifySortition: When activated with input (pk_v, h_v, π_v, seed, role, τ, w_v, W, l_v), first input (pk_v, h_v, π_v, seed||role) to $\mathscr{F}_{\text{VRF}}$, obtaining the (pk_v, h_v, π_v, seed||role, f). If $f = 0$, output `FALSE` (means that verification is not passed). If $f = 1$, compute $p = \tau/W$ and input (h_v, π_v) to the LeaderCounter, obtaining $\hat{l}_v$. If $\hat{l}_v \neq l_v$, output `FALSE`; Otherwise output `TRUE` (means that verification is passed).

Figure 3.4: The protocol (hybrid experiment) $\pi_{\text{SORT}}[\mathscr{F}_{\text{VRF}}]$.

3.4.2 Persistence and liveness

We first formulate a state machine $\mathbb{S}$ with the following four states: `START`, `LEADER`, `COMMIT`, `FINAL`.

Definition 7. *(`START` State). The system is in `START` state when the following conditions hold: (1) $|\{v|v \in V, l_v > 0\}| > 1$; (2) the honest nodes that accepted B_v^{k-1} in the last epoch have finished initialization.*

Functionality $\mathscr{F}_{\text{SORT}}$

LeaderCounter: Upon receiving a (role, w_v, h_v, p) from some node v, verify that if role is FOLLOWER. If so, send $l_v = 0$ to v; If not, compute $\frac{h_v}{2^l}$. Then if $\frac{h_v}{2^l}$ falls in $\left[0, \sum_{k=0}^{l_v} B(k; w_v, p)\right]$, sends $l_v = 1$ to v. Increase l_v until it satisfies the requirement that $\frac{h_v}{2^l} \in \left(\sum_{k=0}^{l_v-1} B(k; w_v, p), \sum_{k=0}^{l_v} B(k; w_v, p)\right]$, and send l_v to v.

Sortition: Upon receiving (sk_v, seed, role, τ, w_v, W) from some node v, send (sk_v, seed||role) to the adversary, receiving (h_v, π_v).

(1) If there is no entry (sk_v, seed||role, h_v, π_v) recorded, record (sk_v, seed||role, h_v, π_v). If there is an existing entry (sk_v, seed||role, h'_v, π'_v) which satisfies $h'_v = h_v$ and $\pi'_v = \pi_v$, do nothing. Then compute $p = \tau/W$ and send (role, w_v, h_v, p) to the LeaderCounter, receiving l_v. Finally, send (h_v, π_v, l_v) to v.

(2) If there is an entry (sk_v, seed||role, h'_v, π'_v) recorded but $h'_v \neq h_v$ or $\pi'_v \neq \pi_v$, send an error message to v.

VerifySortition: Upon receiving (pk_v, h_v, π_v, seed, role, τ, w_v, W, l_v), first send (pk_v, h_v, π_v, seed||role) to the adversary, receiving the (pk_v, h_v, π_v, seed||role, f).

(1) If $f = 0$ or there is no entry (sk_v, seed||role, h_v, π_v) recorded, send 0 to v (means that verification is not passed).

(2) If $f = 1$ and there is an existing entry (sk_v, seed||role, h_v, π_v), compute $p = \tau/W$ and send (h_v, π_v) to the LeaderCounter, receiving $\hat{l_v}$. If $\hat{l_v} \neq l_v$, sends 0 to v; Otherwise send 1 to v (means that verification is passed).

Figure 3.5: The ideal functionality $\mathscr{F}_{\text{SORT}}$

Definition 8. *(`LEADER` State). The system is in `LEADER` state when the following conditions hold: (1) there is a node v with $l_v > 0$ and $\forall u \in V \setminus \{v\}, l_u = 0$; (2) $j = 0$; (3) the size of v's transaction stack $|txp_v| = 0$.*

Definition 9. *(`COMMIT` State). The system is in `COMMIT` state when the following conditions hold: (1) there is a node v with $l_v > 0$ and $\forall u \in V \setminus \{v\}, l_u = 0$; (2) $0 < j < c \cdot i_k$.*

Definition 10. *(`FINAL` State). The system is in `FINAL` state if one of the following two conditions holds: (1) each honest node v has received a valid m_B and accepted the block B_u^k; (2) honest nodes did not receive a block in the $(c+1)i_k$-th round.*

Garay *et al.* [41] proved that a secure distributed ledger should satisfy persistence and liveness properties. Let tx_i^j be the j-th transaction of the i-th block (the 0-th block is the genesis block). We say tx_i^j is *t-stable* when the current block index is larger than $i+t$, where $t > 0$. Then the persistence and liveness properties that BLOWN should guarantee can be defined as follows.

Definition 11. ***Persistence and liveness.*** *Persistence states that if an honest node v proclaims a transaction tx_i^j as t-stable, other honest nodes, if queried, either report the same result or report error messages. On the other hand, the liveness property states that if an honest node generates the transaction tx_i^j and contends to broadcast it in phase two, the BLOWN protocol can add it to the blockchain within finite epochs w.h.p.*

Kiayias and Panagiotakos [50] showed that persistence and liveness can be derived from the following three more concrete properties: chain growth, common prefix, and chain quality.

Definition 12 (Chain growth property)**.** *Consider two chains $\mathscr{C}_1, \mathscr{C}_2$ possessed by two honest nodes at the onset of two epochs $e_1 < e_2$ with e_2 at least k epochs ahead of e_1. It holds that $len(\mathscr{C}_2) - len(\mathscr{C}_1) \geq \tau \cdot k$, where τ is the speed coefficient with $\tau \in (0,1]$ and $k \in \mathbb{N}$.*

Definition 13 (Common prefix property)**.** *The chains $\mathscr{C}_1, \mathscr{C}_2$ possessed by two honest nodes at the onset of the epoch $e_1 < e_2$ satisfy $\mathscr{C}_1^{\lceil k} \preceq \mathscr{C}_2$, where $k \in \mathbb{N}$ and $\mathscr{C}_1^{\lceil k}$ denotes the chain obtained by removing the last k blocks from $\mathscr{C}_1$, and $\preceq$ denotes the prefix relation.*

Definition 14 (Chain quality property)**.** *Consider any portion of length at least l of the chain possessed by an honest party at the onset of an epoch. The ratio of the blocks originating from the adversary is at most $1-\mu$, where $\mu \in (0,1]$ is the chain quality coefficient.*

In the remainder of this section, we intend to prove that $\pi_{\mathrm{B}}[\mathscr{F}_{\mathrm{SIG}}, \mathscr{F}_{\mathrm{SORT}}]$ satisfies chain growth, common prefix, and chain quality properties, indicating that BLOWN guarantees persistence and liveness.

3.4.2.1 Chain Growth

$\pi_{\mathrm{B}}[\mathscr{F}_{\mathrm{SIG}}, \mathscr{F}_{\mathrm{SORT}}]$ meets chain growth as claimed in Theorem 2. We prove Theorem 2 by two steps: 1) each epoch must be terminated within a finite time (or $\mathbb{S}$ never enters a deadlock); 2) the chain growth property should quantify the blockchain growing speed such that new blocks are added to a chain with a speed coefficient $\tau \in (0,1]$. Concretely, we first prove that $\pi_{\mathrm{B}}[\mathscr{F}_{\mathrm{SIG}}, \mathscr{F}_{\mathrm{SORT}}]$ ensures robust communication channels, as without which the protocol can hardly proceed. With such a communication channel, $\pi_{\mathrm{B}}[\mathscr{F}_{\mathrm{SIG}}, \mathscr{F}_{\mathrm{SORT}}]$ supports a successful leader election, which provides correctness, efficiency, and practicality. Then we perform an analysis on the $\mathbb{S}$ of BLOWN to end the proof of Theorem 2.

Theorem 2. *It holds for $\pi_B[\mathcal{F}_{SIG}, \mathcal{F}_{SORT}]$ that each epoch can terminate in $O(cw_{max}\lambda)$, and there are $O(cw_{max})$ transactions added to the blockchain at each epoch w.h.p., at the speed coefficient (following Definition 12) $\tau = 0.5$.*

To start with, we need to prove Theorem 7, which states that $\pi_{\mathrm{B}}[\mathcal{F}_{\mathrm{SIG}}, \mathcal{F}_{\mathrm{SORT}}]$ can ensure a robust communication channel. Recall that the distance between any two nodes is bounded by $R_0 = (P/\beta\theta)^{1/\alpha}$. Therefore for $\forall v \in V$, $D_{R_0}(v)$ can cover all the neighbors of node v so that if at least one node $u \in N_{R_0}(v)$ transmits a message, v would either receive the message or sense a busy channel.

Theorem 3. *If $N_{R_0}(v) \neq \varnothing$, it holds true for BLOWN that runing at least $F = \Omega((T\log N)/\varepsilon + (\log N)^4/(\gamma\varepsilon)^2)$ rounds leads to at least $(1-\varepsilon\beta')\rho e^{\frac{-\rho}{1-\hat{p}}}F$ rounds of successful transmissions against any $((1-\varepsilon), T)$-bounded adversary w.h.p., where $\gamma = O(1/(\log T + \log\log N))$ and ρ is a constant.*

Proof. To prove Theorem 3, we divide $D_{R_0}(v)$ into six sectors of equal angles centered at v, and denote an arbitrary sector as S. Then we refer to $\bar{p}_v = \sum_{w\in S\setminus\{v\}} p_w$ as the aggregated transmission probability of the neighbors of v, and p_S denotes the aggregated transmission probability of all the nodes in S. Lemma 4 can be proved utilizing Lemma 2 and 3, whose proofs can be found in [92]. We divide the F into $(c\log N)/\varepsilon$ consecutive subframes, with each consisting of $c(T + (\log N)^3/(\gamma^2\varepsilon))$ rounds.

Lemma 2. *Consider any node v in S. If $\bar{p}_v > 5 - \hat{p}$ during all rounds of a subframe I' of I and at the beginning of I', $T_v \leq \sqrt{F}$, then p_v is at most $1/N^2$ at the end of I', w.h.p.*

Lemma 3. *For any subframe I in F and any initial value of p_S in I there is at least one round in I with $p_S \leq 5$ w.h.p.*

Lemma 4. *For any subframe I_k in I, if $p_S \leq 5$ occurs during the past subframe I_{k-1}, $p_S \leq 5e^2$ holds throughout I_k w.m.p.*

Proof. Let p_S^t be the cummulative transmission probability of nodes in S at round t. Assume the probability that all nodes in S are not transmitting is q_0, the probability that only one node in S is transmitting is q_1, and the probability that at least two nodes in S are transmitting is q_2. Then one can obtain the upper bound of the expectation of p_S^{t+1} as follows:

$$E[p_S^{t+1}] \leq q_0(1+\gamma)p_S^t + q_1(1+\gamma)^{-1}p_S^t + q_2 \cdot p_S^t. \tag{3.5}$$

This upper bound holds true even if we consider the rounds when $c_v > T_v$, which decreases p_S. Let E_2 be the event when at least two nodes in S transmit. If E_2 does not happen, $q_2 = 0$ and Eq. (3.5) becomes

$$E[p_S^{t+1}] = \frac{q_0}{q_0+q_1}(1+\gamma)p_S^t + \frac{q_1}{q_0+q_1}(1+\gamma)^{-1}p_S^t. \tag{3.6}$$

If $p_S > 5$, we have $q_1 \geq p_S \cdot q_0 \geq 5q_0$. Hence,

$$E[p_S^{t+1}] \leq [\frac{(1+\gamma)}{6} + \frac{5(1+\gamma)^{-1}}{6}]p_S^t \leq (1+\gamma)^{-1/2}p_S^t. \tag{3.7}$$

Considering the case where E_2 might happen, one can rewrite $E[p_S^{t+1}]$ as

$$E[p_S^{t+1}] \leq [q_2 + (1-q_2)(1+\gamma)^{-1/2}]p_S^t. \tag{3.8}$$

Since $q_2 = 1 - q_0 - q_1 < 1 - p_S e^{\frac{p_S}{1-\hat{p}}}$, we have

$$E[p_S^{t+1}] \leq [1 - p_S e^{\frac{p_S}{1-\hat{p}}} + (1+\gamma)^{-1/2} p_S e^{\frac{p_S}{1-\hat{p}}}]p_S^t. \tag{3.9}$$

Suppose in the subframe I_{k-1} there is a round t with $p_S > 5$. One can find

a time interval $I' \subseteq I_{k-1}$, which satisfies $5 < p_S < 5e$ during I', $p_S < 5$ just before I', and $p_S > 5e$ at the end of I'. We intend to bound the probability at which such I' happens. Let $\phi = \log_{1+\gamma}[(1 - p_S e^{\frac{p_S}{1-\hat{p}}} + (1+\gamma)^{-1/2} p_S e^{\frac{p_S}{1-\hat{p}}})^{-1}]$. Since γ is sufficiently small, we have $\phi \in (0.5, 1)$ and $E[p_S^{t+1}] \leq (1+\gamma)^{-\phi}$. On the other hand, $p_S^{t+1} \leq (1+\gamma) p_S^t \leq (1+\gamma)^{2\phi} p_S^t$. Then let $X_S^t = \log_{(1+\gamma)} p_S^t + \sum_{i=0}^{t-1} \phi_k$ and $X_S^0 = \log_{(1+\gamma)} p_S^0$, it is easy to verify that $E[X_S^{t+1}] = X_t$ and $X_S^{t+1} \leq X_S^t + c_{t+1}$, where $c_{t+1} = 3\phi_t$. Leveraging the Azuma–Hoeffding Inequality, it holds that

$$P[X_S^T - X_S^0 > \delta] \leq e^{\frac{-\delta^2}{2\sum_{k=1}^{T} c_k^2}}, \tag{3.10}$$

for $\delta = 1/\gamma + \sum_{k=0}^{T-1} \phi_k$. Therefore

$$P[\log_{(1+\gamma)} p_S^T / p_S^0 > 1/\gamma] \leq e^{\frac{-\delta^2}{2\sum_{k=0}^{T-1} (3\phi_k)^2}}. \tag{3.11}$$

Let $\psi = \sum_{k=0}^{T-1} (\phi_k)^2$, we have $e^{\frac{-\delta^2}{2\sum_{k=0}^{T-1} (3\phi_k)^2}} = \frac{(\psi + 1/\gamma)^2}{18\psi} \geq \frac{1}{9\gamma}$. Hence,

$$P[\log_{(1+\gamma)} p_S^T / p_S^0 > 1/\gamma] \leq e^{-1/9\gamma} \leq \frac{1}{\log^c N}, \tag{3.12}$$

for any constant c if $\gamma = O(1/(\log T + \log\log N)$. Note that $\log_{(1+\gamma)} p_S^T / p_S^0 > 1/\gamma$ indicates $p_S^T / p_S^0 > e$. Considering $p_S^0 > 5$ at the beginning of a subframe I', $P[\log_{(1+\gamma)} p_S^T / p_S^0 > 1/\gamma]$ is the probability at which the aggregated probability of the nodes in S exceeds $5e$ at the end of I'. Hence we prove that if $p_S < 5$ holds at the beginning of I_{k-1}, $p_S < 5e$ holds throughout I_{k-1} w.m.p. Also, it is analogous to prove that if $p_S < 5e$ is true at the beginning of I_{k-1}, $p_S < 5e^2$ holds throughout I_{k-1} w.m.p. Hence, if $p_S \leq 5$ happens during the past subframe I_{k-1}, $p_S < 5e$ holds throughout I_k w.m.p. Since $p_S < 5e$ holds at the beginning of I_k, $p_S < 5e^2$ holds throughout I_k w.m.p., which proves the lemma. □

Lemma 5. *$(1-\varepsilon\beta')$-fraction of subframes in F satisfy $p_V \leq \rho$ w.h.p, where $p_V = \sum_{v\in V} p_v$ is the aggregated probability of all nodes, and $\varepsilon, \beta', \rho$ are constants.*

Proof. Let us focus on a fixed subframe I_k and its previous subframe I_{k-1}. Lemma 3 indicates that there is at least one round in I_{k-1} with $p_S \leq 5$ w.h.p. Then it follows from Lemma 4 that if there is at least one round in I_{k-1} with $p_S \leq 5$, $p_S < 5e^2$ holds throughout I_k w.m.p. Define a subframe I to be *good* if $p_S \leq 5e^2$ holds throughout I, and otherwise I is *bad*. Then it follows from the Chernoff bounds that at most $\varepsilon\beta'/6$ of the subframes in F are bad w.h.p. Since D_{R_0} consists of six sectors and covers all nodes in V, there is at least $(1-\varepsilon\beta')$-fraction of subframes in which the aggregated probability $p_V = \sum_{v\in V} p_v = \sum_{v\in N_{R_0}(v)} p_v$ is bounded by $\rho = 6\times 5e^2 = 30e^2$, which completes the proof. □

Then, the probability on which there exists one successful transmission is given by

$$\begin{aligned}
\sum_{v\in V} p_v \prod_{w\in V\setminus v}(1-p_w) &\geq \sum_{v\in V} p_v \prod_{w\in V}(1-p_w) \\
&\geq \sum_{v\in V} p_v \prod_{w\in V} e^{\frac{-p_w}{1-\hat{p}}} \\
&= \sum_{v\in V} p_v e^{\frac{-p_V}{1-\hat{p}}} \\
&= p_V e^{\frac{-p_V}{1-\hat{p}}} \\
&\geq \rho e^{\frac{-\rho}{1-\hat{p}}}.
\end{aligned} \tag{3.13}$$

□

With the robust communication guarantee, we next need to prove that BLOWN can support a successful leader election, which is the core of the protocol. Most leader election algorithms in wireless networks are only responsible for reaching the state at which one node is the leader and others are followers. Our algorithm goes one-step further by ensuring that all

nodes have an identical view of the network after leader election, which is crucial to the our protocol, as shown in Theorem 4.

Theorem 4. *(Successful leader election). Let w_{max} be the maximum weight among all nodes and λ be a constant to be determined. $\pi_B[\mathcal{F}_{SIG}, \mathcal{F}_{SORT}]$ ensures a successful leader election while satisfying the following three properties: 1)* ***Correctness****: only one node is left as the leader with a positive l_v at the end of P_1; 2)* ***Efficiency****: the success of leader election can be achieved within $O(w_{max}\lambda)$ rounds w.h.p.; 3)* ***Practicality****: the leader and the followers should have the knowledge regarding who is the leader and at which round the leader is elected.*

Proof. We prove the three properties in order. During a leader election process, all nodes contend for broadcasting messages in P_1 until only one node is left with a positive l_v, which can always be achieved inevitably. This can be proved by contradiction. Without loss of generality, we assume that there are two nodes left with a positive l_v. If these two nodes broadcast messages at the same round, they can not receive messages from each other simultaneously. Therefore, there is no chance for two nodes to receive messages in the same round, and there must be only one node surviving at the end. One can trivially expand this result to the cases with $3, 4, \cdots, N$ nodes left with positive l_v values, thus proving that the protocol can always lead to the state when only one node survives as the leader with a positive l_v.

To prove the efficiency property, we resort to Theorem 3, which shows that a constant fraction of the rounds have successful transmissions w.h.p. Concretely, a successful communication should happen once every $\lambda = (1-\varepsilon\beta)^{-1}\rho^{-1}e^{\frac{\rho}{1-\rho}}$ rounds on average w.h.p. Then leader election can be finished in $O(w_{max}\lambda)$ rounds w.h.p. This indicates that $O(1)$ number of

rounds can lead to a successful leader election and the communication complexity is not directly related to the network size.

To prove the practicality, we denote E_v as the event that v broadcasts a message in slot one and senses an idle channel in slot two. In this case, v would know itself as the leader. Let p_v be the probability that v broadcasts a message in slot one, $p_v^{(0)}$ be the probability that v broadcasts a message and there is also at least one node u with $l_u > 0$ broadcasting a message in slot one, $p_v^{(1)}$ be the probability that v broadcasts a message and there exists at least one node u with $l_u > 0$ sensing the channel in slot one, and $p_v^{(2)}$ be the probability that v broadcasts a message in slot one and $l_u = 0, \forall u \in V \backslash \{v\}$. Certainly, $p_v = p_v^{(0)} + p_v^{(1)} + p_v^{(2)}$. If E_v happens, v senses an idle channel in slot two. Then $p_v^{(0)} = 0$ since if u broadcasts a message in slot one, a follower f senses interference and thus broadcasts an m in slot two so that v senses interference in slot two, which contradicts our assumption. Also, $p_v^{(1)} = 0$ because if there exists a node u with $l_u > 0$ sensing the channel in slot one, u has to broadcast a message in slot two which also contradicts the assumption. Therefore, we obtain the result that if E_v happens, v can confirm itself as the unique leader.

Correspondingly, we denote E_f as the event that a follower f recognizes v as the leader when f receives a message from v and obtains $\mathscr{I} + \mathscr{N} < \theta$ in slot one, then senses an idle channel in slot two. Let p_f be the probability that f receives a message from v and obtains $\mathscr{I} + \mathscr{N} < \theta$ in slot one, $p_f^{(0)}$ be the probability that there is at least one node $u \in V \backslash \{v\}$ with $l_u > 0$ sensing the channel in slot one, $p_v^{(1)}$ be the probability that v is the unique leader; then we have $p_f = p_f^{(0)} + p_f^{(1)}$. Assume E_v happens, we have $p_f^{(0)} = 0$ since if $p_f^{(0)} \neq 0$, u has to broadcast a message in slot two and thus a follower senses interference, which contradicts our assumption. As a result, E_f indicates

that v is the unique leader. Additionally, the round at which a successful leader election happens can be found when E_v and E_f occur simultaneously, which ends the proof of the third property. □

Utilizing Theorem 7 and 4 as intermediate conclusions, we are finally ready to prove of Theorem 2.

Proof. The time between `LEADER` and `COMMIT` is fixed to $j = c \cdot i_k$ rounds, where c is an adjustable constant parameter according to different implementation scenarios. If the leader does not broadcast a block in the $(c+1)i_k$-th round, the state transits to the final state since $\mathbb{S}$ satisfies the second condition of a `FINAL` state. Then $\mathbb{S}$ starts the next epoch. According to Theorem 4, each epoch can be terminated in $O(cw_{max}\lambda)$ w.h.p., and there should be $O(cw_{max})$ transactions added to the blockchain in each epoch w.h.p.

Assume an honest node v generates the transaction tx_i^j and contends to broadcast it in P_2. The transaction can be received by an honest leader with probability at least $p = cw_{max}/N$ in each epoch. By applying the Chernoff bound, we obtain that tx_i^j can be added to the blockchain within n epochs with probability at least $1 - e^{-\frac{(np-1)^2}{2}}$, where n is the number of epochs when v broadcasts tx_i^j. The above analysis indicates that $\mathbb{S}$ has no chance of staying at a deadlock in any epoch. Considering the assumption that honest nodes control more than 50% coins, $\pi_{\mathrm{B}}[\mathscr{F}_{\mathrm{SIG}}, \mathscr{F}_{\mathrm{SORT}}]$ with ideal functionalities $[\mathscr{F}_{\mathrm{SIG}}, \mathscr{F}_{\mathrm{SORT}}]$ can ensure a fair sortition based on the nodes' stake distribution. Thus, with probability at least 50%, an honest node can be selected as a leader to propose a new block. When two chains $\mathscr{C}_1, \mathscr{C}_2$ possessed by two honest nodes at the onset of two epochs $e_1 < e_2$ with e_2 at least k epochs ahead of e_1, it holds that $len(\mathscr{C}_2) - len(\mathscr{C}_1) \geq \tau \cdot k$, where $\tau = 0.5$. This completes the proof of Theorem 2. □

3.4.2.2 Common Prefix

Theorem 5. *$\pi_B[\mathcal{F}_{SIG}, \mathcal{F}_{SORT}]$ satisfies the common prefix property (following Definition 13).*

Proof. In $\pi_B[\mathcal{F}_{SIG}, \mathcal{F}_{SORT}]$ a node can directly append a new block B_u^k to its local blockchain only when $\mathcal{F}_{SIG}$ and $\mathcal{F}_{SORT}$ answer with 1 when being queried. Therefore, an adversary who intends to disguise itself as a leader to propose a block should fail since it cannot break [$\mathcal{F}_{SIG}$ and $\mathcal{F}_{SORT}$]. However, a malicious leader (an adversary who wins the leader election) can still diverge the global distributed ledger to cause a Δ-fork defined in Definition 16.

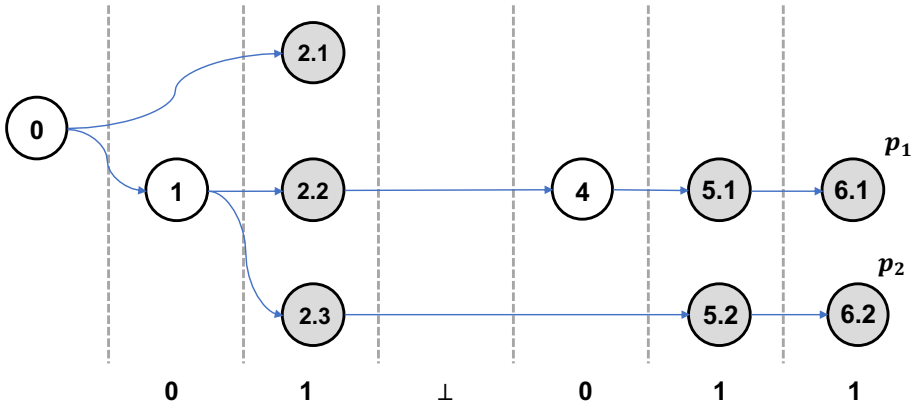


Figure 3.6: A simple example to illustrate the concepts of string, Δ-fork, and divergence. Specifically, we have $s_6 = \{0,1,\perp,0,1,1\}$, $l(p_1) = 5$, and $l(p_2) = 4$. The tree presented here is a Δ-fork with $\Delta = 1$, and $|div(p_1,p_2)| = 4$.

Definition 15 (String)**.** *Consider an epoch e_k during the execution of functionality $\pi_B[\mathcal{F}_{SIG}, \mathcal{F}_{SORT}]$ with adversary $\mathcal{A}$ and environment $\mathcal{Z}$. Let $S = \{e_1, \cdots, e_k\}$ denote a sequence of epochs of length k. The string $s = \{0,1,\perp\}^k$ of S is defined so that $s_i = 1$ if the adversary controls the epoch leader of e_i and broadcasts a block, $s_i = 0$ if an honest node controls the epoch and broadcasts a block, and $s_i = \perp$ if no block is broadcast. We say that the index i is adversarial if $s_i = 1$ and honest otherwise.*

W.l.o.g., let $s_0 = 0$ for e_0 meaning that the genesis block has an honest index.

Definition 16 (Δ-Fork)**.** *Let string $s = \{0,1,\perp\}^k$ of S and Δ be a non-negative interger. A Δ-fork is a directed, acyclic, rooted tree $F = (V,E)$ in which the two longest paths p_1 and p_2 satisfy $|l(p_1) - l(p_2)| \leq \Delta$, where a path p refers to a road from the root to a leaf and $l(p)$ is the hop-count (length) of the path p.*

Definition 17 (Divergence)**.** *Denote the divergence of two paths p_1 and p_2 in a Δ-Fork as $div(p_1,p_2)$, which is defined as*

$$div(p_1,p_2) = \max\{l(p_1),l(p_2)\} - l(p_1 \cap p_2), \tag{3.14}$$

where $l(p_1 \cap p_2)$ is the legnth of the common path of p_1 and p_2, and $div(p_1,p_2)$ is non-negative.

Lemma 6. *The common prefix property is satisfied if and only if for any pair of paths $p_i, p_j, i \neq j$, in a Δ-fork, $div(p_i,p_j) \leq k$.*

Proof. For the "only if" direction, we assume that there exits a path p_1, p_2 (w.l.o.g., $l(p_1) > l(p_2)$) such that $div(p_1,p_2) > k$. That is $\max\{l(p_1),l(p_2)\} - l(p_1 \cap p_2) = l(p_1) - l(p_1 \cap p_2) > k$. Let V_1 (V_2) be the set of honest nodes that store the distributed ledger as the path p_1 (p_2). Once queried a local blockchain, any $v_1 \in V_1$ ($v_2 \in V_2$) responds with $\mathscr{C}_1$ ($\mathscr{C}_2$). Denote the latest point of the common path $p_1 \cap p_2$ as $\hat{v}$, which also called a bifurcation point. The path $p_1^{\lceil k}$ that is obtained by truncating the last k vertices of p_1 still covers $\hat{v}$, which is not the endpoint of p_1 since $l(p_1) - k > l(p_1 \cap p_2)$. Denote the endpoint of p_1 as $end(p_1)$. Then the blocks corresponding to the points from $\hat{v}$ to $end(p_1)$ is included in $\mathscr{C}_1^{\lceil k}$, but the block mapped to $end(p_1^{\lceil k})$ is not included in $\mathscr{C}_2$, thus violating the common prefix property. For the "if" direction,

assuming that the common prefix is violated, there exists a pair of ledger s$\mathscr{C}_1, \mathscr{C}_2$ for $e_1 < e_2$ such that $\mathscr{C}_1^{\lceil k} \not\preceq \mathscr{C}_2$. Mapping such blockchains to two distinct paths p_1, p_2, the endpoint $end(p_1^{\lceil k})$ corresponding to the latest block in $\mathscr{C}_1^{\lceil k}$ is not covered by p_2 and comes after $\hat{v}$. By the definition of divergence, $div(p_1, p_2) = \max\{l(p_1), l(p_2)\} - l(p_1 \cap p_2) > k$. □

Here one can define a common prefix violation as the case when there exit two paths p_1, p_2 in a Δ-fork with $|div(p_1, p_2)| > k$. To prove Theorem 5, we need to show that a common prefix violation happens with an extremely small probability. Generally speaking, $\Delta \leq k$, and p_1, p_2 can be regarded as the respective paths that the honest nodes and adversary go through. This is based on the assumption that all honest nodes strictly follow the longest chain rule, while the adversary focuses on increasing the length of an illegal chain (e.g., including a double-spend transaction). Therefore, a common prefix violation can also be interpreted as a race between honest nodes and the adversary that lasts for more than k blocks, but their view paths still follow $|l(p_1) - l(p_2)| \leq \Delta$. Let $X_i \in \{\pm 1\}$ (for $i = 1, 2, \cdots$) denote a series of independent random variables for which $Pr[X_i = 1] = (1 - \varepsilon)/2$. Note that $\varepsilon \in (0, 1)$ is satisfied in functionality $\pi_{\mathrm{B}}[\mathscr{F}_{\mathrm{SIG}}, \mathscr{F}_{\mathrm{SORsT}}]$ since the adversary controls less than 50% coins and the protocol adopts a hybrid $[\mathscr{F}_{\mathrm{SIG}}, \mathscr{F}_{\mathrm{SORT}}]$ to ensure that the probability of the adversary being a leader is less than $1/2$. Consider k epochs of the biased walk beginning at the bifurcation point. The resulting value is tightly concentrated at $-\varepsilon k$. By applying the Chernoff bound, for each k random walk hitting problem, we have

$$Pr[X_k < \Delta] \leq e^{-(1-\Delta/\varepsilon k)^2 \varepsilon k/2} = e^{-O(k)}, \tag{3.15}$$

where $\Delta \ll k$. This indicates that $\pi_{\mathrm{B}}[\mathscr{F}_{\mathrm{SIG}}, \mathscr{F}_{\mathrm{SORT}}]$ satisfies the common prefix

property w.h.p., which completes the proof. □

3.4.2.3 Chain Quality

The chain quality property requires that a certain fraction of the blocks should satisfy high quality standards (high-quality blocks are the ones generated absolutely by honest nodes). Chain quality can be threatened by Sybil attacks which are particularly harmful in wireless networks [81]. In a Sybil attack, an attacker can behave as many nodes by illegitimately claiming massive identities or impersonating others. A successful attacker chosen as a leader can deny to broadcast a new block or broadcast an invalid block. Since honest nodes can neither wait for more than $c \cdot i_k$ rounds in P_2 nor accept invalid blocks, the attacker cannot hinder the system from changing from `LEADER` state to the `FINAL` state. However, an attacker can make an epoch wasted without any new block being added to the blockchain, thereby harming the liveness. Our BLOWN protocol prevents Sybil attacks and ensures liveness under the assumption that all malicious nodes control no more than 50% stake of the entire network.

Consider one epoch. $\mathscr{F}_{\mathrm{SORT}}$ provides a binomial distribution as $B(k; w_v, p) = \binom{w_v}{k} p^k (1-p)^{w_v - k}$, which has a salient property that splitting coins into multiple sub-users does not give attackers any advantage. In particular, suppose an attacker splits its account balance $w_{\mathscr{A}}$ into $w^1_{\mathscr{A}}, w^2_{\mathscr{A}}, \cdots, w^n_{\mathscr{A}}$, thus each sub-user has a binomial distribution as $X^i_{\mathscr{A}} \sim B(w^i_{\mathscr{A}}, p)$. However, splitting coins does not increase the sum of the values of the leader counter controlled by the attacker since $(X^1_{\mathscr{A}} + X^2_{\mathscr{A}} + \cdots + X^n_{\mathscr{A}}) \sim B(w^1_{\mathscr{A}} + w^2_{\mathscr{A}} + \cdots + w^n_{\mathscr{A}}, p)$. Also, splitting coins decreases the maximum of the leader counter of the sub-users, which makes it harder for a sub-user to survive in P_1. Without loss of generality, suppose each node has an equal value of balance. Then

at each epoch, the probability of a malicious node being chosen as a leader is no more than 50%.

Theorem 6. *Given that the ratio of the adversarial stake $\alpha < 1/2$, $\pi_B[\mathscr{F}_{SIG}, \mathscr{F}_{SORT}]$ satisfies the chain quality property with $\mu = 1-(1+\delta)\alpha$, where $\delta \in (0,1)$.*

Proof. Let X_i denote the event where the ith epoch has an adversarial leader. We have $\mathbb{E}[X_i] \leq \alpha l$. Applying the Chernoff bound we obtain

$$P_r[X \geq (1+\delta)\alpha l] \leq e^{-O(l)}. \tag{3.16}$$

Then the probability that the ratio β for the blocks originated from the adversary is at most $(1+\delta)\alpha$ is given as

$$P_r[\beta \leq (1+\delta)\alpha] = 1 - P_r[X \geq (1+\delta)\alpha l] \geq 1 - e^{-O(l)}. \tag{3.17}$$

When l is sufficiently large, $\beta \leq (1+\delta)\alpha$ w.h.p. Thus we complete the proof of the chain quality property with $\mu = 1-(1+\delta)\alpha$. Note that even though $\mu = 1-(1+\delta)\alpha$ blocks can be proposed by the adversary, these blocks only contain a small fraction of malicious ones (jointly ensured by the chain growth and common prefix properties). □

Thus we can conclude that $\pi_B[\mathscr{F}_{SIG}, \mathscr{F}_{SORT}]$ satisfies the chain growth, common prefix, and chain quality properties, thus guaranteeing persistence and liveness. By applying Theorem 1, BLOWN (i.e., $\pi_B[\pi_{SIG}, \pi_{SORT}]$) naturally ensures persistence and liveness.

3.5 Simulation Study

In this section, we implement a simulator to study how various parameters impact the performance of our BLOWN protocol. Specifically, in Section 3.5.1, we analyze the one-epoch execution of BLOWN to show its correctness and efficiency. Then we present the performance of BLOWN under various jamming and Sybil attacks in Section 3.5.2.

In our simulation, we use crypto library of golang [4] and adopt ed25519 for digital signature, with 64-byte private key, 32-byte public key, and 64-byte signature. Public keys are broadcasted to all nodes on the preset of our simulations. Besides, The Sortition algorithm is implemented with the VRF provided by CONIKS [5]. We adopt two types of 2-dimensional planes of size $d \times d$ units, where $d = 10$ or $d = \sqrt{N}$ with N known as the network size. Nodes are randomly generated and distributed in the plane and no two nodes can have the same coordinates. The unit for an epoch length is round. If not stated otherwise, we adopt the parameters $\alpha = 4$, $\beta = 2$, $\theta = 2$, $P = \beta\theta(\sqrt{2}d)^{\alpha}$, $\hat{p} = 0.1$, $T = 60$, $\varepsilon = 0.3$, $\gamma = 0.1$, $w_v = 20$, $\tau = W/2$, and $c = 10$. Besides, nodes are uniformly distributed and the percentage of Sybil nodes is 0% by default. Without loss of generality, all parameter values are chosen carefully to reflect various real-world cases, but not to aim to optimize the performance. All the experiments are performed under a CentOS 7 operating system running on a machine with an Intel Xeon 3.4 GHz CPU, 120 GB RAM and 1 TB SATA Hard Drive.

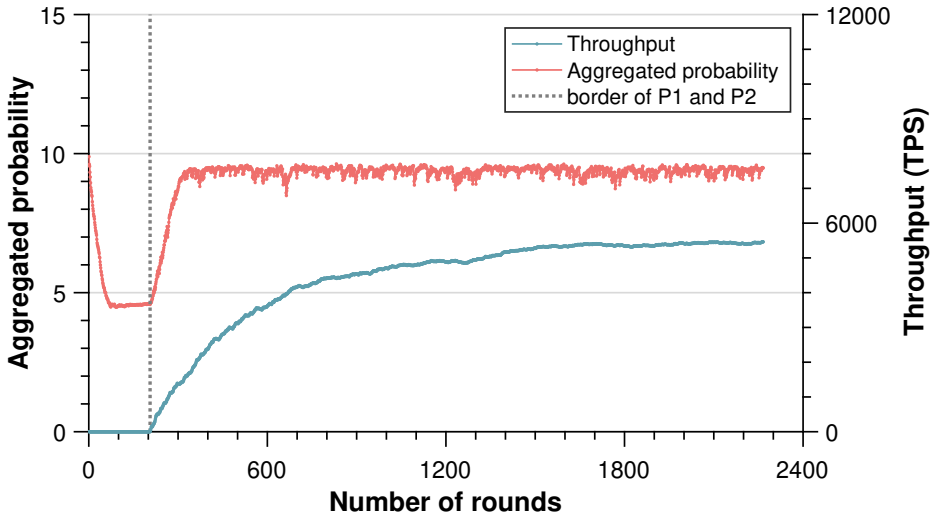


Figure 3.7: Aggregated probability and throughput *vs.* number of rounds, to demonstrate the convergence behavior, where $density = 1, d = 10, N = 100$, and nodes are uniformly distributed.

3.5.1 Correctness and Efficiency

One-Epoch Execution. We denote by $|txp_t|$ the number of transactions received by the leader within $t = i + j$ rounds, with i and j respectively being the number of rounds in P_1 and P_2. Given that the unit slot time for IEEE 802.11 is set to be $50\mu s$, we have throughput as

$$\text{Throughput} = \frac{|txp_t|}{i \times 100\mu s + j \times 50\mu s} \tag{3.18}$$

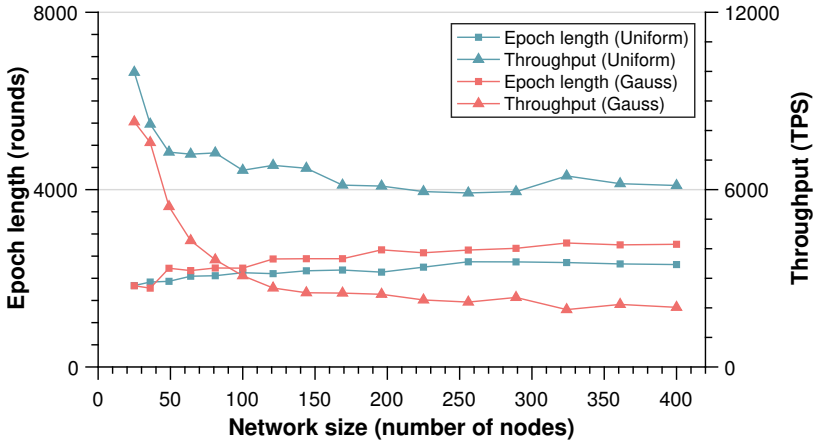
since $r^k_{1,i}$ has two slots while $r^k_{2,j}$ has only one slot. Figure 3.7 presents a typical example to illustrate the convergence of the aggregated probability $p_V = \sum_{v \in V} p_v$ and throughput during one-epoch execution, where $p_V = N \times \hat{p} = 10$ in the outset. There is a gray dash borderline distinguishing P_1 and P_2. Since BLOWN can rapidly adjust the initial parameters by multiplicatively increasing or deceasing p_v, p_V adapts rapidly to reduce the noise in the

[4]https://github.com/golang/crypto
[5]https://github.com/coniks-sys/coniks-go/tree/master/crypto/vrf

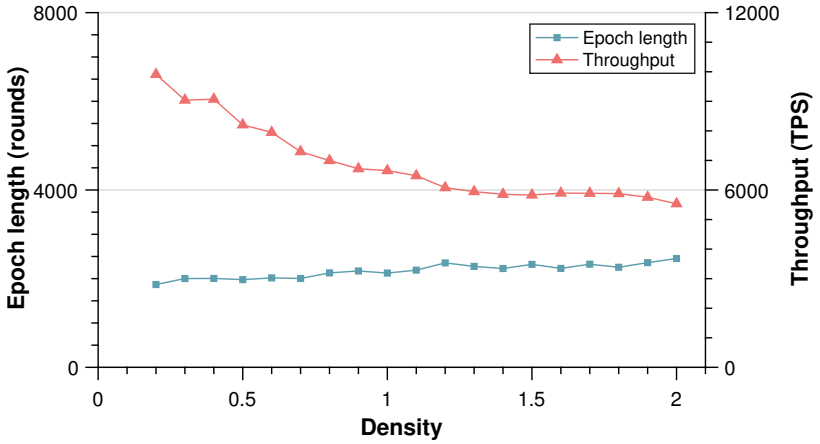
channel to help achieve successful communications. Therefore, it only takes 206 rounds (corresponding to 0.206s in a real-world setting) to complete P_1. Such a quick adaptation contributes to the throughput of the entire protocol. In P_2, nodes all become active to send transactions enabling p_V to grow. The leader collects transactions in the (207-2265)th rounds, and a block is finalized at the 2266th round. p_V and throughput respectively converge to 5399 TPS and 9.37, which are mean values calculated from the last 500 rounds. Besides, we evluate cryptographic overhead (ms, an average of 1000 times running), including the overhead of signing a transaction (0.09 ms), verifying a transaction [0.21 ms], signing a block (1.20 ms), confirming a block (930.14 ms), Sortition(3.02 ms), VerifySortition(4.57 ms).

Network Size. Next we simulate the performance as a function of the network size (or N), where nodes are scattered in the plane of size $d = \sqrt{N} \times \sqrt{N}$ following a uniform or Gauss distribution. As shown in Figure 3.8a, the epoch length slowly increases with a larger N with both uniform and Gauss distributions, which also means that the leader election costs more time for a larger N. On the other hand, throughput decreases with a larger N since the added nodes lead to heavier contention. However, because of the resiliency of our jamming resistant channel, throughput can converge to about 6000 TPS and 2000 TPS for the uniform and Gauss distribution, respectively. Compared with the uniform distribution, Gauss distribution always has a larger epoch length and lower throughput since denser nodes centrally aggregate, leading to stronger contention.

Network Density. We also investigate how the network density impacts on the performance of the BLOWN protocol. Nodes are uniformly distributed in a 10×10 plane, and $density = 0.2, 0, 3, \cdots, 2$. As shown in Figure 3.8b, the epoch length slowly increases from 1867 to 2464 rounds, with the



(a) Epoch length and throughput *vs.* the network size N, where $density = 1, d = \sqrt{N} \times \sqrt{N}$.

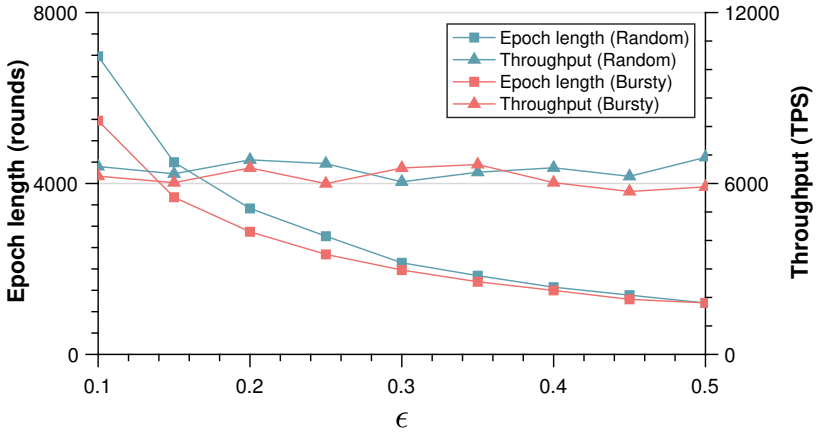


(b) Epoch length and throughput *vs.* the density, where $d = 10, N = 100$.

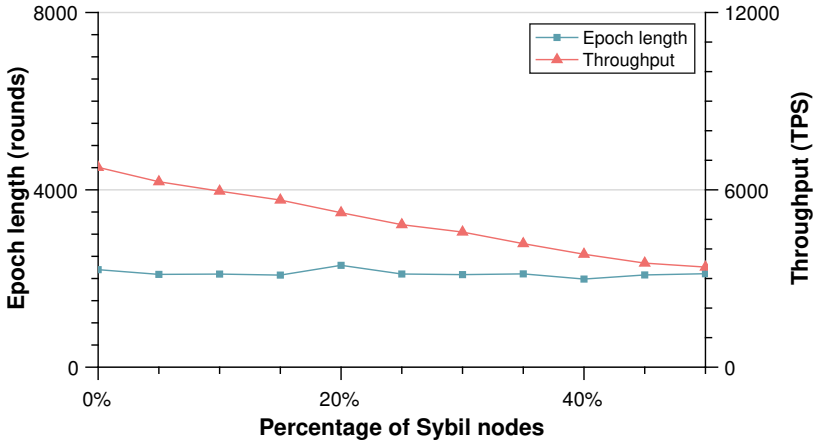
Figure 3.8: The performance of BLOWN *vs.* network size (in a uniform or Gauss distribution) and density.

density rising tenfold. The throughput decreases for larger density and approximately converges to 6000 TPS.

3.5.2 Jamming Attacks and Sybil Attacks



(a) Epoch length and throughput *vs.* ε, where $density = 1, d = 10, N = 100$.



(b) Epoch length and throughput with the percentage of Sybil nodes, where $density = 1$, $d = 10$, and $N = 100$.

Figure 3.9: The performance of BLOWN when confronting jamming attacks (random jammers or bursty jammers) and Sybil attacks.

Jamming Attacks. Here we present our protocol's performance when confronting jammers who can choose different strategies with the constraint of $(1-\varepsilon)T$. We consider two types of jammers: *random* jammers that

can randomly jam $(1-\varepsilon)T$ rounds at any interval of length T and *bursty* jammers who would jam continuous $(1-\varepsilon)T$ rounds at any interval of length T. We test the epoch length and throughput when $\varepsilon = 0.1, 0.15, \cdots, 0.5$, with a higher ε implying a lower attack frequency. The results are demonstrated in Figure 3.9a, which indicate that the epoch length decreases with the increasing ε due to the lower frequency of jamming attacks. Besides, ε does not significantly impact the throughput for both kinds of jammers. The epoch length increases faster with lower ε considering random jammers, indicating that random jammers are more powerful than bursty ones in BLOWN. This is because the introduction of T_v makes it easier to address continuous heavy contentions.

Sybil Attacks. In a Sybil attack, an attacker can control massive malicious nodes that compete for being a leader but refuse to collect transactions and propose blocks. In this circumstance, the epoch with a malicious leader would be abandoned so that there is no valid block to be accepted within such an epoch. Even though we already show in protocol analysis that BLOWN can defend against Sybil attackers who control less than 50% wealth of the entire network, such attackers can harm the liveness of our protocol. In Figure 3.9b, the percentage of Sybil nodes does not impact the epoch length since Sybil nodes are not absent from competing in the leader election. However, the throughput has an evident linear decline for a larger percentage of Sybil nodes. Compared to the setting without Sybil nodes, $50\%N$ Sybil nodes would decrease the throughput by 49.90%.

Chapter 4: wChain: A Fast Fault-Tolerant Blockchain Protocol for Multihop Wireless Networks

4.1 Introduction

In recent years, the popularity of 5G and IoT has arisen more security and privacy issues relevant to identity management, data sharing, and distributed computing in wireless networks. With the inception of Bitcoin, blockchain has been envisioned as a promising technology that can be utilized to support various applications such as online payments and supply chain due to its salient properties of decentralization, immutability, and traceability. Correspondingly, effort has been put on protecting wireless applications using the blockchain technology, e.g., mobile edge computing (MEC) [38], intelligent 5G [28], vehicular networking [70], and wireless sensor networking (WSN) [54].The common idea of applying blockchain in wireless networks is to introduce trustlessness with blockchain so that functions such as identity management and data sharing become more efficient and secure.

However, previous studies on blockchain-enabled wireless applications mostly focus on developing practical applications or proposing architectures based on existing blockchain protocols, which were originally designed for wired network applications and thus are not suitable for wireless scenarios. To defend this point of view, let's consider the state-of-the-art blockchain protocols. The concept of proof of physical resources has been widely adopted, e.g., Proof-of-Work (PoW), Proof-of-Space, Proof-of-Elapsed Time, Proof-of-Space Time. The noteworthy drawback of these protocols is that they require high electricity, storage, or specific hardware (e.g., Intel SGX),

which wireless devices cannot provide. On the other hand, protocols based on virtual resources such as stake, reputation, or credibility, i.e., Proof-of-Stake, Delegated Proof-of-Stake, Proof-of-Authority, Proof-of-Reputation, etc., always have a complicated design in order to avoid the centralization of wealth or power, which justifies why there is still no such a protocol particularly designed for wireless networks.

Another line of blockchain protocols, such as Ripple, Algorand, Tendermint, and Hotstuff, rely on message passing. They provide blockchain systems with safety and liveness in confronting faulty nodes or even Byzantine failures. However, when implemented in wireless networks, the following two major problems need to be addressed:

- Traditional protocols commonly used on the Internet are not efficient enough in wireless networks. For example, PBFT [23] and Tendermint [59] can reach a consensus within $O(N^2)$ successful transmissions where N is the network size, while Hotstuff [117] reduces complexity to $O(N)$ but increases additional overhead due to the introduction of the cryptographic tools.

- Transforming fault-tolerant or Byzantine fault-tolerant protocols used in wired networks to wireless environments remains a tricky problem. A recent work by Poirot *et al.* presented a fault-tolerant wireless Paxos for low-power wireless networks [88]. However, their results still indicate that traditional consensus algorithms require many message exchanges and high bandwidths, which may not be available in wireless networks.

In this chapter, we consider designing a message-passing-based blockchain protocol for wireless networks. To achieve this goal, we primarily need to

overcome the challenge of properly implementing the medium access control (MAC) layer for blockchain. A promissing idea is to adopt the abstract MAC layer service (absMAC) proposed by Kuhn *et al.* [57], which provides low-level network functions to help design distributed algorithms in wireless networks. The most efficient implementation of absMAC was presented in [120], which achieves the optimal bound for a successful transmission in $O(\Delta + \log N)$ slots via carrier sensing, where Δ is the maximum number of neighbors a node may have. Obviously, with this efficient implementation and a fault-tolerant consensus algorithm that requires an optimal $O(N)$ successful transmissions (e.g., Hotstuff), the best we can obtain is a protocol with a communication complexity as high as $O(N(\Delta + \log N))$ slots.

Nevertheless, this is far less than satisfactory. One needs a blockchain protocol that can take into consideration the unique features of wireless networking by deeply integrating wireless communications with blockchain to achieve the necessary properties of efficiency, fault-tolerance, persistence, and liveness. Since the most basic primitives heavily used in wireless consensus algorithms are broadcasts and data aggregations, which are respectively responsible for disseminating and collecting opinions or votes from peers, we use a spanner structure to accelerate the consensus process. A spanner is a hierarchical communication backbone that can organize nodes carefully to speed up data aggregation and dissemination processes. It introduces a sparse topology in which only a small number of links need to be maintained such that efficiency and simplicity can be well-balanced. Due to the need for decentralization and practicality, our spanner is constructed in a distributed manner and works under the realistic Interference-plus-Noise-Ratio (SINR) model. Facilitated with the spanner structure, we develop the two primitives of data aggregation and reaggregation to handle interference

with an adaptive power scheme directly. These two primitives are adopted by our fault-tolerant *wChain* protocol to address faulty behaviors caused by fail-stop errors and dynamic topologies such that the properties of resource conservation, fault-tolerance, efficiency, persistence, and liveness can be achieved.

The main contributions of this chapter are summarized as follows.

1. To the best of our knowledge, *wChain* is the first blockchain protocol that is particularly designed for multihop wireless networks under a realistic SINR model to deeply integrate wireless communications and blockchain technologies.

2. The *wChain* protocol ensures high performance by employing a spanner as the communication backbone. The runtime upper bound of the protocol is $O(\log N \log \Gamma)$ when crash failures happen in a low frequency, and the worst-case upper bound is $O(f \log N \log \Gamma)$ where $f = \lfloor \frac{N}{2} \rfloor$ is the maximum number of faulty nodes that *wChain* can tolerate.

3. Our *wChain* protocol simultaneously achieves properties of resource conservation, fault-tolerance, efficiency, persistence, and liveness, which are proved by theoretical analysis and verified by simulation studies.

4.2 Models and Preliminaries

Blockchain Basics. Each node v maintains a blockchain locally, denoted by BC_v, which is a hash-chain of blocks. B_v^i refers to the ith block in BC_v. We also denote BC_v^{i+} (BC_v^{i-}) as the partial blockchain of BC_v before (after) B_v^i. Each block contains multiple transactions, and tx_i^j stands for the jth transaction in B_v^i. Assume the latest block of BC_v is B_v^k; then v' *view* is defined as a tuple $\{seq, hash\}_v$, where *seq* and *hash* are the sequence number and

block hash of B_v^k, respectively. Besides, we adopt the UTXO model due to its remarkable properties such as large scalability and high level of security. We further assume nodes are supported by public key infrastructure, and the cryptographic primitives such as signature leveraged in our design are secure so that no malicious entity can spoof the messages.

Network Model. We consider a multihop wireless ad hoc network with a set V of N nodes deployed in a 2-dimensional geographic plane. Let $d(u,v)$ denote the Euclidean distance between nodes u and v, $D_R(v)$ denote the disk centered at v with a radius R, and $N_R(v)$ denote the set of nodes excluding v within $D_R(v)$. For simplicity, we normalize the minimum distance between any two nodes to be 1, and denote by Γ the ratio of the maximum distance to the minimum distance between any two nodes. Each node has a unique id and knows no advanced information other than the network size N. The transmission power of each node can be controlled for interference mitigation. Assume nodes can crash at any time, which means that each node is either functioning normally or completely stop working. A node is regarded as *faulty* if it crashes in the current epoch or does not have the latest view due to crash failures in previous epochs. Without loss of generality, we assume N is odd. Our protocol can tolerate at most f faulty nodes where $f = \lfloor \frac{N}{2} \rfloor$.

Interference and SINR Model. We adopt the Signal-to-Interference-plus-Noise-Ratio (SINR) model, which captures the wireless network interference in a more realistic and precise way than a graph-based one. A standard SINR model can be formulated as follows, which states that a message sent by u is correctly received by v if and only if

$$SINR(u,v) = \frac{P_u \cdot d(u,v)^{-\alpha}}{\mathcal{N} + \sum_{w \in S \setminus \{u\}} P_w \cdot d(w,v)^{-\alpha}} \geq \beta \tag{4.1}$$

holds, where $\alpha \in (2,6]$ is the path-loss exponent, threshold $\beta > 1$ is determined by hardware, $P_u \cdot d(u,v)^{-\alpha}$ is the received signal power of node v from u, $\mathcal{N}$ is the ambient noise, $P_w \cdot d(w,v)^{-\alpha}$ is the received signal power of node v from w, $S \subseteq V$ denotes the set of nodes transmitting simultaneously with u. Besides, we assume that nodes can perform physical carrier sensing.

Maximal Independent Set (MIS). A set $S \subseteq V$ is an *independent set* of V with respect to distance r if for any pair of nodes u and v in S, $d(u,v) > r$; and S is referred to as a *maximal independent set* if for any node $w \notin S$, there is a node $x \in S$ such that $d(w,x) \leq r$. MIS has been widely researched in recent years, and there exist a number of methods computing an MIS in a distributed manner. In this chapter, we adopt the approach presented in [118], which computes a distributed MIS in optimal time $O(\log N)$ if the nodes' density is a constant.

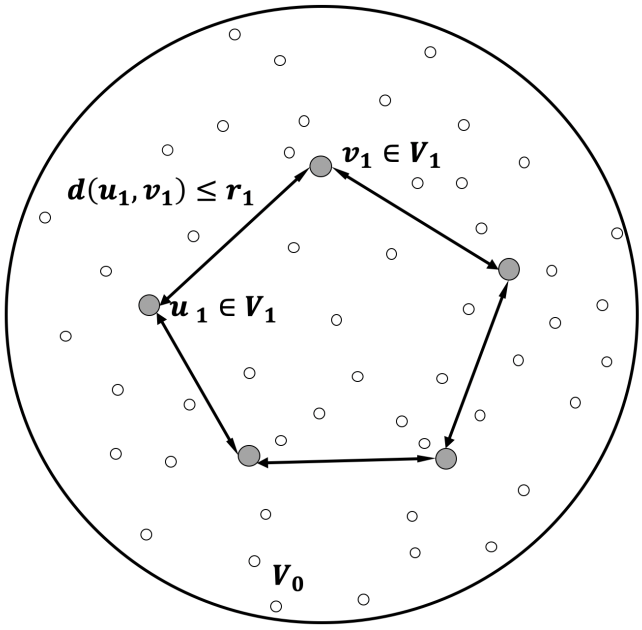


Figure 4.1: Construction of the first two layers of a spanner followed by an MIS; here hollow and solid circles represent V_0 and V_1, respectively.

Spanner Construction. A spanner is a network backbone possessing

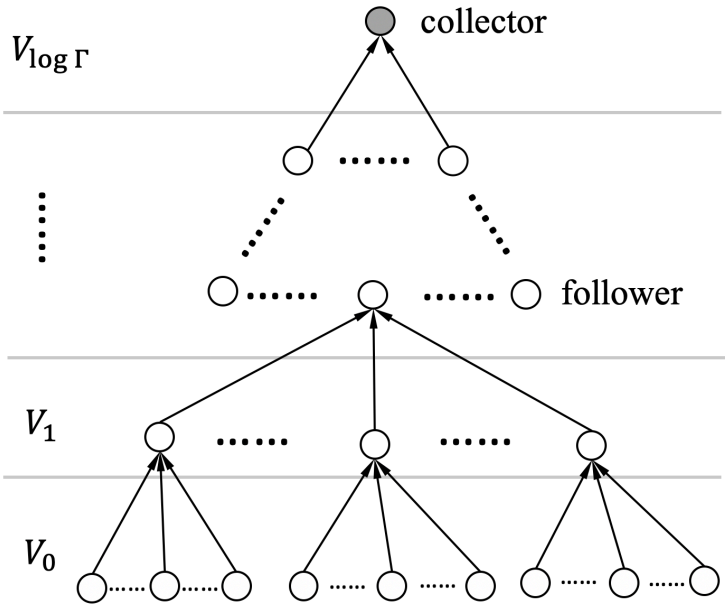


Figure 4.2: Spanner visualization: A $(\log\Gamma+1)$-level full spanner where arrows stand for the MIS relationship: each MIS node at a certain level covers a set of nodes in the next lower level.

the following properties: it only needs to maintain a small number of links, and can balance well between efficiency and simplicity compared to other topologies. Taking advantages of these features we employ a spanner to facilitate the deployment of our data aggregation algorithm in *wChain*. More specifically, we adopt the distributed spanner construction algorithm presented in [118], which can construct a sparse spanner, denoted as H, with a bounded maximum degree, in $O(\log N\log\Gamma)$ slots with a high probability. As illustrated in Figure 4.2, the construction process of H contains $\log\Gamma$ rounds; and in the ith round, where $i=1,2,\cdots,\log\Gamma$, V_i contains the nodes in a maximal independent set elected from V_{i-1} by running a distributed MIS algorithm with respect to $r_i=2^i$; thereby $V_{\log\Gamma}\subseteq\cdots\subseteq V_1\subseteq V_0=V$. One can see that the constructed H holds the following properties:

- nodes in V_i constitute an MIS of V_{i-1} with respect to r_i;

- each node $v \in V_{i-1} \setminus V_i$ has a parent node $u \in V_i$ and $d(v,u) \leq r_i$;
- $V_{\log\Gamma}$ contains only one node, i.e., the root.

For ease of explanation, nodes in $V_0 \setminus V_{\log\Gamma}$ and $V_{\log\Gamma}$ are referred to as *followers* and *collector*, respectively. In this chapter, we say that an event E occurs with high probability (w.h.p.) if for any $c \geq 1$, E occurs with probability at least $1 - 1/N^c$. A summary of all critical notations and their semantic meanings is presented in Table 4.1.

Table 4.1: Summary of Notations (*wChain*)

Symbol	**Description**
BC_v	the blockchain locally stored at node v
B_v^i	the ith block in BC_v
BC_v^{i+}	the partial blockchain of BC_v before B_v^i
BC_v^{i-}	the partial blockchain of BC_v after B_v^i
C_v	the set of child nodes of v
d_v	v's data (to be aggregated)
id_v	v's unique id
id_v^p	the unique id of v's parent
M_v	the message queue maintained by v
m_v	a single message sent by v
p	the uniform transmission probability
P_i	the transmission power in the ith round
$\hat{P}$	the maximum transmission power
R_i	the ith round
tx_i^j	the jth transaction in the ith block
V_i	the set of nodes as an MIS of V_{i-1}
σ	a sufficiently large constant to determine p
λ'	an upper bound of the network density
μ	a constant to determine the round length

4.3 The Protocol

In this section, we first present the design objectives, an overview on *wChain* and the involved utilities; then we detail the data aggregation and

reaggregation algorithms, the two subroutines of our blockchain protocol; finally, we propose the three-phase fast fault-tolerant *wChain* protocol.

4.3.1 Design Objectives

The primary design objectives of *wChain* include full decentralization, high efficiency, fault-tolerance, persistence and liveness.

1. **Full decentralization:** *wChain* should be a distributed protocol that does not rely on a centralized authority. This also indicates that the protocol should be free from any single point of failure.

2. **High Efficiency:** *wChain* has low communication complexity, providing low latency and high throughput.

3. **Fault-Tolerance:** *wChain* should be robust enough to counter against $f = \lfloor \frac{N}{2} \rfloor$ faulty nodes according to the network model. Particularly, we should address the tricky and challenging problem when nodes become faulty within an epoch.

4. **Persistence and Liveness:** As a blockchain protocol, *wChain* should satisfy persistence and liveness, which are defined in the protocol analysis section.

4.3.2 Technical Overview and Utilities

4.3.2.1 Technical Overview

The *wChain* protocol is executed in disjoint and consecutive time intervals called *epochs*, and at each epoch, no more than one block can be generated. Non-faulty nodes should append the new block to their local blockchain so that they can jointly maintain a consistent global view. Within each

epoch, a spanner is first established as a communication backbone, and the collector of the spanner is appointed as the leader to take charge of *wChain* for the entire epoch. However, if a crash failure occurs, a new spanner should be constructed in a reaggregation procedure for the same epoch. Under this circumstance, the collector might be changed, but the leader holds the line. Data can be aggregated from followers to the new collector who subsequently sends the aggregated data to the leader to complete the data aggregation process of the current epoch. *wChain* proceeds by three phases, namely `PREPARE`, `COMMIT`, and `DECIDE`. In the `PREPARE` phase, an incumbent leader aggregates view messages from all followers to learn about the latest view. If more than f followers respond with the same view as that of the leader, the leader can aggregate transactions in the `COMMIT` phase. Otherwise, the leader sends a message to the followers to abandon the current epoch. The threshold $(f+1)$ (including f followers plus one leader) is specifically designated to satisfy the *quorum intersection property* which implies that among the $(f+1)$ nodes, at least one node is non-faulty and has the correct information. In the `DECIDE` phase, the leader verifies the collected transactions, organizes them into a new block, and then sends the new block together with the view update information to the entire network, by which the nodes can update their local blockchain to obtain the latest view.

4.3.2.2 Utilities

Before delving into details of the *wChain* protocol, we explain its commonly used utilities as shown in Algorithm 0. First of all, $MSG(data_v)$ is used the most often in *wChain* to generate a single message m embodying a variable data field, which might be a string, a transaction, or a block. For example,

Algorithm 6 Utilities for node v

function MSG($data_v$)
    $m.data \leftarrow data_v$
    $m.timestamp \leftarrow time$
    $m.kindred \leftarrow \{id_v, id_v^p\}$
    $m.role \leftarrow \{role_v, level_v\}$
    return m
function ADD(M_v, M_u)
    return $M_v \leftarrow M_v \cup M_u$
function PACK(M_v)
    for $m \in M_v$ **do**
        if $m.data$ is a valid transaction **then** organize $m.data$ into B_v
    return B_v
function APPEND(BC_v, B_u)
    if B_u contains the hash of the B_v^k **then**
        return $BC_v \leftarrow BC_v + B_u$
function EXTRACT(BC_v, M_v^{view})
    $i \leftarrow$ the $(f+s)$th highest seq searched in M_v^{view}
    return BC_v^{i+}
function UPDATE(BC_v, BC_u^{i+})
    for $j = 1$ to $|BC_u^{i+}|$ **do**
        if $B_u^j \notin BC_v$ **then** $append(BC_v, B_u^j)$
    return BC_v

the parameter $data_v$ can be $view_v$ (view information), or tx_v (transaction); it also indicates that the input data is signed by v. Other than the data field, m also includes the `timestamp`, `kindred`, and `role` fields for the verification purpose. The `kindred` is a tuple consisting of v's identity id_v and the identity id_v^p of v's parent. The `role` field clarifies the role (i.e., follower, collector, or leader) and the specific level information such as $V_0 \setminus V_1$. A receiver recognizes m as valid if m includes correct identity and role information. Moreover, M_v is a message queue to hold multiple messages. Duplicate messages in M_v, which can be identified based on the timestamp information, should be discarded. A node v can perform an $add(M_v, M_u)$ operation to append M_u to its local M_v. For block formation, we provide $pack(M_v)$ to read all messages from M_v and organize valid transactions into a block. Besides, $append(BC_v, B_u)$ can append a block B_u from node u to BC_v only if B_u contains the hash of the last block of BC_v. In addition, $extract(BC_v, M_v^{view})$ intends to extract a partial blockchain BC_v^{i+} from BC_v where i is determined by the $(f+s)$th highest seq searched in M_v^{view} and s is an adjustable constant to be determined later. For node recovery, the $update(BC_v, BC_u^{i+})$ function is designed to help a node v update BC_v by complementing missed blocks from the received BC_u^{i+}.

4.3.3 Data Aggregation and Reaggregation

In this subsection, we present two critical algorithms to realize data aggregation and reaggregation. The objective of data aggregation is to rapidly collect data from all followers to the leader in $O(\log n \log \Gamma)$ w.h.p. However, messages might be lost due to crash failures. Therefore, we propose the data reaggregation subroutine to remedy such a situation and ensure that the leader can completely aggregate the data from all non-faulty nodes within an epoch.

4.3.3.1 Broadcast-Oriented Communications

When blockchains are to be implemented in a wireless network, unicast and multicast are generally not needed. Instead, we should exploit the broadcast nature of the wireless medium. Decomposing a typical consensus process one can see that there exist three major communication patterns, namely one-to-many, many-to-one, and many-to-many, that heavily employ broadcast and data aggregation, the two communication primitives. Broadcast can improve blockchain's efficiency since a one-to-many communication only costs a one-time broadcast to disseminate a node's message to all its peers within the communication range. For many-to-many, the communication complexity is $O(N^2)$ for a network size of N if using unicast and multicast. With broadcast, the complexity can be reduced to $O(N)$.

However, to reach a consensus, medium contention and packet collision need to be carefully addressed in wireless networks. For this purpose we restrict the transmission probability to be p and transmission power to be P_i, which are formally described in Section 4.3.3 (also see line 6 in Algorithm. 7).

4.3.3.2 Data Aggregation

Leveraging a spanner, we propose the *DataAggregation*($data_v$) subroutine to aggregate the data level by level. As shown in Algorithm 7, a node v executing *DataAggregation*($data_v$) takes $data_v$ as input, generates a message m_v by *MSG(data)*, and initializes $M_v = \{m_v\}$. Then data aggregation proceeds by $\log\Gamma$ rounds. We denote by R_i the ith round. Recall that for $i = 1, 2, \cdots, \log\Gamma$, a node $v \in V_{i-1} \setminus V_i$ has a parent node $u \in V_i$ and $d(v,u) \leq r_i$. As a consequence, R_i is responsible for aggregating the data from $V_{i-1} \setminus V_i$ to V_i, and $R_{\log\Gamma}$ is the final round when the collector receives the data from $V_{\log\Gamma-1} \setminus V_{\log\Gamma}$.

Algorithm 7 *DataAggregation*($data_v$) Subroutine

```
1: function DataAggregation(data_v)
2:     Initially, m_v ← MSG(data_v), M_v = {m_v}
3:     ▷ In R_i(i = 1,2,··· ,logΓ):
4:     if v ∈ V_{i−1} \ V_i then
5:         for μ·logN slots do
6:             send M_v with probability p = 1/(σλ′) and power P_i = 2Nβr_i^α
7:     else
8:         if v ∈ V_i then
9:             for μ·logN slots do
10:                listen on the channel
11:                if receive a valid M_u then
12:                    M_v ← add(M_v, M_u)
```

In a specific round R_i, the nodes in $V_{i-1} \setminus V_i$ constantly send M_v for $\mu \cdot \log N$ slots with probability $p = \frac{1}{\sigma\lambda'}$, where $\lambda' = 25$, μ and σ are sufficiently large constants whose lower bounds are given in Section 4.4.1. Note that $\mu \cdot \log N$ is the optimal number of slots to ensure that without crash failures, the data from all child nodes can be completely aggregated to their parent w.h.p. The transmission power is set to be $P_i = 2\mathcal{N}\beta r_i^\alpha$ so that the transmission range of each node is $2^{1/\alpha} r_i$, where $r_i = 2^i$. This transmission range is slightly larger than r_i such that for any child v, its parent is within the unit disk centered at v with a radius r_i. This power control strategy improves child nodes' ability to resist interference outside the unit disk, thus contributing to the success of the transmissions. Moreover, a node $v \in V_i$ as a parent in R_i listens on the channel for $\mu \cdot \log N$ slots to receive messages from its children. If v receives a M_u from a child u, it appends M_u to M_v. The entire data aggregation process can be finished in $\mu \log N \log \Gamma$ slots, and the data from all non-faulty nodes can be aggregated w.h.p., which is proved in Section 4.4.1.

Algorithm 8 *Reaggregation*($data_v$) Subroutine

▷ as a leader
while *true* **do**
  ▷ slot one
  broadcast M_ℓ^{data}
  ▷ slot two
  listen on the channel
  ▷ slot three
  if sense noise $> \mathcal{N}$ in slot two **then**
    broadcast $m_\ell \leftarrow MSG(reaggregation_\ell)$
  else
    broadcast $m_\ell \leftarrow MSG(stop_\ell)$ and break
  ▷ data reaggregation
  wait for aggregated data from a collector
▷ as a follower
while *true* **do**
  ▷ slot one
  listen on the channel
  ▷ slot two
  if receive M_ℓ^{data} in slot one and $data_v \notin M_\ell^{data}$ **then**
    broadcast $m_v \leftarrow MSG(miss_v)$
  ▷ slot three
  listen on the channel
  ▷ data reaggregation
  if receive reaggregation message in slot three **then**
    run *SpannerConstruction*
    run *DataAggregation*($data_v$)
  else
    break

4.3.3.3 Reaggragation

The reaggregation subroutine has two stages: a three-slot integrity check stage (lines 3-11, lines 16-22) and a data reaggregation stage (lines 12-13, lines 23-28). We define a broadcast operation (used in lines 4/9/11/20) as transmitting a message with $\hat{P} = 2\mathcal{N}\beta r^{\alpha}_{\log\Gamma}$ so that a node listening on the channel can either receive a message from the sender or sense noise exceeding $\mathcal{N}$. Concretely, the integrity check stage intends to examine whether the leader loses any message from non-faulty nodes using physical carrier sensing. In slot one, the leader l broadcasts its current M_{ℓ}^{data} to the entire network, where M_{ℓ}^{data} is the message queue embodying the messages whose type is *data* (e.g., M_{ℓ}^{view} is the message queue of the view messages). Upon receiving M_{ℓ}^{data}, each node v examines if its $data_v$ is included in M_{ℓ}^{data}. If not, v broadcasts $m_v \leftarrow MSG(miss_v)$ so that in slot two the leader can get the notice saying that some messages are missed by sensing noise greater than $\mathcal{N}$, and broadcast $m_{\ell} \leftarrow MSG(reaggregation_{\ell})$ in slot three to start the second stage. In the data reaggregation stage, all nodes except the leader run *SpannerConstrcution* to reconstruct a spanner free from faulty nodes. The *SpannerConstrcution* procedure is the same as the one we illustrate in Section 4.2. The new spanner does not include the leader and it elects a new collector who is responsible for sending the aggregated data to the leader. Afterwards, the nodes whose messages are missed in M_{ℓ}^{data} run $DataAggregation(data_v)$. Only when no messages from non-faulty nodes are missed can the leader broadcast $m_{\ell} \leftarrow MSG(stop_{\ell})$ to end the reaggregation process.

Algorithm 9 Fast Fault-Tolerant Blockchain Protocol

▷ PREPARE
▷ as a leader
broadcast $m_\ell \leftarrow MSG(view_\ell)$
listen on the channel for $\mu \log N \log \Gamma$ slots
execute $Reaggregation(view_\ell)$
▷ as a follower
if receive $view_u$ from a leader **then**
    run $DataAggregation(view_v)$
else
    abandon the current epoch
execute $Reaggregation(view_v)$
▷ COMMIT
▷ as a leader
if $|\{m \in M_\ell^{view} | m.data = view_\ell\}| \geq f+1$ **then**
    broadcast $m_\ell \leftarrow MSG(correct_\ell)$
    listen on the channel for $\mu \log N \log \Gamma$ slots
execute $Reaggregation(tx_v)$
▷ as a follower
if receive $correct_\ell$ from a leader **then**
    run $DataAggregation(tx_v)$
else
    abandon the current epoch
execute $Reaggregation(tx_v)$
▷ DECIDE
▷ as a leader
$B_\ell \leftarrow pack(M_\ell^{tx})$, and $BC_\ell \leftarrow append(BC_\ell, B_\ell)$
broadcast $BC_\ell^{i+} \leftarrow extract(BC_\ell, M_\ell^{view})$
▷ as a follower
if receive BC_ℓ^{i+} from the leader **then**
    update(BC_v, BC_ℓ^{i+})
else
    abandon the current epoch

4.3.4 Fast Fault-Tolerant Blockchain Protocol

wChain is a three-phase protocol that can achieve consensus on the sequence of blocks and handle failures caused by wireless node crashes. Concretely, *wChain* leverages broadcast communications, data aggregation and reaggregation, which are all specifically designed for wireless networks. At each epoch, *wChain* proceeds by three phases, namely `PREPARE`, `COMMIT`, and `DECIDE`. In the following, we depict each phase to demonstrate how fast fault-tolerance can be achieved in *wChain*.

Prepare. The `PREPARE` phase intends to help a leader obtain a global view. Recall that when a spanner is constructed for the first time, *wChain* appoints the collector as the leader to take charge of the current epoch. As a leader, ℓ broadcasts $m_\ell \leftarrow MSG(view_\ell)$ to the entire network. Each follower v runs $DataAggregation(view_v)$ upon receiving the view information from ℓ in the previous slot. Otherwise, the follower abandons the current epoch. Note that all nodes should execute the reaggregation subroutine to ensure that the data from all non-faulty nodes are completely aggregated.

Commit. Denote by M_ℓ^{view} the message queue of ℓ embodying its view information. The requirement of $|\{m \in M_\ell^{view} | m.data = view_\ell\}| \geq f+1$ means that the leader should successfully receive no less than f view messages that have identical views as itself. That is also to say, at least $f+1$ nodes (f followers and one leader) have an identical view. If such a requirement is satisfied, the leader can broadcast a correct message and listen on the channel to receive transactions while the followers receiving a correct signal start transaction aggregation. The reaggregation subroutine is still executed to ensure the full aggregation of transactions.

Decide. When the leader ℓ receives all transactions from non-faulty nodes, it packs the transactions into a block B_ℓ, and appends the B_ℓ to its

local BC_ℓ. Then the leader ℓ executes $extract(BC_\ell, M_\ell^{view})$ to formulate a BC_ℓ^{i+}, which is used to help recover at least s nodes that have crashed in previous epochs and need to update their blockchains to become non-faulty. Each non-faulty follower v updates BC_v by running $update(BC_v, BC_\ell^{i+})$.

4.4 Protocol Analysis

In this section, we analyze the protocol in terms of efficiency of data aggregation and reaggregation, persistence, and liveness.

4.4.1 Efficiency of Data Aggregation and Reaggregation

Theorem 7. *The runtime of the data aggregation subroutine is upper bounded by $O(\log N \log \Gamma)$ slots w.h.p., and the runtime of the reaggregation subroutine is upper bounded by $O(f \log N \log \Gamma)$ slots w.h.p.*

Proof. We first present Lemma 7, which focuses on one slot in a given round R_i.

Lemma 7. *For a given slot in round R_i, if $v \in V_i$ is a parent node of some node in V_{i-1}, for any node $u \in V_{i-1} \cap N_{r_i}(v)$, if u transmits, v can receive the message with a constant probability.*

Proof. For a given slot in round R_i, we denote the aggregated transmission probability of $V_{i-1} \cap N_{r_i}(v)$ by $P_i(v) = \sum_{w \in V_{i-1} \cap N_{r_i}(v)} p_w$. Let's first prove that $P_i(v)$ can be bounded by $\frac{1}{\sigma}$, where σ is a sufficiently large constant. Due to the property of the maximum independent set, the disks of radius $r_i/2$ centered at any node $w \in V_{i-1} \cap N_{r_i}(v)$ are disjoint. We define density λ as the number of nodes in $V_{i-1} \cap N_{r_i}(v)$, then the upper bound of density is

$$\lambda \leq \frac{\pi[r_1 + r_0/2]^2}{\pi(r_0/2)^2} = 25, \tag{4.2}$$

where $\lambda = 25$ when $i = 1$. Since for each $w \in V_{i-1} \cap N_{r_i}(v)$, $p_w = \frac{1}{\sigma\lambda'}$, we have $P_i(v) = \lambda \cdot \frac{1}{\sigma\lambda'} \leq \frac{1}{\sigma}$, where $\lambda' = 25$.

Then we partition the whole space outside $D_{r_i}(v)$ into rings R_j for $j \geq 1$, where R_j is the ring with a distance in the range $[jr_i, (j+1)r_i]$ from v. Denote by S_j the set of nodes in V_{i-1} that also fall into R_j. Considering the property of maximum independent set, one can see that the disks of radius $r_i/2$ centered at the nodes in R_j are disjoint. Then we have

$$|S_j| \leq \frac{\pi[(j+1)r_i + r_i/2]^2 - \pi[jr_i - r_i/2]^2}{\pi(r_i/2)^2} \leq 24j. \tag{4.3}$$

Let $I(v,w)$ be the interference at v caused by w. Denote by I_{out} the interference caused by the nodes outside $D_{r_i}(v)$. Then one can calculate I_{out} as follows:

$$\begin{aligned} I_{out} &= \sum_{j=1}^{\infty} \sum_{w \in S_j} I(v,w) = \sum_{j=1}^{\infty} \sum_{w \in S_j} \frac{P_i}{d(v,w)^\alpha} \cdot p \\ &\leq \sum_{j=1}^{\infty} |S_j| \frac{1}{\sigma} \cdot \frac{2\mathcal{N}\beta r_i^\alpha}{(jr_i)^\alpha} \leq \frac{24\beta(\alpha-1)}{\sigma(\alpha-2)} \cdot N \leq N/2, \end{aligned} \tag{4.4}$$

where the last inequality holds when $\sigma > \frac{48\beta(\alpha-1)}{(\alpha-2)}$. Considering the case when u is the only node that transmits in the current slot, since $u \in V_{i-1} \cap N_{r_i}(v)$, $d(v,u) \leq r_i$, we have

$$SINR(v,u) = \frac{\frac{P_i}{d(v,u)^\alpha}}{N + I_{out}} \geq \frac{\frac{2\mathcal{N}\beta r_i^\alpha}{r_i^\alpha}}{N + N/2} \geq \beta, \tag{4.5}$$

which indicates that if u is the only node who transmits, v can receive the message. Then we bound the probability that u is the only transmitting

node. Since $P_i(v) \leq \frac{1}{\sigma}$, the probability that only u transmits at each slot is

$$\begin{aligned} p_u \prod_{w \in V_i \cap N_{r_i}(v) \setminus u} (1-p_w) &\geq p_u \prod_{w \in V_i \cap N_{r_i}(v)} (1-p_w) \\ &\geq p_u \prod_{w \in V_i \cap N_{r_i}(v)} e^{\frac{-p_w}{1-p}} \\ &= p e^{\frac{-P_i(v)}{1-p}} \\ &\geq (\sigma\lambda')^{-1} e^{\frac{-\lambda'}{\sigma\lambda'-1}} \\ &\in \Omega(1). \end{aligned} \tag{4.6}$$

This implies that with a constant probability, v can receive u's message.

□

In Algorithm 7, each round consists of a fixed number of $\mu \cdot \log N$ slots. At each slot, a child $u \in V_{i-1} \setminus V_i$ transmits constantly with probability $p = \frac{1}{\sigma\lambda'}$. Lemma 7 indicates that at each slot, u can succeed in sending a message to its parent v with a constant probability denoted by $\hat{p}$. Thus, by applying the Chernoff bound (see Appendix), the probability that u succeeds in sending a message to its parent after $\mu \cdot \log N$ slots is $1-(1-\hat{p})^{\mu \log N} \geq 1-e^{-\hat{p}\mu \log N} \geq 1-N^{-2}$ if $\mu \geq 2/\hat{p}$. Since the density of the active nodes is bounded by $\lambda' = 25$, the probability that all children succeed is $(1-N^2)^{\lambda'}$.

Next, assume that the nodes have synchronized clocks. Then at each round R_i, the nodes in different independent sets can send messages to their parents at the same time. The probability that the data has been aggregated to all parent nodes in R_i is at least $(1-N^2)^{\lambda' V_{i+1}} \geq (1-N^{-1})$ since $\lambda' V_{i+1} < N$. Thus, the one-round data aggregation succeeds in $O(\log N)$ w.h.p. Considering the $(\log \Gamma)$-round aggregation process, one can immediately derive that data aggregation succeeds in $O(\log N \log \Gamma)$ w.h.p.

Unlike a normal data aggregation, Algorithm 0 terminates when the

messages from all non-faulty nodes are received by the collector without any loss. A message can be lost when crash failure happens. The number of faulty nodes is bounded by f, thus during the data reaggregation process, there are at most f times of the execution of the spanner construction and data aggregation, which gives the upper bound of the runtime as $O(f \log N \log \Gamma)$. □

4.4.2 Persistence and Liveness

In this subsection, we demonstrate how our protocol ensures persistence and liveness properties whose definitions are adapted from the rigorous ones proposed by Garay *et al.* [41].

Theorem 8. ***Persistence.*** *If a non-faulty node v proclaims a transaction tx_v in the position tx_i^j, other nodes, if queried, should report the same result.*

Proof. To prove the persistence property, we need to show that for any two blockchains BC_v and BC_u of nodes v and u, respectively, one cannot find two different transactions $tx_v \in BC_v$ and $tx_u \in BC_u$ that are in the same position tx_i^j. To prove by contradiction, we assume that such tx_v and tx_u exist, and there are two cases when the assumption can hold.

C1: tx_v and tx_u are respectively appended to blockchains BC_v and BC_u at the same epoch. This indicates that a leader broadcasts two different blocks in the same epoch, which is not permissible in *wChain*, thus contradicting our assumption.

C2: tx_v and tx_u are appended to their corresponding blockchains BC_v and BC_u in two different epochs e_m and e_n. Let tx_i^j also denote the transaction generated for the first time in position tx_i^j and appended to the blockchains of at least $f+1$ nodes in e_i. Since a leader cannot broadcast two different blocks in the same epoch, the nodes who append tx_i^j to their local blockchain

in e_i should have an identical view of tx_i^j. Without loss of generality, assume $i < m < n$. Using contradiction, we assume $tx_v \neq tx_i^j$. Since $i < m$, v should crash before e_m and recover in e_m so that tx_v is appended to BC_v when v updates its blockchain by applying $update()$. The leader who sends the update information in e_m has an identical view with at least f nodes, which means that at least $f+1$ nodes have the same view on tx_i^j in e_m. Since there are also at least $f+s$ nodes who agree on tx_i^j in e_i and the network size $N = 2f+1$, we have a contradiction saying that $N > 2f+1+s > N$. That is, one can only have $tx_i^j = tx_v$. By applying the same proof, we can obtain $tx_i^j = tx_u$. Hence $tx_i^j = tx_v = tx_u$, which contradicts the assumption that tx_v and tx_u are different.

In a nutshell, all the nodes queried for a transaction in a specific position should report the same result or report error messages. □

Theorem 9. ***Liveness.*** *If a non-faulty node generates a transaction and contends to send it, wChain can add it to the blockchains within T slots w.h.p., where the upper bound of T is $O(\log N \log \Gamma)$ when crash failures happen in a low frequency, and the worst-case upper bound of T is $O(f \log N \log \Gamma)$.*

Proof. In a specific epoch, the best case for *wChain* occurs when no crash failures happen for all nodes throughout the epoch so that the leader executing the $Reaggregation(data_\ell)$ subroutine can broadcast $m_\ell \leftarrow MSG(stop_\ell)$ without the need of waiting for the aggregated data from a new collector. By Theorem 7, the view messages and the transactions can be fully aggregated within $O(\log N \log \Gamma)$ w.h.p. Besides, the `DECIDE` phase takes $O(1)$ slots. Hence the upper bound of T is $O(\log N \log \Gamma)$. In a normal case, failures happen in a low frequency $\in O(1)$ so that data aggregation can be executed in $O(1)$ time during the data reaggregation process, and the `PREPARE` and `COMMIT` phases take $O(\log N \log \Gamma)$ slots in total. Therefore, the upper bound of T is

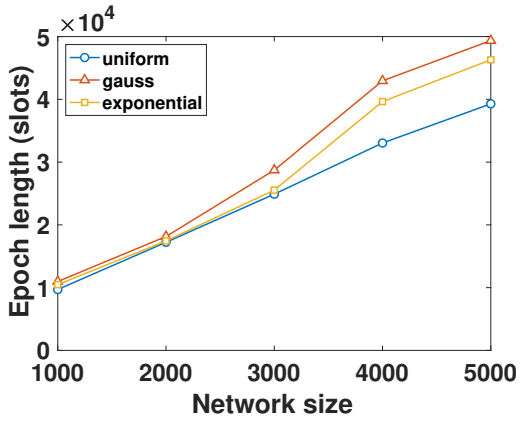
still $O(\log N \log \Gamma)$.

From the worst-case perspective, we assume that before epoch e_i, nodes are all non-faulty, and f nodes crash during e_i. If a leader crashes, the worst-case runtime of the current epoch is still $O(\log N \log \Gamma)$ when it crashes in the `DECIDE` phase. If a follower v crashes, some data from v's children cannot be collected to the leader such that the leader must execute *SpannerConstruction* and $DataAggregation(data_v)$, which take $O(\log N \log \Gamma)$ slots. Thus the extra runtime brought by a one-time crash failure is $O(\log N \log \Gamma)$. If f nodes crash, the total time spent on waiting for a successful `DECIDE` phase is bounded by $O(f \log N \log \Gamma)$. This gives the normal and worst-case upper bound of T as $O(\log N \log \Gamma)$ and $O(f \log N \log \Gamma)$, respectively. □

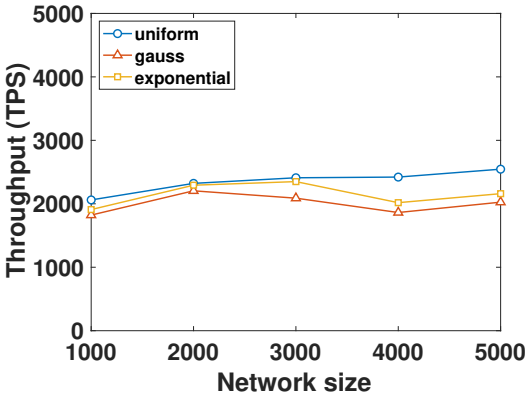
4.5 Simulation Results

In this section, we conduct simulation experiments to validate the performance of *wChain*. The impacts of various parameters are investigated, including the SINR model parameters, network size N, and Γ, the ratio of the maximum distance to the minimum distance between nodes. If not stated otherwise, we adopt the following parameter settings: $\alpha = 3$, $\beta = 3, s = 100, \lambda' = 25, \mathcal{N} = 1, P_i = 2\mathcal{N}\beta 2^{i\alpha}$. The frequency of node crashes is set to be $1\% \times N$ nodes per second which corresponds to a moderate crash rate in real-world settings. To evaluate the performance, we adopt two metrics, namely *epoch length* and *throughput*. The epoch length is the number of slots within an epoch; given that the unit slot time for IEEE 802.11 is set to be $50\mu s$, one can calculate throughput as

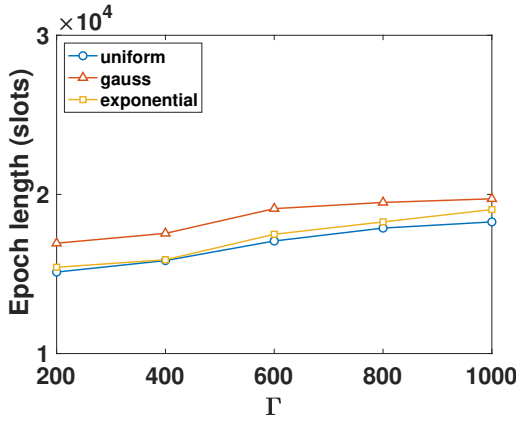
$$\text{Throughput} = \frac{\textit{The number of transactions}}{\textit{Epoch length} \times 50\mu s}, \tag{4.7}$$



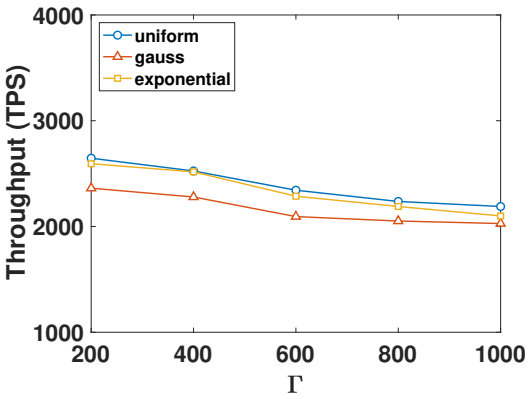
(a) 150×150 plane



(b) 150×150 plane



(c) $N = 2000$



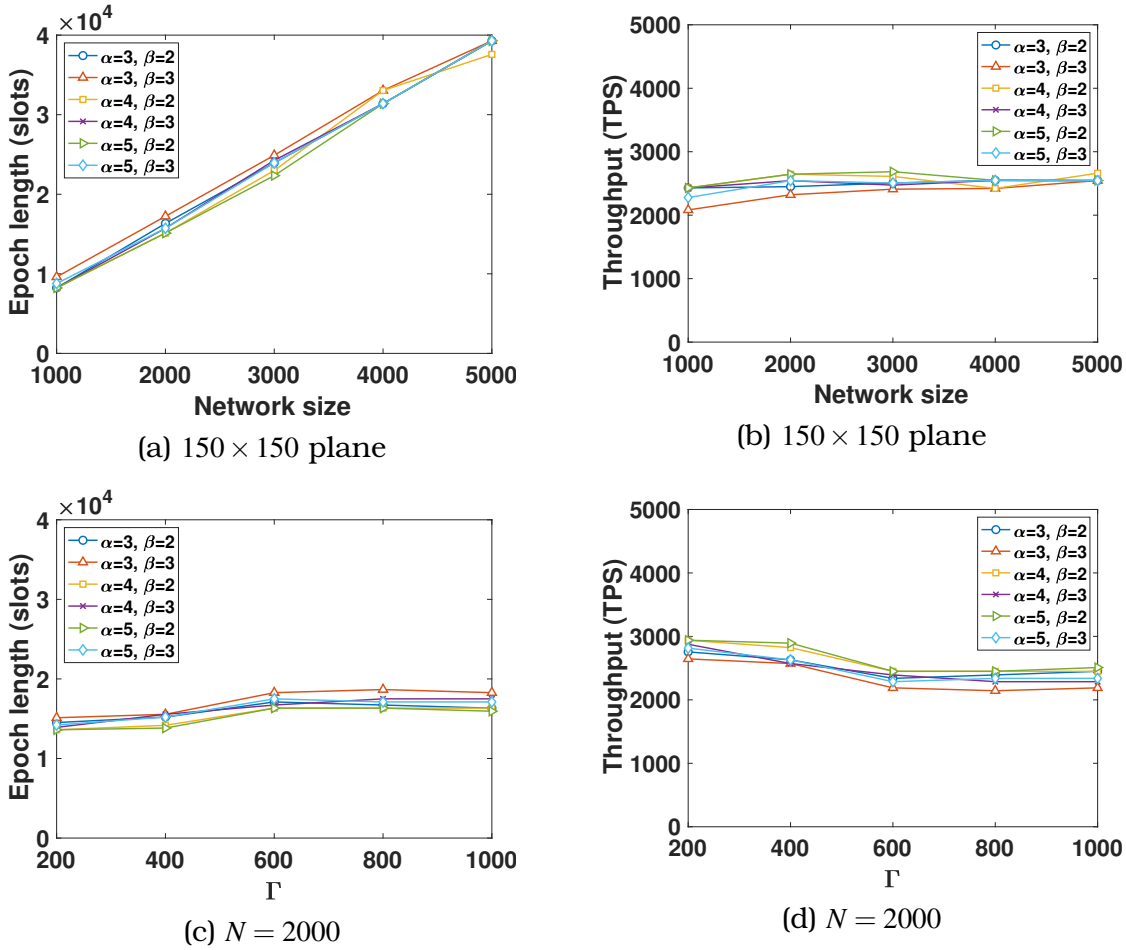
(d) $N = 2000$

Figure 4.3: The performance of *wChain* *vs.* the network size N and Γ (under uniform, normal, and exponential distributions).

hence the unit of throughput is transactions per second (TPS).

The simulation program is written in C and all the experiments are performed under a CentOS 7 operating system running on a machine with an Intel Xeon 3.4 GHz CPU, 120 GB RAM, and 1 TB SATA Hard Drive. Over 20 runs are carried out to get the average for each result.

4.5.1 Impacts of Network Size and Γ



(a) 150×150 plane

(b) 150×150 plane

(c) $N = 2000$

(d) $N = 2000$

Figure 4.4: The performance of *wChain* with various α and β (under a uniform distribution).

Since liveness is mainly determined by N and Γ, we first study the impacts of N and Γ on the performance of *wChain*. We consider three types

of distributions, namely uniform, normal, and exponential. To investigate the impacts of N, we adopt parameters $\alpha = 3, \beta = 3$ for the SINR model, and the plane is of size 150×150. The results are reported in Figure 4.3.

One can observe from Figure 4.3a that the epoch length increases with N. With a uniform distribution and $N = 5000$, it is 49364 slots (about 2.47s). Under normal and exponential distributions, the nodes have larger epoch lengths since they are denser in the center or the corner. They may suffer from heavier contention and spend more time transmitting a message. This result is consistent with our model assumption which states that the network density should be limited.

Figure 4.3b indicates that under uniform distributions *wChain* has the highest throughput, which only increases with N. When $N = 5000$, the throughput with uniform distributions reaches 2546 TPS and is about 28% higher than that for normal distributions. Under normal distributions, the throughput can reach 1986 TPS when $N = 5000$. Note that that the throughput under normal or exponential distributions has a small decrease from $N = 3000$. This is because the density is so high in some areas with a large number of nodes that contend heavily, negatively affecting throughput.

Then we investigate the impact of Γ and set $N = 2000$. In Figure 4.3c, with an increasing Γ, the epoch length increases since the spanner has more levels. The epoch lengths under normal and exponential distributions are still larger than the one under uniform distributions, which is caused by the same reason as Figure 4.3a.

Concerning that the throughput is a function of Γ, our protocol running under uniform distributions yields the largest throughput. However, the throughput decreases with a larger Γ because a larger Γ indicates that the spanner has more levels, and the data aggregation process takes a longer

time.

4.5.2 Impacts of the SINR model parameters

We perform four experiments to explore how the SINR model parameters, namely α and β, impact the performance of *wChain*. Assume that nodes are uniformly distributed in the plane. We choose the combinations of $\alpha = 3,4,5$ and $\beta = 2,3$, which are tested with different N and Γ in our experiments. Such combinations are chosen due to the fact that $\alpha = 3,4,5$ can cover various environments, which was demonstrated by Akpaida *et al.* [3]. In addition, simulations with $\beta = 2,3$ can cover a wide range of hardware since common hardware is sensitive enough to have β close to 1 (the more sensitive the hardware, the smaller the β).

As shown in Figure 4.4a, the epoch lengths scale from about 1×10^4 to 4×10^4 slots with N under different settings of α and β but they differ very little for different α and β and the same N. In 4.4b, even though we observe a small throughput decrease when $\alpha = 3$ and $\beta = 3$, the epoch length and throughput are independent of the values of α and β with different N.

In Figure 4.4c and 4.4d, the epoch length and throughput slightly change with different α and β. This is because α, as the path-loss exponent, works closely with the distance between nodes as well as Γ. The impacts of α and β on throughput are in an allowable range so that one can claim that the performance of *wChain* is insensitive to them.

Chapter 5: Conclusion and Future Directions

In this thesis, we first propose the General Blockchain System Analytic Framework (GBSAF) that generalizes blockchain systems, defines major components and internal interactions and supports component-wise analysis of complex blockchain systems. This work gives us inspiration for designing wireless blockchains. Then we propose a 2-phase blockchain protocol, namely BLOWN. BLOWN establishes a jamming-resistant communication channel and combines the Sortition algorithm and our newly proposed PoC consensus algorithm for efficient and secure leader election. Besides, BLOWN prevents double-spending attacks and Sybil attacks. Analysis and simulation results demonstrate the efficiency, effectiveness, and security properties of the BLOWN protocol. As a complementary protocol to BLOWN, *wChain* is a fast fault-tolerant blockchain protocol, which can ensure the fast data aggregation leveraging a spanner as the communication backbone. The runtime upper bound of our protocol is $O(\log N \log \Gamma)$ when crash failures happen in a low frequency, and the worst-case upper bound of *wChain* is $O(f \log N \log \Gamma)$. Besides, *wChain* tolerates at most $f = \lfloor \frac{N}{2} \rfloor$ faulty nodes and is capable of handling node recovery while satisfying persistence and liveness, the two crucial properties for a blockchain protocol to function well. Both theoretical analysis and simulation studies are conducted to validate our design.

In the future, we intend to propose a formalized version of GBSAF, address more details of each component, define standards based on the framework, provide a systematic blockchain design reference manual. We would also like to investigate the byzantine fault-tolerant versions of BLOWN and *wChain*,

consider more realistic problems caused by fading channels and obstacles explore wireless blockchains in the 5/6G-enabled networks. To conduct a comprehensive comparison between heterogeneous wired and wireless blockchains, it is worth building up a simulator based on existing platforms (e.g., OMNeT++, ns-3). Such a simulator is also helpful in implementing wireless blockchains in real-world cases, such as trusted, intelligent swarms (fully decentralized and in need of trustlessness).

Bibliography

[1] Adnan Abu-Dayya. Determining sinr in a communications system, November 23 1999. US Patent 5,991,273.

[2] Marcos K Aguilera, Naama Ben-David, Irina Calciu, Rachid Guerraoui, Erez Petrank, and Sam Toueg. Passing messages while sharing memory. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 51–60, 2018.

[3] VOA Akpaida, FI Anyasi, Stanley Idiake Uzairue, and AI Idim. Determination of an outdoor path loss model and signal penetration level in some selected modern residential and office apartments in ogbomosho, oyo state, nigeria. *Journal of Engineering Research and Reports*, pages 1–25, 2018.

[4] Mustafa Al-Bassam, Alberto Sonnino, Shehar Bano, Dave Hrycyszyn, and George Danezis. Chainspace: A sharded smart contracts platform. *arXiv preprint arXiv:1708.03778*, 2017.

[5] Amjad Aldweesh and Aad van Moorsel. A survey about blockchain software architectures. In *Newcastle Univ*, 2016.

[6] Muhammad Salek Ali, Massimo Vecchio, Miguel Pincheira, Koustabh Dolui, Fabio Antonelli, and Mubashir Husain Rehmani. Applications of blockchains in the internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 21(2):1676–1717, 2018.

[7] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, page 30. ACM, 2018.

[8] Stefano De Angelis, Leonardo Aniello, Roberto Baldoni, Federico Lombardi, Andrea Margheri, and Vladimiro Sassone. Pbft vs proof-of-authority: applying the cap theorem to permissioned blockchain. In *Italian Conference on Cyber Security (06/02/18)*, January 2018.

[9] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. A survey of attacks on ethereum smart contracts (sok). In *International Conference on Principles of Security and Trust*, pages 164–186. Springer, 2017.

[10] Baruch Awerbuch, David Holmer, Cristina Nita-Rotaru, and Herbert Rubens. An on-demand secure routing protocol resilient to byzantine

failures. In *Proceedings of the 1st ACM workshop on Wireless security*, pages 21–30, 2002.

[11] Tuncer C Aysal, Anand D Sarwate, and Alexandros G Dimakis. Reaching consensus in wireless networks with probabilistic broadcast. In *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 732–739. IEEE, 2009.

[12] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. Enabling blockchain innovations with pegged sidechains. *URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains*, page 72, 2014.

[13] Leemon Baird. Hashgraph consensus: fair, fast, byzantine fault tolerance. *Swirlds Tech Report, Tech. Rep.*, 2016.

[14] Paul Baran. On distributed communications networks. *IEEE transactions on Communications Systems*, 12(1):1–9, 1964.

[15] Massimo Bartoletti, Stefano Lande, Livio Pompianu, and Andrea Bracciali. A general framework for blockchain analytics. In *Proceedings of the 1st Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*, page 7. ACM, 2017.

[16] Georg Becker. Merkle signature schemes, merkle trees and their cryptanalysis. *Ruhr-University Bochum, Tech. Rep*, 2008.

[17] Iddo Bentov, Rafael Pass, and Elaine Shi. Snow white: Provably secure proofs of stake. *IACR Cryptology ePrint Archive*, 2016:919, 2016.

[18] Richard Gendal Brown, James Carlyle, Ian Grigg, and Mike Hearn. Corda: an introduction. *R3 CEV, August*, 2016.

[19] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 315–334. IEEE, 2018.

[20] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437*, 2017.

[21] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145, 2001.

[22] R. Canetti. Universally composable signature, certification, and authentication. In *Proceedings. 17th IEEE Computer Security Foundations Workshop, 2004.*, pages 219–233, 2004.

[23] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.

[24] Nutthakorn Chalaemwongwan and Werasak Kurutach. State of the art and challenges facing consensus protocols on blockchain. In *2018 International Conference on Information Networking (ICOIN)*, pages 957–962. IEEE, 2018.

[25] Wei Cheng, Jiguo Yu, Feng Zhao, and Xiuzhen Cheng. Ssdnet: Small-world super-dense device-to-device wireless networks. *IEEE Network*, 32(1):186–192, 2017.

[26] Gregory Chockler, Murat Demirbas, Seth Gilbert, Calvin Newport, and Tina Nolte. Consensus and collision detectors in wireless ad hoc networks. In *Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, pages 197–206, 2005.

[27] Anton Churyumov. Byteball: A decentralized system for storage and transfer of value. *URL https://byteball. org/Byteball. pdf*, 2016.

[28] Yueyue Dai, Du Xu, Sabita Maharjan, Zhuang Chen, Qian He, and Yan Zhang. Blockchain and deep reinforcement learning empowered intelligent 5g beyond. *IEEE Network*, 33(3):10–17, 2019.

[29] Yueyue Dai, Du Xu, Sabita Maharjan, Zhuang Chen, Qian He, and Yan Zhang. Blockchain and deep reinforcement learning empowered intelligent 5g beyond. *IEEE Network*, 33(3):10–17, 2019.

[30] Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 66–98. Springer, 2018.

[31] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*, pages 1–10. IEEE, 2013.

[32] Tien Tuan Anh Dinh, Rui Liu, Meihui Zhang, Gang Chen, Beng Chin Ooi, and Ji Wang. Untangling blockchain: A data processing view of blockchain systems. *IEEE Transactions on Knowledge and Data Engineering*, 30(7):1366–1385, 2018.

[33] Qi Dong and Donggang Liu. Resilient cluster leader election for wireless sensor networks. In *2009 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 1–9. IEEE, 2009.

[34] John R Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer, 2002.

[35] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)*, 35(2):288–323, 1988.

[36] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In *Annual Cryptology Conference*, pages 585–605. Springer, 2015.

[37] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, 61(7):95–102, 2018.

[38] Jie Feng, F Richard Yu, Qingqi Pei, Jianbo Du, and Li Zhu. Joint optimization of radio and computational resources allocation in blockchain-enabled mobile edge computing systems. *IEEE Transactions on Wireless Communications*, 2020.

[39] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. Technical report, Massachusetts Inst of Tech Cambridge lab for Computer Science, 1982.

[40] Weichao Gao, William G Hatcher, and Wei Yu. A survey of blockchain: techniques, applications, and challenges. In *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–11. IEEE, 2018.

[41] 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*, pages 281–310. Springer, 2015.

[42] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 51–68, 2017.

[43] Zbigniew Gołębiewski, Marek Klonowski, Michał Koza, and Mirosław Kutyłowski. Towards fair leader election in wireless networks. In *International Conference on Ad-Hoc Networks and Wireless*, pages 166–179. Springer, 2009.

[44] Maurice Herlihy. Atomic cross-chain swaps. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 245–254. ACM, 2018.

[45] Garrick Hileman and Michel Rauchs. Global blockchain benchmarking study. *Cambridge Centre for Alternative Finance, University of Cambridge*, 122, 2017.

[46] Josep Colom Ikuno, Stefan Pendl, Michal Šimko, and Markus Rupp. Accurate sinr estimation model for system level simulation of lte networks. In *2012 IEEE International Conference on Communications (ICC)*, pages 1471–1475. IEEE, 2012.

[47] Daniel R Jeske and Ashwin Sampath. Method of estimating a signal-to-interference+ noise ratio (sinr), February 27 2007. US Patent 7,184,497.

[48] Kostis Karantias, Aggelos Kiayias, and Dionysis Zindros. Proof-of-burn. In *International Conference on Financial Cryptography and Data Security*, pages 523–540. Springer, 2020.

[49] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2014.

[50] Aggelos Kiayias and Giorgos Panagiotakos. Speed-security tradeoffs in blockchain protocols. *IACR Cryptol. ePrint Arch.*, 2015:1019, 2015.

[51] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*, pages 357–388. Springer, 2017.

[52] Aggelos Kiayias and Dionysis Zindros. Proof-of-work sidechains. Technical report, Cryptology ePrint Archive, Report 2018/1048, 2018.

[53] Aggelos Kiayias, Dionysis Zindros, et al. Proof-of-stake sidechains. In *Proof-of-Stake Sidechains*, page 0. IEEE, 2018.

[54] T. Kim, R. Goyat, M. K. Rai, G. Kumar, W. J. Buchanan, R. Saha, and R. Thomas. A novel trust evaluation process for secure localization using a decentralized blockchain in wireless sensor networks. *IEEE Access*, 7:184133–184144, 2019.

[55] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. Omniledger: A secure, scale-out, decentralized ledger via sharding. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 583–598. IEEE, 2018.

[56] Kari Korpela, Jukka Hallikas, and Tomi Dahlberg. Digital supply chain transformation toward blockchain integration. In *proceedings of the 50th Hawaii international conference on system sciences*, 2017.

[57] Fabian Kuhn, Nancy Lynch, and Calvin Newport. The abstract mac layer. In *International Symposium on Distributed Computing*, pages 48–62. Springer, 2009.

[58] James F Kurose. *Computer networking: A top-down approach featuring the internet, 3/E*. Pearson Education India, 2005.

[59] Jae Kwon. Tendermint: Consensus without mining. *Draft v. 0.6, fall*, 1:11, 2014.

[60] Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. Inclusive block chain protocols. In *International Conference on Financial Cryptography and Data Security*, pages 528–547. Springer, 2015.

[61] Xiaoqi Li, Peng Jiang, Ting Chen, Xiapu Luo, and Qiaoyan Wen. A survey on the security of blockchain systems. *Future Generation Computer Systems*, 2017.

[62] Zhetao Li, Jiawen Kang, Rong Yu, Dongdong Ye, Qingyong Deng, and Yan Zhang. Consortium blockchain for secure energy trading in industrial internet of things. *IEEE transactions on industrial informatics*, 14(8):3690–3700, 2017.

[63] Mengting Liu, Yinglei Teng, F Richard Yu, Victor CM Leung, and Mei Song. A deep reinforcement learning-based transcoder selection framework for blockchain-enabled wireless d2d transcoding. *IEEE Transactions on Communications*, 2020.

[64] Mengting Liu, F Richard Yu, Yinglei Teng, Victor CM Leung, and Mei Song. Joint computation offloading and content caching for wireless blockchain networks. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 517–522. IEEE, 2018.

[65] Zhuotao Liu, Yangxi Xiang, Jian Shi, Peng Gao, Haoyu Wang, Xusheng Xiao, Bihan Wen, and Yih-Chun Hu. Hyperservice: Interoperability and programmability across heterogeneous blockchains. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 549–566. ACM, 2019.

[66] Ziyao Liu, Nguyen Cong Luong, Wenbo Wang, Dusit Niyato, Ping Wang, Ying-Chang Liang, and Dong In Kim. A survey on blockchain: a game theoretical perspective. *IEEE Access*, 7:47615–47643, 2019.

[67] Marta Lokhava, Giuliano Losa, David Mazières, Graydon Hoare, Nicolas Barry, Eli Gafni, Jonathan Jove, Rafał Malinowsky, and Jed McCaleb. Fast and secure global payments with stellar. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, pages 80–96, 2019.

[68] Yang Lu. Blockchain: A survey on functions, applications and open issues. *Journal of Industrial Integration and Management*, 3(04):1850015, 2018.

[69] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 17–30. ACM, 2016.

[70] Nisha Malik, Priyadarsi Nanda, Xiangjian He, and Ren Ping Liu. Vehicular networks with security and trust management solutions: proposed secured message exchange via blockchain technology. *Wirel. Networks*, 26(6):4207–4226, 2020.

[71] Ralph C Merkle. A digital signature based on a conventional encryption function. In *Conference on the theory and application of cryptographic techniques*, pages 369–378. Springer, 1987.

[72] Andrew Miller, Ari Juels, Elaine Shi, Bryan Parno, and Jonathan Katz. Permacoin: Repurposing bitcoin work for data preservation. In *2014 IEEE Symposium on Security and Privacy*, pages 475–490. IEEE, 2014.

[73] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of bft protocols. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 31–42, 2016.

[74] Du Mingxiao, Ma Xiaofeng, Zhang Zhe, Wang Xiangwei, and Chen Qijun. A review on consensus algorithm of blockchain. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2567–2572. IEEE, 2017.

[75] Henrique Moniz, Nuno F Neves, and Miguel Correia. Byzantine fault-tolerant consensus in wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 12(12):2441–2454, 2012.

[76] Tal Moran and Ilan Orlov. Proofs of space-time and rational proofs of storage. *IACR Cryptology ePrint Archive*, 2016:35, 2016.

[77] Ujan Mukhopadhyay, Anthony Skjellum, Oluwakemi Hambolu, Jon Oakley, Lu Yu, and Richard Brooks. A brief survey of cryptocurrency

systems. In *2016 14th annual conference on privacy, security and trust (PST)*, pages 745–752. IEEE, 2016.

[78] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, Manubot, 2019.

[79] Calvin Newport. Consensus with an abstract mac layer. In *Proceedings of the 2014 ACM symposium on Principles of distributed computing*, pages 66–75, 2014.

[80] Calvin Newport and Peter Robinson. Fault-tolerant consensus with an abstract mac layer. *arXiv preprint arXiv:1810.02848*, 2018.

[81] James Newsome, Elaine Shi, Dawn Song, and Adrian Perrig. The sybil attack in sensor networks: analysis & defenses. In *Third international symposium on information processing in sensor networks, 2004. IPSN 2004*, pages 259–268. IEEE, 2004.

[82] Oscar Novo. Blockchain meets iot: An architecture for scalable access management in iot. *IEEE Internet of Things Journal*, 5(2):1184–1195, 2018.

[83] Kelly Olson, Mic Bowman, James Mitchell, Shawn Amundson, Dan Middleton, and Cian Montgomery. Sawtooth: an introduction. *The Linux Foundation*, 2018.

[84] Kim Joseph Olszewski. Sinr measurement method for ofdm communications systems, August 21 2007. US Patent 7,260,054.

[85] Oluwakayode Onireti, Lei Zhang, and Muhammad Ali Imran. On the viable area of wireless practical byzantine fault tolerance (pbft) blockchain networks. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2019.

[86] Kazim Rifat Ozyilmaz, Harsh Patel, and Ankit Malik. Split-scale: Scaling bitcoin by partitioning the UTXO space. *CoRR*, abs/1809.08473, 2018.

[87] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 643–673. Springer, 2017.

[88] Valentin Poirot, Beshr Al Nahas, and Olaf Landsiedel. Paxos made wireless: Consensus in the air. In *EWSN*, pages 1–12, 2019.

[89] Serguei Popov. The tangle. *cit. on*, page 131, 2016.

[90] Vaskar Raychoudhury, Jiannong Cao, and Weigang Wu. Top k-leader election in wireless ad hoc networks. In *2008 Proceedings of 17th International Conference on Computer Communications and Networks*, pages 1–6. IEEE, 2008.

[91] Yongjun Ren, Yepeng Liu, Sai Ji, Arun Kumar Sangaiah, and Jin Wang. Incentive mechanism of data storage based on blockchain for wireless sensor networks. *Mobile Information Systems*, 2018, 2018.

[92] Andrea Richa, Christian Scheideler, Stefan Schmid, and Jin Zhang. A jamming-resistant mac protocol for multi-hop wireless networks. In *International Symposium on Distributed Computing*, pages 179–193. Springer, 2010.

[93] Andrea Richa, Christian Scheideler, Stefan Schmid, and Jin Zhang. Self-stabilizing leader election for single-hop wireless networks despite jamming. In *Proceedings of the Twelfth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 1–10, 2011.

[94] Marten Risius and Kai Spohrer. A blockchain research framework. *Business & Information Systems Engineering*, 59(6):385–409, 2017.

[95] Tara Salman, Maede Zolanvari, Aiman Erbad, Raj Jain, and Mohammed Samaka. Security services using blockchains: A state of the art survey. *IEEE Communications Surveys & Tutorials*, 21(1):858–880, 2018.

[96] Lakshmi Siva Sankar, M Sindhu, and M Sethumadhavan. Survey of consensus protocols on blockchain applications. In *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 1–5. IEEE, 2017.

[97] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 515–532. Springer, 2016.

[98] David Schwartz, Noah Youngs, Arthur Britto, et al. The ripple protocol consensus algorithm. *Ripple Labs Inc White Paper*, 5, 2014.

[99] Gesualdo Scutari and Sergio Barbarossa. Distributed consensus over wireless sensor networks affected by multipath fading. *IEEE Transactions on Signal Processing*, 56(8):4100–4106, 2008.

[100] Pallavi Sethi and Smruti R Sarangi. Internet of things: architectures, protocols, and applications. *Journal of Electrical and Computer Engineering*, 2017, 2017.

[101] Atul Singh et al. Eclipse attacks on overlay networks: Threats and defenses. In *In IEEE INFOCOM*. Citeseer, 2006.

[102] Y Sompolinsky and A Zohar. Phantom and ghostdag: A scalable generalization of nakamoto consensus. Technical report, IACR Cryptology ePrint Archive, Report 2018, 2018.

[103] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. Spectre: A fast and scalable cryptocurrency protocol. *IACR Cryptology ePrint Archive*, 2016:1159, 2016.

[104] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 507–527. Springer, 2015.

[105] Beatriz Soret, Klaus I Pedersen, Niels TK Jørgensen, and Víctor Fernández-López. Interference coordination for dense wireless networks. *IEEE Communications Magazine*, 53(1):102–109, 2015.

[106] Yao Sun, Lei Zhang, Gang Feng, Bowen Yang, Bin Cao, and Muhammad Imran. Performance analysis for blockchain driven wireless iot systems based on tempo-spatial model. In *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 348–353. IEEE, 2019.

[107] Yao Sun, Lei Zhang, Gang Feng, Bowen Yang, Bin Cao, and Muhammad Ali Imran. Blockchain-enabled wireless internet of things: Performance analysis and optimal communication node deployment. *IEEE Internet of Things Journal*, 6(3):5791–5802, 2019.

[108] Alex Tapscott and Don Tapscott. How blockchain is changing finance. *Harvard Business Review*, 1(9):2–5, 2017.

[109] Bruno Tavares, Filipe Figueiredo Correia, André Restivo, João Pascoal Faria, and Ademar Aguiar. A survey of blockchain frameworks and applications. In *International Conference on Soft Computing and Pattern Recognition*, pages 308–317. Springer, 2018.

[110] Florian Tschorsch and Björn Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys & Tutorials*, 18(3):2084–2123, 2016.

[111] Sudarshan Vasudevan, Brian DeCleene, Neil Immerman, Jim Kurose, and Don Towsley. Leader election algorithms for wireless ad hoc networks. In *Proceedings DARPA Information Survivability Conference and Exposition*, volume 1, pages 261–272. IEEE, 2003.

[112] Jiaping Wang and Hao Wang. Monoxide: Scale out blockchains with asynchronous consensus zones. In *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*, pages 95–112, 2019.

[113] Wenbo Wang, Dinh Thai Hoang, Peizhao Hu, Zehui Xiong, Dusit Niyato, Ping Wang, Yonggang Wen, and Dong In Kim. A survey on consensus mechanisms and mining strategy management in blockchain networks. *IEEE Access*, 7:22328–22370, 2019.

[114] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014.

[115] Yang Xiao, Ning Zhang, Wenjing Lou, and Y Thomas Hou. A survey of distributed consensus protocols for blockchain networks. *arXiv preprint arXiv:1904.04098*, 2019.

[116] Zehui Xiong, Yang Zhang, Dusit Niyato, Ping Wang, and Zhu Han. When mobile blockchain meets edge computing. *IEEE Communications Magazine*, 56(8):33–39, 2018.

[117] Maofan Yin, Dahlia Malkhi, Michael K Reiter, Guy Golan Gueta, and Ittai Abraham. Hotstuff: Bft consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 347–356, 2019.

[118] D. Yu, L. Ning, Y. Zou, J. Yu, X. Cheng, and F. C. M. Lau. Distributed spanner construction with physical interference: Constant stretch and linear sparseness. *IEEE/ACM Transactions on Networking*, 25(4):2138–2151, 2017.

[119] Dongxiao Yu, Li Ning, Yifei Zou, Jiguo Yu, Xiuzhen Cheng, and Francis CM Lau. Distributed spanner construction with physical interference: constant stretch and linear sparseness. *IEEE/ACM Transactions on Networking*, 25(4):2138–2151, 2017.

[120] Dongxiao Yu, Yong Zhang, Yuyao Huang, Hai Jin, Jiguo Yu, and Qiang-Sheng Hua. Exact implementation of abstract mac layer via carrier sensing. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 1196–1204. IEEE, 2018.

[121] Dongxiao Yu, Yifei Zou, Minghui Xu, Yicheng Xu, Yong Zhang, Bei Gong, and Xiaoshuang Xing. Competitive age of information in dynamic iot networks. *IEEE Internet of Things Journal*, 2020.

[122] Zhaoyang Yu, XiaoGuang Liu, and Gang Wang. A survey of consensus and incentive mechanism in blockchain derived from p2p. In *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 1010–1015. IEEE, 2018.

[123] Kaiwen Zhang and Hans-Arno Jacobsen. Towards dependable, scalable, and pervasive distributed ledgers with blockchains. In *ICDCS*, pages 1337–1346, 2018.

[124] Yanchao Zhao, Jie Wu, and Sanglu Lu. Efficient sinr estimating with accuracy control in large scale cognitive radio networks. In *2011 IEEE 17th International Conference on Parallel and Distributed Systems*, pages 549–556. IEEE, 2011.

[125] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, 14(4):352–375, 2018.

[126] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang. An overview of blockchain technology: Architecture, consensus, and future trends. In *2017 IEEE International Congress on Big Data (BigData Congress)*, pages 557–564. IEEE, 2017.

Appendix A: Probabilistic Tools

A.1 Chernoff Bound

Given a set of independent binary random variables $X_1, X_2, \cdots, X_n$, let $X = \sum_1^n X_i$ and $\mu = \sum_1^n p_i$, where $X_i = 1$ with probability p_i. If $\mathbb{E}[\prod_{i\in S} X_i \leq \prod_{i\in S} q_i]$, where $S \subseteq \{0,1,\cdots,n\}$, then it holds for any $\delta > 0$ that

$$P_r[X \geq (1+\delta)\mu] \leq e^{-\frac{\delta^2\mu}{2(1+\delta/3b)}}.$$

If $\mathbb{E}[\prod_{i\in S} X_i \geq \prod_{i\in S} q_i]$, where $S \subseteq \{0,1,\cdots,n\}$, then for any $\delta \in (0,1]$, we have

$$P_r[X \leq (1-\delta)\mu] \leq e^{-\frac{\delta^2\mu}{2}}.$$

A.2 Azuma–Hoeffding Inequality

Suppose X_k $(k = 0,1,\cdots)$ is a martingale (or super-martingale) and $|X_k - X_{k-1}| \leq c_k$. Then for all positive integers N and $\delta > 0$, we have

$$P[X_N - X_0 > \delta] \leq e^{\frac{-\delta^2}{2\Sigma_{k=1}^N c_k^2}}.$$