Blockchain and Distributed Consensus: From Security Analysis to Novel Applications

Yang Xiao

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

Wenjing Lou, Chair
Y. Thomas Hou
Luiz A. Periera da Silva
Jeffrey H. Reed
Ning Zhang

May 6, 2022
Falls Church, Virginia

Keywords: Blockchain, distributed consensus, network security

Blockchain and Distributed Consensus: From Security Analysis to Novel Applications

Yang Xiao

(ABSTRACT)

Blockchain, the technology behind cryptocurrency, enables decentralized and distrustful parties to maintain a unique and consistent transaction history through consensus, without involving a central authority. The decentralization, transparency, and consensus-driven security promised by blockchain are unprecedented and can potentially enable a wide range of new applications that prevail in the decentralized zero-trust model. While blockchain represents a secure-by-design approach to building zero-trust applications, there still exist outstanding security bottlenecks that hinder the technology's wider adoption, represented by the following two challenges: (1) blockchain as a distributed networked system is multi-layered in nature which has complex security implications that are not yet fully understood or addressed; (2) when we use blockchain to construct new applications, especially those previously implemented in the centralized manner, there often lack effective paradigms to customize and augment blockchain's security offerings to realize domain-specific security goals. In this work, we provide answers to the above two challenges in two coordinated efforts.

In the first effort, we target the fundamental security issues caused by blockchain's multi-layered nature and the consumption of external data:

- Existing analyses on blockchain consensus security overlooked an important cross-layer factor—the heterogeneity of the P2P network's connectivity. We first provide a comprehensive review on notable blockchain consensus protocols and their security properties. Then we focus one class of consensus protocol—the popular Nakamoto consensus—for which we propose a new analytical model from the networking perspective that quantifies the impact of heterogeneous network connectivity on key consensus security metrics, providing insights on the actual "51% attack" threshold (safety) and mining revenue distribution (fairness).

- The external data truthfulness challenge is another fundamental challenge concerning the decentralized applications running on top of blockchain. The validity of external data is key to the system's operational security but is out of the jurisdiction of blockchain consensus. We propose DecenTruth, a system that combines a data mining technique called truth discovery and Byzantine fault-tolerant consensus to enable decentralized nodes to collectively extract truthful information from data submitted by untrusted external sources.

In the second effort, we harness the security offerings of blockchain's smart contract functionality along with external security tools to enable two domain-specific applications—data usage control and decentralized spectrum access system:

- First, we use blockchain to tackle a long-standing privacy challenge of data misuse. Individual data owners often lose control on how their data can be used once sharing the data with another party, epitomized by the Facebook-Cambridge Analytica data scan-

dal. We propose PrivacyGuard, a security platform that combines blockchain smart contract and hardware trusted execution environment (TEE) to enable individual data owner's fine-grained control over the usage (e.g., which operation, who can use on what condition/price) of their private data. A core technical innovation of PrivacyGuard is the TEE-based execution and result commitment protocol, which extends blockchain's zero-trust security to the off-chain physical domain.

- Second, we employ blockchain to address the potential security and performance issues facing dynamic spectrum sharing in the 5G or next-G wireless networks. The current spectrum access system (SAS) designated by the FCC follows a centralized server-client service model which is vulnerable to single-point failures of SAS service providers and also lacks an efficient, automated inter-SAS synchronization mechanism. In response, we propose a blockchain-based decentralized SAS architecture dubbed BD-SAS to provide SAS service efficiently to spectrum users and enable automated inter-SAS synchronization, without assuming trust on individual SAS service providers.

We hope this work can provide new insights into blockchain's fundamental security and applicability to new security domains.

Blockchain and Distributed Consensus: From Security Analysis to Novel Applications

Yang Xiao

(GENERAL AUDIENCE ABSTRACT)

Blockchain, the technology behind cryptocurrency, enables decentralized and distrustful parties to maintain a unique and consistent transaction history through consensus, without involving a central authority. The decentralization, transparency, and consensus-driven security promised by blockchain are unprecedented and can potentially enable zero-trust applications in a wide range of domains. While blockchain's secure-by-design vision is truly inspiring, there still remain outstanding security challenges that hinder the technology's wider adoption. They originate from the blockchain system's complex multi-layer nature and the lack of effective paradigms to customize blockchain for domain-specific applications. In this work, we provide answers to the above two challenges in two coordinated efforts.

In the first effort, we target the fundamental security issues caused by blockchain's multi-layered nature and the consumption of external data. We first provide a comprehensive review on existing notable consensus protocols and their security issues. Then we propose a new analytical model from a novel networking perspective that quantifies the impact of heterogeneous network connectivity on key consensus security metrics. Then we address the external data truthfulness challenge concerning the decentralized applications running on top of blockchain which consume the real-world data, by proposing DecenTruth, a system that combines data mining and consensus to allow decentralized blockchain nodes to collectively extract truthful information from untrusted external sources.

In the second effort, we harness the security offerings of blockchain's smart contract functionality along with external security tools to enable two domain-specific applications. First, eyeing on our society's data misuse challenge where data owners often lose control on how their data can be used once sharing the data with another party, we propose PrivacyGuard, a security platform that combines blockchain smart contract and hardware security tools to give individual data owner's fine-grained control over the usage over their private data. Second, targeting the lack of a fault-tolerant spectrum access system in the domain of wireless networking, we propose a blockchain-based decentralized spectrum access system dubbed BD-SAS to provide spectrum management service efficiently to users without assuming trust on individual SAS service providers.

We hope this work can provide new insights into blockchain's fundamental security and applicability to new security domains.

Dedication

To my parents.

Acknowledgments

First and foremost, I would like to express the deepest gratitude to my advisor Prof. Wenjing Lou. She is not only a wonderful researcher but also the most caring mentor I have met in my educational journey. I was fortunate to join Prof. Lou's group as a Ph.D. student in 2017. She introduced me to the exciting world of network and information security and provided me invaluable guidance and support. She taught me how to formulate research problems, come up with rigorous solutions, and generate outcomes of the highest quality. Working with Prof. Lou is the most precious experience of my academic life up to this day. Her vision and teaching also inspired me to pursue academic research as a life-long endeavor.

I would like to extend my gratitude to my committee members Prof. Thomas Hou, Prof. Luiz DaSilva, Prof. Jeffrey Reed, Prof. Ning Zhang for their insightful comments and suggestions on my research. I would also like to thank Prof. Ning Zhang for his help on the system security aspect of my research and his advice on my career development.

I would like to thank my colleagues at the Complex Network and Security Research (CNSR) laboratory at Virginia Tech, Wenhai Sun, Changlai Du, Yimin Chen, Ning Wang, Shanghao Shi, Yang Hu, Md Hasan Shahriar, Chaoyu Zhang, Hexuan Yu, Shaoyu Li, for the discussions.

Last but not least, I want to thank my parents, Weixing Yang and Deshan Xiao, for bringing me up and teaching me the utmost importance of perseverance and curiosity. It would be impossible for me to pursue my academic dream without their unconditional love.

Funding Acknowledgments

Yang Xiao's work was supported in part by the US National Science Foundation under grants CNS-1800650, CNS-1837519, and CNS-1916902, the Office of Naval Research under grant N00014-19-1-2621, the Virginia Commonwealth Cyber Initiative (CCI), and a gift from InterDigital.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Since Bitcoin's inception in late 2008, cryptocurrencies and the underlying blockchain technology have piqued great interest from the financial industry and society as a whole. Blockchain is widely cited as a fully decentralized system and a secure-by-design technology. The blockchain itself is a database that keeps track of all transactions occurred in the network and is replicated at every participating node. It essentially realizes a distributed ledger without relying on a central authority to bootstrap the trust among participants or to clear the transactions. At its core, a blockchain system is secured by public-key cryptography and distributed consensus, enabling mutually distrustful parties to transact with each other and curate a unified, consistent transaction history even some parties are malfunctioning or malicious. Because of this unprecedented harmonization of security and decentralization, blockchain has attracted huge interest from hobbyists, investors and researchers alike and spawned a plethora of new applications beyond the financial world. However, underneath the grandeur of blockchain technology are many unsolved challenges, in both system security and performance. There is an urgency to address these outstanding challenges related to the security of the blockchain system itself as well as the potential applications it enables.

First, blockchain is a decentralized ledger system and has a disparate security landscape compared to centralized databases. The secure-by-design property of blockchain stems from the security of blockchain's consensus core. Notably, classical analysis on blockchain consensus security has long been established idealistically with only trust assumptions on consen-

sus participants (i.e., miners), represented by the honest majority assumption in Bitcoin's Nakamoto consensus (also known as the 50% security threshold) [90, 151]. However, the classical analysis ignores the important factors from other system layers, including the heterogeneity in network connectivity and mining strategy, which can significantly impact the dynamics of the blockchain consensus process. Their inclusion in the analytical model will lead to a more practical view on the core consensus security.

Second, although the blockchain ledger is able to keep a validated and immutable record of system history, it only ensures the validity of transaction data that happened within the blockchain financial system; the validity of external data is out of check. In many application scenarios of blockchain, with smart contract being a prominent one, feeding on external data is necessary for the application to have a real-world impact. Current solutions [174, 244, 245] aim to build a trusted data feed channel that fetches data from external sources but still need to assume the sources themselves are trustworthy. This culminates into a fundamental issue which we call the data truthfulness challenge. To this end, there needs new mechanisms on top of the blockchain network to extract reliable information from untrustworthy external sources and identify those who feed false data.

Third, when it comes to applying the blockchain technology to novel applications with specific security goals, there often needs to integrate blockchain's security functions into domain-specific security goals to replicate the functionalities of the previously centralized version. This integration process is hardly smooth since blockchain as a tool cannot cover all security aspects, let alone its own security and performance challenges. On the one hand, the transparency of blockchain ledger often leads to data privacy concerns. This concern becomes profound when the new application runs on sensitive personally identifiable information. On the other hand, the decentralization of blockchain leads to performance bottlenecks, represented by the scalability of the number of participating nodes and the transaction

processing capacity. In commercial applications, the performance bottlenecks are often the major hindrance to blockchain's adoption.

In this work, we address the above challenges in two coordinated efforts: (1) analyzing and addressing the fundamental security issues of the blockchain system which originate from its multi-layer nature, and (2) designing novel applications that harmonize blockchain's security offerings with new security mechanisms to realize domain-specific security goals, while not introducing significant performance overhead. In specific, we make the following contributions in this work:

- Chapter 2 presents a review and qualitative analysis on the state-of-the-art blockchain consensus protocols. To facilitate the discussion of our analysis, we first introduce the key definitions and relevant results in the classic theory of fault tolerance which helps to lay the foundation for further discussion. We identify five core components in a blockchain, namely, block proposal, block validation, information propagation, block finalization, and incentive mechanism. A wide spectrum of established blockchain consensus protocols are then carefully reviewed accompanied by algorithmic abstractions and vulnerability analyses. The analyses and comparisons provide us new insights in the fundamental differences of various proposals in terms of their suitable application scenarios, key assumptions, expected fault tolerance, scalability, drawbacks and trade-offs.

- Chapter 3 provides a model-based analysis on the blockchain's core consensus security from a networking perspective. Observing the potential impact of the heterogeneity of blockchain's P2P network on the core consensus security, we introduce an analytical model that takes into account network connectivity and adversarial strategy. We are able to quantify the communication capability of nodes involved in a blockchain fork

race and estimate the adversary's mining revenue and its impact on security properties of the consensus protocol. Using Bitcoin's Nakamoto consensus as the target, under different adversarial mining cases, this model can evaluate a node's winning chance for a fork race and predict key security metrics, including long-term fork rate which reveals the actual "51% attack" threshold (safety) and the mining revenue distribution among all nodes (fairness). Our modeling and analysis provide a paradigm for assessing the security impact of cross-layer factors in a blockchain system.

- Chapter 4 addresses one of blockchain fundamental challenges—the external data truthfulness challenge, which is faced by every decentralized applications that run on top of a blockchain system. We introduce a new truth discovery (TD) model called decentralized truth discovery where a group of decentralized nodes work to extract reliable information on common data objects from untrustworthy multi-sourced inputs in a dynamic and adversarial environment. To tackle this problem, we propose a decentralized truth discovering data feed architecture called DecenTruth that enables blockchain nodes to agree on the same estimated truths with efficiency guarantee in the presence of Byzantine sources and peer nodes.

- Chapter 5 presents a novel application of blockchain to address an outstanding data privacy issue plaguing our digitized economy—individuals often lose control on how their private data can be used one sharing them with a third party, which was epitomized by the Facebook-Cambridge Analytica data scandal. In response, eyeing on the blockchain smart contract's potential of automatic rule enforcement, we propose PrivacyGuard, a platform that enables individual data owner's control over the access and usage of their private data. Smart contracts are used to specify data owner's data usage policy, i.e., which data consumers can use which data under what conditions and what computation to perform. To prevent exposing private data on the

publicly viewable blockchain and address the scalability problem of on-chain contract execution, PrivacyGuard incorporates a TEE-based off-chain contract execution engine along with a novel two-party commitment protocol to securely commit the off-chain execution result to the data consumer while fulfilling data owner compensation and usage recording on blockchain.

- Chapter 6 presents another novel application of blockchain to tackle the security and trust challenges in dynamic spectrum sharing, which is an emerging sub-area of wireless networks. We target the fundamental security of spectrum access system (SAS), a widely recognized framework for dynamic spectrum sharing in the 5G era. The current SAS designated by the FCC follows a centralized service model in that a spectrum user queries a commercial SAS server for spectrum access assignment. This model, however, faces several security and performance challenges, with respect to the single-point risk of SAS service provide and the efficiency of inter-SAS service synchronization. To fill this gap, we propose a blockchain-based decentralized SAS architecture dubbed BD-SAS to provide SAS service efficiently to spectrum users and enable automated inter-SAS synchronization, without assuming trust on individual SAS service providers.

- Chapter 7 summarizes this dissertation and identifies three future directions: (1) extending the modeling analysis in Chapter 3 to more generalized network measurement cases, (2) designing robust consensus mechanisms that are resilient to dynamic and unstable network conditions, and (3) designing novel decentralized architectures and applications in the zero trust scenario.

Chapter 2

A Survey of Distributed Consensus Protocols for Blockchain Networks

(Copyright notice[1])

2.1 Introduction

The security properties promised by blockchain is unprecedented and truly inspiring. Among the many technical components that a blockchain system is composed of, the distributed consensus protocol is the key technology that enables blockchain's decentralization, or more specifically, that ensures all participants agree on a unified transaction ledger without the help of a central authority. The distributed consensus protocol specifies message passing and local decision making at each node. Various design choices in the consensus protocol can greatly impact a blockchain system's performance, including its transaction capacity, scalablity, and fault tolerance.

The Nakamoto consensus protocol [151] is the protocol implemented in the Bitcoin network. With the help of this consensus protocol, Bitcoin became the first digital currency sys-

[1]This chapter previously appeared as a part of a journal paper published in IEEE Communications Surveys & Tutorials. ©2020 IEEE. Reprinted, with permission, from Yang Xiao, Ning Zhang, Wenjing Lou, and Y. Thomas Hou, "A Survey of Distributed Consensus Protocols for Blockchain Networks," in IEEE Communications Surveys & Tutorials, 22(2):1432–1465, 2020 [233].

tem to resist double-spending attacks in a decentralized peer-to-peer network of little trust. As the Bitcoin network continues to grow, Nakamoto consensus has encountered several performance bottlenecks and sustainability problems. Researchers in blockchain communities have raised the following concerns on Nakamoto consensus and particularly its proof-of-work (PoW) mining mechanism: 1) unsustainable energy consumption, 2) low transaction capacity and poor scalability, 3) long-term security concerns as mining rewards diminish. For instance, the Bitcoin network currently consists of roughly ten thousand nodes [26], while the maximum transaction capacity of Bitcoin is 7 transactions per second (TPS) and can be increased to at most 25 TPS by tuning protocol parameters without jeopardizing consensus safety [61]. In contrast, the Visa network consists of millions of participants and was able to handle more than 24,000 TPS in 2012 [213]. A single Bitcoin transaction (May 2022) consumes the equivalent amount of electricity that would power 72 average U.S. households for one day [71].

In response to the above performance limitations of PoW mining, blockchain researchers have been investigating new block proposing mechanisms such as proof of stake (PoS), proof of authority (PoA), and proof of elapsed time (PoET) which do not require computation-intensive mining, thus effectively reducing energy consumption. In some cases, cryptographic methods can be used to establish trust among nodes, enabling the use of more coordinated block proposing schemes such as round-robin and committee-based block generation. Appropriate incentives that will continue to encourage honest participation in the blockchain network is another key component of consensus protocol. Therefore, alternative block proposing schemes are often accompanied by a new incentive mechanism that promotes participation fairness and increases overall system sustainability. Popular blockchain consensus protocols encompassing these ideas include Peercoin [111], Bitcoin-NG [83], Ourosboros (Cardano) [110], Snow White [63], and EOSIO [27], POA Network [169], etc.

Besides block proposing and incentive mechanisms, researchers have been seeking solutions from the prior wisdom—primarily classical Byzantine fault tolerant (BFT) consensus and secure multi-party computation (MPC)—for efficient block finalization methods. For example, the state machine replication (SMR) based BFT consensus algorithms have great potential in permissioned blockchain networks operated with static and revealed identities, of which Tendermint [117], Algorand [96], Casper FFG [39], and Hyperledger Fabric [7] are well-known use cases. Moreover, asynchronous consensus protocols such as HoneyBadgerBFT [148] and BEAT [72] were proposed to provide robust block finalization under severe network conditions with uncertain message delays.

With more blockchain consensus mechanisms being proposed, there is a pressing need to analyze and compare them in a formal and cohesive manner. In this chapter we present a comprehensive review and analysis of the state-of-the-art blockchain consensus protocols and their development history, with a special focus on their performance, fault tolerance, and security implications. Our information sources include academic papers, consensus protocol white papers, official documentation and statistics websites of cryptocurrencies. Specifically, this work features the following contributions:

- Providing a background of fault-tolerant distributed consensus research, including partially synchronous and asynchronous BFT protocols that are applicable to blockchain consensus;
- Reviewing a broad array of blockchain consensus protocols with a proposed five-component framework and analyzing their design philosophy and security issues;
- Identifying four classes of proof of stake (PoS) based consensus protocols and providing algorithmic abstractions for them;
- Comparing all mentioned consensus protocols with respect to the five-component

framework, fault tolerance, and transaction processing capability.

- Providing a succinct tutorial on blockchain consensus protocol design with respect to the security-decentralization-scalability trilemma.

2.2 Fault-Tolerant Distributed Consensus

The fault-tolerant (FT) distributed consensus problem has been extensively studied in distributed systems since the late 1970s and recently gained popularity in the blockchain community, especially for permissioned blockchains where every consensus participant reveals its identity. Generally, consensus in a distributed system represents a state that all participants agree on the same data values. Depending on the medium for message exchange, distributed systems are classified into two types: message passing and shared memory [11]. In this section we are interested in message passing systems because of their resemblance to contemporary blockchain systems, wherein distributed consensus on a single network history is reached through peer-to-peer communication. We will use the terms process/node/server interchangeably, as they all refer to an individual participant of distributed consensus.

2.2.1 System Model

We consider a distributed system that consists of N independent processes. Each process p_i begins with an individual initial value x_i and communicates with others to update this value. Each local value can be used for a certain task, such as computation or just storage. If the processes are required to perform the same task, consensus on a single value is required before they proceed to the task.

Process Failure. A process suffers a crash failure if it abruptly stops working without resuming. The common causes of a crash failure include power shutdown, software errors, and DoS attacks. A Byzantine failure, however, is much severer in that the process can act arbitrarily while appearing normal. It can send contradicting messages to other processes in hope of sabotaging the consensus. "Byzantine" was coined by Lamport et al. [121] in 1982 when describing the Byzantine Generals Problem, an allegorical case for single-value consensus among distributed processes. The common cause of a Byzantine failure is adversarial influence, such as malware injection and physical device capture. Multiple Byzantine processes may collude to deal more damage.

Network Synchrony. Network synchrony defines the level of coordination among all processes. Three levels of synchrony, namely synchronous, partially synchronous, and asynchronous, are often assumed in the literature [10, 230]:

- In a synchronous network, operations of processes are coordinated in rounds with clear time constraints. In each round, all processes perform the same type of operations. This can be achieved by a centralized clock synchronization service and good network connectivity. Practically, a network is considered synchronous if message delivery is guaranteed within a fixed delay Δ, for which the network is also called Δ-synchronous.
- In a partially synchronous network, operations of processes are loosely coordinated in a way that message delivery is guaranteed but with uncertain amount of delays. Within the scope of partial synchrony, weak synchrony requires message delay not grow faster than the elapsing time indefinitely [49], while eventual synchrony ensures Δ-synchrony only after some unknown instant [74]. In either case, operations of the networked processes can still follow that of a synchronous network if the time horizon is long enough.

- In an asynchronous network, operations of processes are hardly coordinated. There is no delay guarantee on a message except for its eventual delivery. And the coordination of processes (if there is any) is solely driven by the message delivery events. This is often caused by the absence of clock synchronization (thus no notion of shared time) or the dominance of a mighty adversary over all communication channels.

It has been shown by Fischer, Lynch, and Paterson [85] that under the asynchronous case, consensus cannot be guaranteed with even a single crash failure. This is commonly known as the FLP impossibility, for the authors' namesake. However, this impossibility can be practically circumvented using randomized decision making and a relaxed termination property, as we will show in §2.2.5.

2.2.2 Byzantine Fault Tolerant Consensus

We call a consensus protocol crash fault tolerant (CFT) or Byzantine fault tolerant (BFT) if it can tolerate a certain amount of crash or Byzantine process failures while keeping normal functioning. Because of the inclusive relationship between a crash failure and Byzantine failure, a BFT consensus protocol is naturally CFT. BFT consensus is defined by the following four requirements [10, 74, 230]:

- Termination: Every non-faulty process decides an output.
- Agreement: Every non-faulty process eventually decides the same output $\hat{y}$.
- Validity: If every process begins with the same input $\hat{x}$, then $\hat{y} = \hat{x}$.
- Integrity: Every non-faulty process' decision and the consensus value $\hat{y}$ must have been proposed by some non-faulty process.

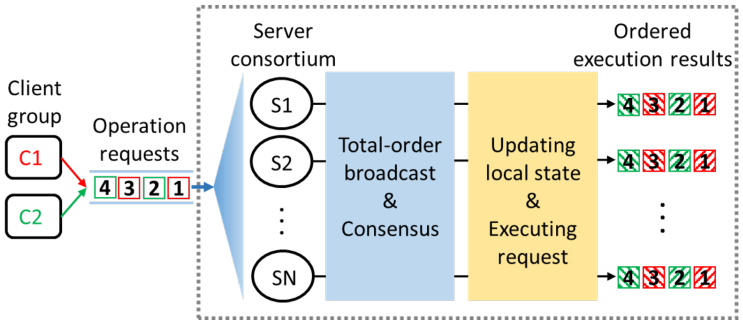


Figure 2.1: A high-level illustration of SMR-based distributed computing that serves four operation requests from two clients.

The four requirements provide a general target for distributed consensus protocols. For any consensus protocol to attain these BFT requirements, the underlying distributed network should satisfy the following condition: $N \geq 3f + 1$ where f is the number of Byzantine processes. This fundamental result was first proved by Pease et al. [168] in 1980 and later adapted to the BFT consensus framework. The proof involves induction from $N = 3$ and partitioning all processes into three equal-sized groups, with one containing the faulty ones. Interested readers are referred to [10, 168] for detailed proofs.

2.2.3 Consensus in Distributed Computing

Consensus in distributed computing is a more sophisticated realization of the aforementioned distributed system. In a typical distributed computing system, one or more clients issue operation requests to the server consortium, which provides timely and correct computing service in response to the requests despite some of servers may fail. Here the correctness requirement is two-fold: correct execution results for all requests and correct ordering of them. According to Alpern and Schneider's work on liveness definition [4] in 1985, the correctness of consensus can be formulated into two requirements: safety—every server correctly executes the same sequence of requests, and liveness—all requests should be served.

To fulfill these requirements even in the presence of faulty servers, server replication schemes especially state machine replication (SMR) are often heralded as the de facto solution. SMR, originated from Lamport's early works on clock synchronization in distributed systems [118, 119], was formally presented by Schneider [185] in 1990. Setting in the client-server framework, SMR sets the following requirements:

- All servers start with the same initial state;
- Total-order broadcast: All servers receive the same sequence of requests as how they were generated from clients;
- All servers receiving the same request shall output the same execution result and end up in the same state.

Total-order broadcast is also known as atomic broadcast (ABC) [60], which is in contrast to the reliable broadcast (RBC) [32] primitive. The latter only requires all servers receive the same requests without enforcing the order. It is shown in [50, 68] that atomic broadcast and distributed consensus are equivalent problems.

A high-level diagram of SMR-based distributed computing is illustrated in Fig. 2.1. The N-server consortium accepts client requests and servers confirm each other's state before reaching consensus and executing requests. In many cases, especially randomized consensus protocols, there can be an alternating procedure of total-order broadcast and local state update until a certain consensus target is met, before moving on to execution. In practice, SMR is often implemented in a leader-based fashion. A primary server (say S1 in Fig. 2.1) receives client requests and starts the broadcast procedure so that the other $N-1$ replica servers receive the same requests and update their local states to that of the primary.

In the rest of this section we summarize several well-known consensus protocols (some

are based on SMR) designed under different network synchrony assumptions.

2.2.4 Consensus Protocols for Partially Synchronous Network

The ground-breaking work by Dwork, Lynch, and Stockmeyer [74] in 1988 laid the theoretical foundation of partially synchronous consensus. By dissecting the consensus objective into termination and safey, the authors were able to formally prove the feasibility of four consensus goals, including CFT, omission-tolerance, BFT, unauthenticated BFT, under the Δ-synchrony/eventual synchrony condition. Notably, this work has inspired numerous proposals for partially synchronous consensus schemes, including the later known PBFT.

The DLS Protocol. The same paper [74] also proposes a prototype consensus protocol (called DLS for authors' namesake) featuring a broadcast primitive for each consensus cycle. Specifically, the broadcast primitive is started by an arbitrary process p and consists of two initial rounds and subsequent iterative rounds. Through message exchanges in each round, the iterative procedure eventually drives the processes to reach agreement on a common value (either the one proposed by p or a default value). At message complexity $O(N^2)$ (N is the number of processes), the broadcast primitive essentially enables the DLS protocol to tolerate f Byzantine processes if $N \geq 3f + 1$. The cryptocurrency Tendermint uses an adapted version of DLS for block finalization.

Viewstamped Replication (VR). Proposed by Oki and Liskov [161] in 1988, viewstamped replication is a server replication scheme for handling server crashes. It was later extended into a consensus protocol by Liskov and Cowling [133] in 2012, which we will refer to as VR. VR is a SMR scheme designed in the client-server framework and consists of three subprotocols: (1) Normal-operation, (2) View-change, and (3) Recovery. The primary server

receives a client request and starts the normal operation, as is shown in Fig. 2.2a. In the case of a crash failure of the primary, the View-change protocol is triggered at every replica per the timeout of the Prepare message. They broadcast View-change messages to each other and count the receptions. After receiving View-change messages from more than half of the replicas, the next-in-line replica becomes the new primary and informs the others to resume the normal operation. The Recovery protocol is used by any server to recover from a crash. VR can tolerate f crashed replicas if the network population $N \geq 2f + 1$. However it does not tolerate any Byzantine failure, because the replicas simply follow the instructions from the primary without mutual state confirmation nor communication with the client. On the up side, this makes VR efficient, with $O(N)$ message complexity.

Paxos. Paxos is a SMR scheme proposed by Lamport [120] in 1989 that imitates the ancient Paxos part-time parliament and later elaborated in 2001 [123]. It was designed specifically for fault tolerant consensus while bearing many similarities to VR. Paxos classifies nodes into three roles: proposers, acceptors, and learners. A proposer suggests a value in the beginning and the system goal is to make acceptors agree on a single value, and learners learn this value from acceptors. In the client-server scenario depicted in Fig. 2.2b, the client is the learner, the primary is the proposer, and the replicas are acceptors. After updating to the same state, all servers execute the request and send it to the client who then chooses the majority result. When the proposer suffers a crash failure, the acceptors elects a new leader through a similar propose-accept procedure. Akin to VR, Paxos can tolerate f crashed acceptors when $N \geq 2f + 1$, but no Byzantine failures. Because of the mutual messaging during the accept phase, the message complexity of Paxos is $O(N^2)$.

Embarking from its original design, Paxos has grown into a family of consensus protocols, including multi-Paxos, cheap-Paxos, and fast-Paxos, each features a specific goal. Raft, a

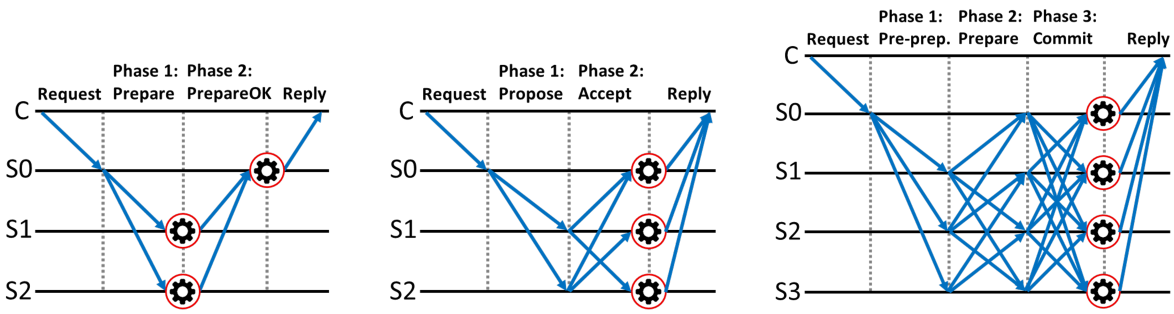
SMR consensus protocol developed by Ongaro and Ousterhout [162] in 2014 and popular in the blockchain community, is based off Paxos but with a more understandable design.

Practical Byzantine Fault Tolerance (PBFT). Developed by Castro and Liskov [49] in 1999, PBFT is the first SMR-based BFT consensus protocol that has gained wide recognition for practicality. It has become almost synonymous to BFT consensus in the blockchain community. PBFT originated from VR and took inspiration from Paxos. PBFT consists of three sub-protocols: (1) Normal-operation, (2) Checkpoint, and (3) View-change.

Algorithm 1: PBFT (Normal-operation protocol)

/* Request */
1 Client sends an operation request to the primary;
/* Phase 1: Pre-prepare */
2 The primary relays this request to replicas via Pre-prepare messages;
3 Replicas record the request and update local states;
/* Phase 2: Prepare */
4 Replicas send Prepare messages to all servers (replicas and the primary);
5 Once receiving $\geq 2f + 1$ Prepare messages, a server updates local state and is ready to commit;
/* Phase 3: Commit */
6 Servers send Commit messages to each other;
7 Once receiving $\geq 2f + 1$ Commit messages, a server starts to execute the client request and then updates local state;
/* Reply */
8 Every server replies its result to the client.

The Normal-operation protocol is shown in Algorithm 8 and Fig. 2.2c. Ideally, all results replied to the client should be the same; otherwise the client chooses the majority result. The Checkpoint protocol serves as a logging tool that keeps a sliding window (of which the lower bound is the stable checkpoint) to track active operation requests. The latest stable checkpoint is used for safely discarding older requests in the operation log and facilitating the view change protocol. In the case of a primary failure, the View-change protocol is triggered at every replica that detects the timeout of the primary's message. They oust the incumbent primary and broadcast view-change messages to each other and count receptions. After receiving View-change messages from $2f$ peers, the next-in-line replica becomes the



(a) VR (b) Paxos (c) PBFT

Figure 2.2: Messaging diagram during the normal operation of three SMR protocols.

new primary and informs the rest to resume the normal operation.

The message complexity of PBFT normal operation is $O(N^2)$ because of the mutual messaging in Prepare and Commit phase. As for the fault tolerance, since a server needs to receive more than $2f+1$ Prepare (Commit) messages in Prepare (Commit) phase before proceeding to the next action, there will be at least $2f+1$ (as $N \geq 3f+1$) honest servers in the same state after Commit phase and producing the same result; the f Byzantine servers are not able to sway the majority consensus. Therefore PBFT can tolerate f Byzantine replicas when the server population $N \geq 3f+1$, which is in accordance to the fundamental 1/3 BFT threshold. Interested readers are referred to [49, 230] for detailed proofs.

PBFT has inspired numerous BFT consensus protocols with enhanced security and performance. Well-known proposals include Quorum/Update (QU) [1], Hybrid Quorum (HQ) [59], Zyzzyva (using speculative execution) [115], FaB [140], Spinning [210], Robust BFT SMR [55], Aliph [101], BFT-SMaRt [23], and Hotstuff [237]. Interested readers are referred to Bessoni's tutorial [24] for an overview of these protocols.

A comparison of communication paradigm between VR, Paxos, and PBFT is shown in Figure 2.2. C is the client. S0 is the primary server (leader) who receives requests from the client and starts the consensus. S1+ are replica servers. Every server updates local state

after receiving a message. Circled gear icon represents execution. VR and Paxos can tolerate one crash failure when $N = 3$. PBFT can tolerate one Byzantine failure when $N = 4$.

2.2.5 Consensus Protocols for Asynchronous Network

For distributed systems that are predominantly built upon wired communication and reliable transport-layer protocols, partial synchrony is a practical assumption. However in scenarios such as mobile ad hoc network (MANET) and delay tolerant networks (DTN), the network is considered of near-to-none synchrony. As is proved by the FLP impossibility [85], consensus can not be guaranteed in a fully asynchronous network with even one crash failure. Moreover, unreliable communication links have an equivalent effect of a Byzantine scheduler. Nonetheless, this impossibility result can be practically circumvented by two primitives: probabilistic termination and randomization.

First of all, according to [31] the termination property presented in §2.2 can be subdivided into two classes:

- Deterministic Termination: Every non-faulty process decides an output by round r, a predetermined parameter.
- Probabilistic Termination: The probability that a non-faulty process is undecided after r rounds approaches zero as r grows to infinity.

For synchronous or partially synchronous networks where message delay and round period are bounded, protocols like PBFT can exploit a timeout mechanism to detect anomaly of the primary, which makes deterministic termination an achievable goal. For asynchronous networks where messages delivery has no timing guarantee, the consensus process can only be driven by the message delivery events themselves. This limitation demands probabilistic

termination. To realize probabilistic termination, randomization (simultaneously proposed by Ben-Or [18] and Rabin [175] in 1983) can be instantiated in the consensus protocol. The basic idea is that a process makes a random choice when there are not enough trusted messages received for making a final decision.

Next we introduce four primitives/protocols that aim to solve asynchronous BFT consensus. Though in different contexts, they all feature probabilistic termination and make use of randomization.

Bracha's RBC and Asynchronous Consensus Protocol. Bracha et al. [30] proposed the pioneering reliable broadcast (RBC) primitive and an asynchronous consensus protocol in 1984 to solve the Byzantine Generals Problem [121], in which all non-faulty processes should eventually make the same binary decision. Bracha's RBC guarantees that non-faulty processes will never accept contradicting messages from any process and forces the faulty ones to output either nothing (mimicking the crash failure) or the correct value. Bracha's asynchronous consensus protocol, adapted from Ben-Or's 1983 work [18], runs by phases and each phase contains three RBC rounds for inter-process value exchange. We show the round-3 of each phase, which contains the randomization step. After receiving at least $N - f$ value messages (f is the presumed Byzantine population), a process P_i does: (1) If receiving a value v from more than $2f$ peers, decide v; (2) else if receiving a value v from more than f peers, hold v as proposal value and go to the next phase; (3) else, toss a coin (1/2 chance for 0 or 1) for the proposal value and go to the next phase.

When enough phases pass, the executions of step (2) and step (3) of RBC round-3 at all non-faulty processes will gradually filter out the influence of contradicting messages and eventually make the correct decision via step 1. Note this convergence only happens if $N \geq 3f+1$, which is the fundamental bound of BFT consensus. In terms of performance, the

message complexity of RBC is $O(N^2)$ in each round and the expected number of rounds to reach consensus is $O(2^N)$ if $f = O(N)$, which gives a total message complexity of $O(N^2 2^N)$. If $f = O(\sqrt{N})$ (the benign case), it is shown in [18, 30] that the expected number of rounds to reach consensus for the randomized protocol is a constant, yielding a total message complexity of $O(N^2)$.

Ben-Or's ACS Protocol for MPC. Agreement on a common subset (ACS) was used by Ben-Or et al. [19] in 1994 as a consensus primitive for secure and efficient multi-party computation (MPC) under asynchronous setting. In a network of N players, each player holds a private input x_i that was acquired secretly. The goal of MPC is to let the players collectively compute a function $\mathcal{F}(x_1, ..., x_N)$ and obtain the same result. Assuming f players can be faulty, the ACS primitive requires the players to agree on a common subset $ComSubset$ of at least $N - f$ honest inputs, which are then used for computing $\mathcal{F}(\cdot)$. Ben-Or's ACS protocol builds on two primitives: RBC and binary asynchronous Byzantine agreement (ABA) which allows players to agree on the value of a single bit. Bracha's RBC [30] and Canetti et al.'s Fast ABA [46] are suggested respectively in [19] and used as black-boxes. Algorithm 14 shows the ACS protocol at each player. In the end, there will be at least $N - f$ completed ABA instances with output 1, yielding a $\mathcal{F}$-computable $ComSubset$.

Because both Bracha's RBC and Canetti's ABA can tolerate f Byzantine players when $N \geq 3f + 1$, the same fault tolerance result is inherited by Ben-Or's ACS protocol. For complexity analysis, Bracha's RBC (in the benign case) and Canetti's ABA have message complexity of $O(N^2)$ and $O(N^3)$ [46] respectively, and all ABA instances end in constant rounds. As a result, Ben-Or's ACS protocol has a bit-denominated communication complexity of $O(mN^2 + N^3)$ at each player, where m is the maximum bit-size of any input.

As we will see next, ACS can be conveniently adapted to asynchronous BFT consensus

Algorithm 2: Ben-Or's ACS protocol (at player P_i)

```
/* Phase 1: Reliable Broadcast                                   */
Start RBC_i to propose my input x_i to the network;
Participate in other RBC instances;
/* Phase 2: Asynchronous BA                                      */
while round ≤ MaxRound do
    if receiving x_j from RBC_j then
        Join ABA_j with input 1;
    end
    if completion of N − f ABA instances then
        Join other BA instances with input 0;
    end
    if completion of all N ABA instances then
        ComSubset = {x_k | ABA_k outputs 1};
        return ComSubset;
    end
end
```

for blockchain systems, by substituting inputs with transaction sets.

HoneyBadgerBFT. Proposed by Miller et al. [148] in 2016, HoneyBadgerBFT is the first asynchronous BFT consensus protocol specifically designed for blockchain. It essentially realizes atomic broadcast: N players with different sets of transactions work to agree on a common set of sorted transactions that will be included in a block.

Though using the multi-value Byzantine agreement primitive (MVBA) from Cachin et al. [42] as the benchmark, HoneyBadgerBFT actually follows Ben-Or's ACS construction [19] for better communication efficiency. HoneyBadgerBFT's ACS cherry-picks the design of its sub-components: Cachin and Tessaro's erasure-coded RBC [40] and Mostéfaouil et al.'s common-coin based ABA [149]. They together incur $O(mN + N^2 \log N)$ communication complexity at each player. To prevent an adversary from censoring particular transactions, threshold public key encryption (TPKE) [13] is used before ACS so that consensus is performed on ciphertexts. In the decryption phase when a player receives enough shares from peers (generated by TPKE.DecShare) that exceed a threshold, it proceeds to the actual de-

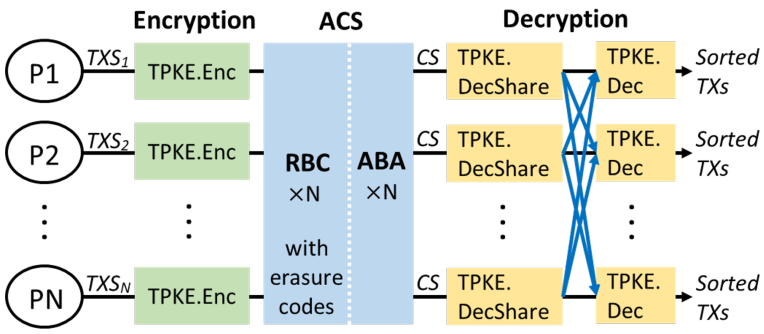


Figure 2.3: HoneyBadgerBFT workflow.

cryption task (TPKE.Dec) and sorts the transactions. The communication complexity of the decryption process is $O(N^2)$. Fig. 2.3 illustrates the workflow of these components for one block cycle. TXS_i represents the set of transactions proposed by player P_i. CS represents the common subset output of ACS, comprising of at least $N - f$ encrypted transactions. The decryption process outputs sorted transactions that will be finalized in the block.

HoneyBadgerBFT processes transactions in batches. Let B be the predefined batch size, denoting the maximum number of transactions that a block may enclose. For every block cycle, each player proposes a set of B/N transactions, which are randomly chosen from recorded transactions. This is to ensure transaction sets proposed by different players are mostly disjoint so as to maximize blockchain throughput. Assuming the average bit-size of a transaction set $m := \frac{|t|B}{N} \gg N$ where $|t|$ is the average transaction bit-size. Then the protocol's communication overhead will be dominated by the RBC, yielding overall communication complexity of $O(|t|B)$ at one player, or $O(|t|N)$ for one transaction.

Compared to popular partially synchronous consensus protocols such as PBFT, HoneyBadgerBFT has a higher cryptography overhead but features two advantages. First, as an asynchronous protocol HoneyBadgerBFT does not rely on a timeout mechanism for detecting malfunctioning players. This makes HoneyBadgerBFT less sensitive to unpredictable network delays that might stall consensus. Second, HoneyBadgerBFT does not need a leader

rotation scheme. In PBFT every round of consensus is started by a leader (the primary), while in HoneyBadgerBFT every node starts its own broadcast and Byzantine agreement instance for proposed transactions; the concurrent execution of these instances effectively saves the need of a leader. As a result, the bandwidth of any individual leader will not become the bottleneck of overall network's capacity. Currently the blockchain initiative POA Network [169] is considering to adopt HoneyBadgerBFT.

On the other hand, due to HoneyBadgerBFT's asynchronous design philosophy that consensus progress is driven by message deliveries, transaction confirmation latency is externally influenced and uncontrollable. This leads HoneyBadgerBFT to overly emphasize high transaction throughput and decentralization. In various applications such as industrial control and supply chain management, low transaction latency is often times a more important metric than throughput.

In response to the said inflexibility of HoneyBadgerBFT, Duan et al. [72] proposed BEAT in 2018, which is a collection of five asynchronous BFT protocols based off HoneyBadgerBFT but with carefully picked components that are optimized for different objectives. Among the five constituent protocols, the baseline BEAT0 uses a more efficient threshold encryption scheme [190] and outperforms HoneyBadgerBFT in throughput, latency and access overhead. BEAT1 and BEAT2 adopt a more efficient broadcast scheme, Bracha's RBC [31], and are optimized for transaction latency. BEAT3 is optimized for throughput and storage and bandwidth saving while BEAT4 further reduces the bandwidth usage for clients that read particular stored transactions. Interested readers are referred to the BEAT paper [72] for detailed discussion on the authors' design choices.

2.2.6 Blockchain Compatibility of Classical BFT-SMR Protocols

In a blockchain network, every consensus participant can validate transactions and propose new blocks. For BFT-SMR consensus protocols that rely on a dedicated primary server to receive client requests and start the consensus, the following adaptation is needed: allowing all servers to act as a primary to propose transactions/blocks and reaching consensus on the finality of multiple transactions/blocks concurrently. For example, Casper FFG [39], a BFT-style blockchain protocol, allows every eligible participant to propose a block during a checkpoint cycle. The network finalizes only one block out of multiple proposed blocks for each checkpoint.

For blockchain networks with a complex application layer such as smart contract, transaction execution often incurs significant computation. While in popular BFT-SMR schemes such as PBFT execution is integrated into the consensus process. An early work by Yin et al. [236] presents an alternative BFT-SMR framework that separates consensus (i.e. agreement on execution order) from execution, as the latter conveniently requires only an honest majority of execution nodes instead of an honest two-thirds of consensus nodes required by the former. This separation scheme is adopted by Zyzzyva [115] and Tendermint [117] wherein a small group of nodes are dedicated to the consensus task. Hyperledger Fabric [7] further separates the consensus task into ordering and validation services for better modularity.

From the performance perspective, BFT protocols are notorious for their limited scalability in network size. Epitomized by PBFT, the message complexity of partially synchronous BFT protocols grows quadratically with the network size N. This means that given a fixed network bandwidth at each node, a growing network size leads to exploding communication overhead and diminishing transaction capacity. According to the performance evaluation in [148], PBFT achieves a maximum throughput of 16,000 TPS when $N = 8$; this figure

drops to around 3,000 when $N = 64$. On the other hand, for asynchronous protocols like HoneyBadgerBFT where erasure coding and threshold encryption are used to reduce communication complexity and enhance security, the extensive use of cryptography also brings non-negligible computation overhead, adding to local processing delays. On the bright side, a typical BFT protocol achieves deterministic finality, which is also known as forward security [67] in that a settled transaction will never be altered. As we will discuss in §2.4, this allows BFT protocols to take advantage of shorter block intervals and attain high transaction throughput.

Other blockchain compatibility challenges for BFT-SMR protocols include: 1) allowing nodes to join and leave flexibly without interrupting consensus while countering Sybil attacks; 2) adapting to real-world peer-to-peer networks that are sparsely connected. In later sections we will revisit these issues for blockchain protocols that incorporate BFT consensus.

2.3 A Modular View of Blockchain Consensus Protocol

Compared to traditional distributed computing with a clear client-server model, a blockchain network allows every participant to be both a client (to issue transactions) and a server (to validate and finalize transactions). The underlying ledger data structure, the blockchain, is the consensus target and consists of chronologically ordered and hash-chained blocks. Each block contains a bundle of valid transactions and transactions across the blockchain should be consistent with each other (i.e. no double-/over-spending nor appropriation). Meanwhile, a blockchain system is often associated with a financial application and bears the responsibility of transaction processing and clearing. As a result, the responsibility of a blockchain consensus protocol is further-reaching than traditional distributed consensus protocols. In this section we provide a background of the blockchain network and data structure, intro-

Table 2.1: A Comparison of Permissionless and Permissioned Blockchain.

	Permissionless blockchain	Permissioned blockchain
Governance	Public	Private / Consortium
Participation	Free join and leave	Authorized
Node identity	Pseudonymous	Revealed
Transparency	Open	Closed / Open
Network size	Large (thousands or more)	Small (tens∼hundreds)
Network connectivity	Low	High (oft. fully-connected)
Network synchrony	Asynchronous / partially synchronous	Partially synchronous / synchronous
Transaction capacity	Low (oft. sub-ten∼tens TPS)	High (oft. thousands TPS)
Application examples	Cryptocurrency, smart contract, public record, DApp	Inter-bank clearing, business contract, supply chain

duce the blockchain consensus goal adapted from the BFT consensus paradigm and the five-component framework that we use to analyze blockchain consensus protocols.

2.3.1 Blockchain Infrastructure

Network. The foundational infrastructure of blockchain, as is adopted by most public cryptocurrencies and distributed ledger systems, is a peer-to-peer overlay network on top of the Internet. Every node (or peer) in the network operates autonomously with respect to the same set of rules that cover peering protocol, consensus protocol, transaction processing, ledger management, and in some cases a wire protocol for transport-layer communication [77, 156].

Depending on the control of network participation, blockchain networks generally fall into two categories: permissionless and permissioned.

- A permissionless blockchain allows for free join and leave without any authorization, as long as the node holds a valid pseudonym (account address) and is able to send, receive, and validate transactions and blocks by common rules. Permissionless blockchain is

also known as public blockchain for that there is usually one such blockchain network instance on a global scale which is subject to public governance. Specifically, anyone can participate in blockchain consensus, though one's voting power is typically proportional to its possession of network resources, such as computation power, token wealth, storage space, etc. The operational environment of permissionless blockchain is often assumed to be zero-trust, which often cautions the community against increasing transaction processing capacity or using more efficient consensus schemes [230].

- A permissioned blockchain requires participants to be authorized first and then participate in network operation with revealed identity. The network governance and consensus body can be either the subsidiaries of a single private entity or a consortium of entities [41]. Compared to permissionless blockchain, the identity-revealing requirement and more effective network governance of permissioned blockchain make it ideal for internal or multi-party business applications. Meanwhile, the limited size of a permissioned blockchain's consensus body allows for the deployment of more efficient consensus protocols that achieve higher transaction capacity [228, 230].

Table 2.1 summarizes the major differences between permissionless and permissioned blockchain in nine aspects.

Beneath the blockchain peer-to-peer network lies the basic infrastructure of the Internet. Thanks to the transport layer protocols (especially the retransmission mechanism), message delivery is considered guaranteed, while the message delay may vary but most likely will not grow longer as time elapses (weak synchrony) or remains within a certain bound (Δ-synchrony). Therefore we often consider a practical blockchain network partially synchronous, just like most distributed networks overlaying on the Internet. This allows the consensus protocol to take advantage of the timing services of the Internet. For example,

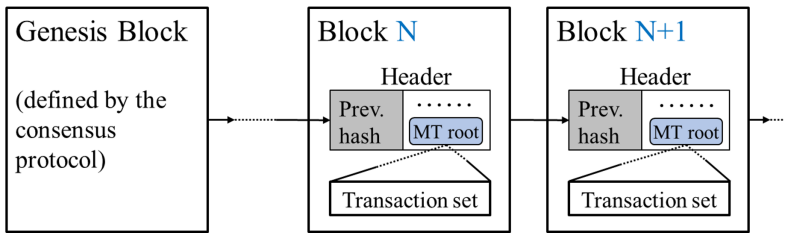


Figure 2.4: Blockchain data structure. Blocks are sequentially chained together via hash pointers. The Merkle tree root (MT root) is a digest of all transactions included in a block.

in Bitcoin the partial synchrony assumption is echoed by its usage of local timestamps for loose chronological ordering, showing time consciousness. For the blockchain networks that reside on an ad hoc infrastructure not based on the Internet, the message transmission is subject to unexpected network delays, which gives rise to asynchronous consensus protocols such as HoneyBadgerBFT, as we discussed in Section 2.2.

Transaction. A blockchain transaction can be regarded as a public static data record showing the token value redistribution between sender and receiver [217]. Take Bitcoin as an example, a transaction transfers token ownership from the sender account to the receiver account(s). It specifies a list of inputs and a list of outputs, with each input claiming a previous unspent transaction output (UTXO) that belongs to the sender, who needs to attach its signature to the inputs to justify the claim. Each output specifies how many tokens go to which receiver and the total token value of the outputs is equal to the UTXOs claimed by the inputs. Therefore, we can always recover the ownership records of any specific token by tracing back the signatures along the chain of transactions. The token balance of an account equals to the summed UTXOs that belong to the account.

Blockchain Data Structure. Blockchain is the underlying data structure for transaction ledger keeping. It is also the consensus target of the network. The basic structure of blockchain is illustrated in Fig. 2.4. Every block encloses a set of transactions that should

Table 2.2: Five components of a blockchain consensus protocol.

	Purpose	Counterpart in traditional SMR consensus protocols	Available options
Block proposal	Generating blocks and attaching essential generation proofs (for Sybil attack resistance).	Clients issuing operation requests and the primary server starting the consensus.	Proof of work (PoW), proof of stake (PoS), proof of authority (PoA), proof of retrievability (PoR), proof of elapsed time (PoET), round robin, committee-based, etc.
Information propagation	Disseminating blocks and transactions across the network.	Reliable broadcast of operation requests.	Advertisement-based gossiping, block header soliciting, unsolicited block push (broadcast), relay network (for mining pools), etc.
Block validation	Checking blocks for generation proofs and validity of enclosed transactions.	Signature check and execution of operation requests.	Proof checking (for proof-of-X block proposal), digital signature & eligibility checking (for committee-based block proposal), etc.
Block finalization	Reaching agreement on the acceptance of validated blocks.	Servers reaching an agreement on current state, executing requests and logging the result.	Longest-chain rule, GHOST rule, BFT and other Byzantine agreements, checkpointing, etc.
Incentive mechanism	Promoting honest participation and creating network tokens.	N/A.	Network token rewards (block rewards, transaction fees), eligibility for issuing new transactions, etc.

be valid and clear of double spending. As was pioneered by Bitcoin, the transactions are often organized in a Merkle tree. Merkle tree is a data structure widely used for data storage and efficient data integrity check [145]. In blockchain, every block contains one Merkle tree in which each leaf node is labeled with a transaction hash. The Merkle tree root serves as a digest of the transaction set and is placed in the block header. The block header also contains a hash of the previous block (except in the genesis block) and other configuration information, which typically includes a timestamp and the blockchain state at block generation. The growing chain of blocks and the aforementioned transaction format essentially constitute the blockchain data structure used for the storage, serialization, and validation of new transactions which are continuously injected into the network.

Aside from recording the transaction history, the blockchain can also record auxiliary information used for other purposes. The locking and unlocking scripts associated with transaction inputs and outputs can be repurposed for constructing off-chain payment channel (eg. Lightning Network [172]) and global state machine that helps build smart contracts, which are the basis of many important applications such as supply chain management and decentralized autonomous organization. The block header may also contain extra fields that facilitate system coordination. For example, Ethereum's proof-of-stake (PoS) scheme Casper FFG [180] utilizes smart contract to implement the staking process; Algorand [96] attaches a cryptographic proof to each new block to show the block proposer's eligibility to propose. As a result, the blockchain can hold the necessary control information usable by the consensus protocol. We will revisit these protocols in later sections.

2.3.2 Consensus Goal

The goal of a blockchain consensus protocol is to ensure that all participating nodes agree on a common network transaction history, which is serialized in the form of a blockchain. Based on the previous discussion on BFT consensus and the consensus goal abstraction provided in [230], we similarly define the following requirements for blockchain consensus:

- Termination: At every honest node, a new transaction is either discarded or accepted into the blockchain, within the content of a block.
- Agreement: Every new transaction and its holding block should be either accepted or discarded by all honest nodes. An accepted block should be assigned the same sequence number by every honest node.
- Validity: If every node receives a same valid transaction/block, it should be accepted into the blockchain.

- Integrity: At every honest node, all accepted transactions should be consistent with each other (no double spending). All accepted blocks should be correctly generated and hash-chained in chronological order.

The termination and validity requirements are similar to their counterparts in classical distributed consensus, as they represent the system's liveness. The agreement requirement is enhanced with total ordering, which represents the serialization of blocks and transactions. The integrity requirement dictates the correctness of the origin of transactions and blocks, fulfilling the promise of anti-double-spending and ledger tamper-proofing. These requirements can serve as the design principles of new blockchain protocols. For different application scenarios, they can be tailored or supplemented with more specification.

2.3.3 Components of Blockchain Consensus Protocol

Based on the discussion on consensus goal and our digest of the blockchain documentation corpus, we identify five key components of a blockchain consensus protocol:

- Block Proposal: Generating blocks and attaching generation proofs.
- Information Propagation: Disseminating blocks and transactions across network.
- Block Validation: Checking blocks for generation proofs and transaction validity.
- Block Finalization: Reaching agreement on the acceptance of validated blocks.
- Incentive Mechanism: Promoting honest participation and creating network tokens.

For each component we also specify its counterpart in traditional SMR consensus protocols and a list of available options in Table 2.2. The available options are non-exhaustive, as

many more are being developed at the time of writing. It is worth noting that the incentive mechanism is unique to blockchain consensus and has no counterpart in traditional SMR consensus protocols. The reason is that traditional SMR protocols are purely designed for transaction processing and serialization within a preexisting network of participants, of which the continuous participation of honest parties is presumed. Meanwhile, a typical blockchain network allows for voluntary participation and often bears numerous real-world obligations. To this end, a fair and universal incentive mechanism is needed to encourage honest participation, so as to sustain the system's reliable operation. For large-scale permissionless blockchains, a robust incentive mechanism along with the block generation proofs also help demoralize Sybil attackers.

Though the five components are all vital to successful blockchain consensus, a new blockchain consensus protocol proposal may not cover all of them. For example, the incentive mechanism is indispensable to permissionless blockchain networks especially those carrying a financial responsibility; however for permissioned blockchains in which participation is sanctioned as a privilege (similar to a traditional distributed computing system), it is not a must-have. Interestingly, many new public blockchain initiatives have been fixating only on block proposal, while inheriting the other four components from the Nakamoto consensus protocol of Bitcoin. This is likely due to that Bitcoin's PoW-based block proposal attracts the most criticism for its limited scalability and inefficient energy use. For this reason, block proposal mechanism can be a good reference angle for a general classification of consensus protocols.

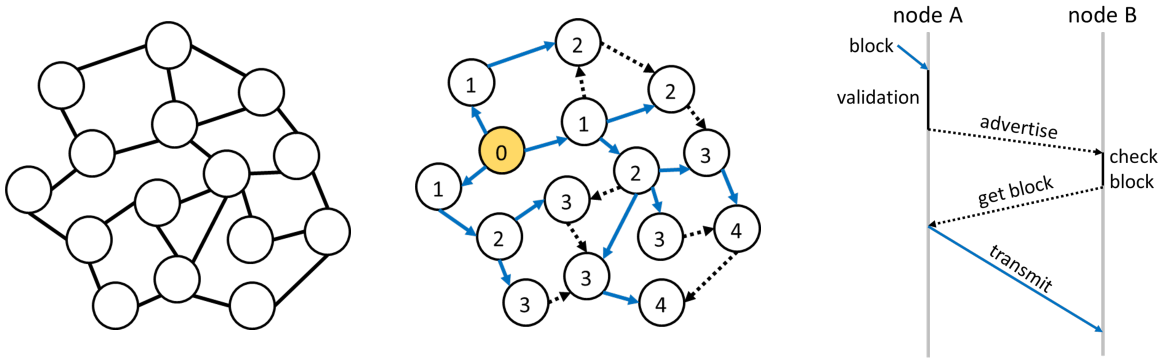


Figure 2.5: A toy example of block propagation in the Bitcoin network. Left: The P2P network structure, an undirected graph. Middle: The gossiping process. A solid blue arrow represents one-hop block propagation (advertise→get block→transmit), while a dotted black arrow represents only advertise. Number denotes the gossiping hop (0 for the block producer). Right: Block propagation in one hop.

2.4 The Nakamoto Consensus Protocol and Variations

The Nakamoto consensus protocol is the key innovation behind Bitcoin [151] and many other established cryptocurrency systems such as Ethereum [227] and Litecoin [134]. In this section we use Bitcoin as the application background to introduce the Nakamoto consensus protocol and summarize its drawbacks and vulnerabilities. We also introduce two well-known improvement proposals and four hybrid PoW-BFT protocols in the later part of this section.

2.4.1 Network Setting and Consensus Goal

In blockchain networks, block or transaction messages are propagated across the P2P network through gossiping. Fig. 2.5 shows an example of block propagation in the Bitcoin network. Specifically, the one-hop propagation adopts advertisement-based gossiping, as was first characterized in [66]. For each new block received and validated, a node advertises it to peers, who will request for this block if it extends their local blockchain. The gossiping process continues until every node in the network has this block.

Compared to the general consensus goal introduced in Section 2.3, Nakamoto enhances the termination requirement with the probabilistic finality specification:

- Probabilistic Finality: For any honest node, every new block is either discarded or accepted into its local blockchain. An accepted block may still be discarded but with an exponentially diminishing probability as the blockchain continues to grow.

The probabilistic finality property echoes the probabilistic termination property for asynchronous consensus. As we will show later, because of this property the Nakamoto consensus protocol can only achieve eventual double-spending resistance in a decentralized network of pseudonymous participants.

2.4.2 The Nakamoto Consensus Protocol

In correspondence to the five components of a blockchain consensus protocol, the Nakamoto consensus protocol can be summarized by the following rules:

- Proof of Work (PoW): Block generation requires finding a preimage to a hash function so the hash result satisfies a difficulty target, which is dynamically adjusted to maintain an average block generation interval.

- Gossiping Rule: Any newly received or locally generated transaction or block should be immediately advertised and broadcast to peers.

- Validation Rule: A block or transaction needs to be validated before being broadcast to peers or appended to the blockchain. The validation includes double-spending check on transactions and proof-of-work validity check on block header.

- Longest-chain Rule: The longest chain represents network consensus, which should be accepted by any node who sees it. Mining should always extend the longest chain.
- Block Rewards and Transaction Fees: Generator of a block can claim a certain amount of new tokens plus fees collected from all enclosed transactions, in the form of a coinbase transaction to itself.

The hashing-intensive PoW mechanism is designed for mitigating Sybil attacks. Due to Bitcoin's permissionless and pseudonymous nature, Sybil attackers can obtain new identities or accounts with little effort. Hashing power, however, comes from real hardware investment and cannot be easily forged. The longest-chain rule implies that the stabilized prefix of the longest chain can act as a common reference of the network history, given that no one is authoritative in Bitcoin's decentralized network. Block Brewards and transaction fees are used to incentivize miners to participate honestly and inject new coins into circulation.

To better illustrate how these rules harmonize with each other, we present an abstracted version of the Nakamoto protocol in Algorithm 3. During block generation, a higher mining difficulty demands more brute-force trials in order to find a fulfilling nonce. To ensure every block is sufficiently propagated before the next block comes out, the mining difficulty is adjusted every 2016 blocks so that the expected block interval remains a constant value (10 minutes in Bitcoin) no matter how the gross hashing power fluctuates.

Fork Resolution. Ideally, the 10-minute block interval should be enough to ensure the thorough propagation of a new block so that no block of the same height is proposed. However due to the delay during the message propagation and the probabilistic nature of the hashing game, the possibility of two blocks of the same height being propagated concurrently in the network can not be ignored. This situation is called a "fork", detectable by any node. Correspondingly, the longest-chain rule provides the criterion for fork resolution. Assume a

Algorithm 3: Nakamoto consensus protocol general procedure

```
/* Joining network */
Join the network by connecting to known peers;
Start BlockGen();
/* Main loop */
while running do
    if BlockGen() returns block then
        Write block into blockchain;
        Reset BlockGen() to the current blockchain;
        /* Gossiping rule */
        Broadcast block to peers;
    end
    /* Longest-chain/validation rule */
    if block received & is valid & extends the longest chain then
        Write block into blockchain;
        Reset BlockGen() to the current blockchain;
        Relay block to peers;
    end
end
/* PoW-based block generation */
Function BlockGen():
    Pack up transactions (including coinbase);
    Prepare a block header context C containing the transaction Merkle tree root, hash of the last block
      in the longest chain, timestamp, and other essential information reflecting blockchain status;
    /* PoW hashing puzzle */
    Find a nonce that satisfies the following condition:

                        Hash(C|nonce) < target

      wherein more preceding zero bits in target indicates a higher mining difficulty;
    return new block;
end
```

miner receives two valid blocks B_1^k, B_2^k of the same block height k sequentially, then a fork is detected by this miner. It chooses B_1^k (the first arrived) to continue and may encounter the following cases:

- Case 1: If receiving or successfully generating a block B_1^{k+1} confirming B_1^k, accept B_1^k, B_1^{k+1} and orphans B_2^k.

- Case 2: If receiving a block B_2^{k+1} confirming B_2^k, switch to B_2^{k+1} and accept B_2^k, then orphans B_1^k.

- Case 3: If simultaneously receiving two blocks B_1^{k+1} and B_2^{k+2} confirming respectively B_1^k and B_2^k, choose one to follow and continue until case 1 or 2 is met.

Wherein to "orphan" a block means to deny it into the main chain. Because of the randomized nature of PoW mining, the likelihood of encountering case 3 drops exponentially as time elapses, reflecting the probabilistic finality of Nakamoto consensus.

Security Analysis. In contrast to the classical distributed computing system whose fault tolerance capability is characterized by the number of faulty nodes the system can tolerate, fault tolerance of Nakamoto consensus is by characterized by percentage of adversarial hashing power the system can tolerate. It is proved by Garay et al. [90] that if the network synchronizes faster than the PoW-based block proposing rate, an honest majority among the equally-potent (in hashing power and bandwidth) miners can guarantee the consensus on an ever-growing prefix of the blockchain. The prefix represents the probabilistically stable part of the blockchain. As long as less than 50% of total hashing power is maliciously controlled, the blocks produced by honest miners are timely propagated, the main chain contributed by the honest majority can eventually outgrow any malicious branch.

From the perspective of classical distributed consensus, Nakamoto consensus cleverly circumvents the fundamental 1/3 BFT bound by adopting probabilistic finality. In classical BFT consensus if more than 1/3 of population are malicious, the honest nodes will end up deciding conflicting values, leading to consensus failure. In Nakamoto consensus, however, conflicting decisions are allowed temporarily in the form of blockchain forks, as long as they will be eventually trimmed out by continuing effort of the honest majority. Therefore, the 1/3 BFT bound is not applicable to Nakamoto consensus or other blockchain consensus protocols designed for probabilistic finality. Readers are referred to Abraham et al. [2] for an interesting discussion on the correspondence between Nakamoto consensus and classical

BFT-SMR framework.

As for double-spending resistance, assuming the adversary controls α fraction of the total hashing power and wishes to double-spend an output which is m blocks old, it needs to redo the PoW mining all the way from m blocks behind and grow a malicious chain fast enough to overtake the incumbent main chain. The probability of this adversarial catch-up is $(\frac{\alpha}{1-\alpha})^m$ if $\alpha < 50\%$, which drops exponentially as m increases, reflecting probabilistic finality. This probability equals to 1 if $\alpha \geq 50\%$. As a result, the 50% threshold is the safeguard behind Bitcoin's probabilistic finality, as well as resistance to double-spending and transaction history tampering.

2.4.3 Deficiencies and Vulnerabilities of Nakamoto Consensus

The popular implementations of Nakamoto Consensus such as the one used by Bitcoin suffers from several known deficiencies and vulnerabilities.

Tight Tradeoff Between Performance and Security: The Nakamoto consensus is widely criticized for its low transaction throughput. For instance, Bitcoin can process up to 7 TPS meanwhile the Visa payment network can process more than 24,000 TPS [213]. The limited performance of Nakamoto consensus follows from the security implication of its probabilistic finality and two protocol parameters: block interval and block size. As we discussed previously, the 10-minute block interval ensures every new block is sufficiently propagated before a new block is mined. Reducing the block interval increases the transaction capacity, but will leave new blocks insufficiently propagated and causes more forks incidents, undermining the security of the main chain. Note that although any fork can be resolved given enough time, the higher the fork rate, the larger the portion of honest mining power is wasted, which enables a double-spending attacker to overthrow the main chain with less

than 50% mining power (estimated 49.1% by Decker et al. [66] in 2013). On the other hand, increasing the block size (currently 1MB) has the same effect, since larger block sizes lead to higher block transmission delays and insufficient propagation. According to the measurement and analysis by Croman et al. [61] in 2016, given the current 10-min block interval the maximum block size should not exceed 4MB, which yields a peak throughput of 27 TPS.

Energy Inefficiency: As of May 2022, an average Bitcoin transaction consumes 2111 KWh of electricity which can power 72 U.S. households for a day [71]. This enormous energy consumption is directly caused by the PoW-based block proposing scheme of Nakamoto consensus. As Bitcoin network's gross mining capacity grows, the Nakamoto consensus protocol has to raise the mining difficulty to maintain the average 10-min block interval, which in turn encourages miners to invest into more mining equipment with higher hashing rates. This vicious cycle shall continue as Bitcoin gains more popularity. In response, the blockchain community has come up with various block proposing schemes such as proof of stake (PoS), proof of authority (PoA), proof of elapsed time (PoET) as energy-saving alternatives to PoW.

Eclipse Attack: As was discussed above, the security of Bitcoin network relies on the hashing power and communication capability of honest miners. If a powerful attacker manages to dominate the in/outward communication between a victim miner and the main network (i.e. "eclipsing"), then the victim will no longer be able to contribute to the extension of the main chain [104]. Assume the percentage of hashing power controlled by the eclipse attacker, the eclipsed victims, the remaining honest miners are α, ϵ, $1-\alpha-\epsilon$, then the attacker's mining power shall be amplified to at least $\frac{\alpha}{1-\epsilon}$. If the attacker decides to exploit the eclipsed victims for growing the malicious chain, its mining power can be further enhanced up to $\alpha+\epsilon$ [95]. As a result, a double-spending attack becomes viable for the

eclipse attacker if $\alpha + \epsilon > 50\%$. Eclipse attack is in fact an exploit of the weak connectivity of permissionless peer-to-peer network based upon the Internet, which is subject to unpredictable physical bottlenecks and adversarial influences. A general approach to counter eclipse attacks is to secure the communication channels and increase the connectivity and geographical diversity of the peer-to-peer connections.

Selfish Mining: The 50% fault tolerance of Nakamoto consensus is built upon the assumption that all miners (both honest and malicious) strictly follow the broadcast rule that new blocks are broadcast immediately upon successful generation. If a malicious mining group withholds newly mined blocks and strategically publicizes them to disrupt the propagation of blocks mined by honest miners, they can partially nullify the work of honest miners and amplify their effective mining power. This strategy is known as selfish mining. It is shown by Eyal et al. [82] that a selfish mining group can generate a disproportionately higher revenue than that from honest mining if the group's mining power surpasses a certain threshold θ, assuming the group has a certain communication capability measured by $\gamma \in [0, 1]$, which is the fraction of honest nodes that will follow the malicious chain in case of forks. As a result, the selfish mining group can attract new miners and eventually outgrow the honest miners. Notably, this threshold approaches zero if the selfish mining group are able to convince almost all honest miners to follow the malicious chain (i.e. $\gamma \to 1$). It is also shown that by adopting a randomized chain selection strategy at honest miners, which is equivalent to setting $\gamma = 0.5$, the threshold can be raised up to 25%. A later work by Sapirshtein et al. [183] shows that an optimized selfish mining strategy can further enhance the selfish mining pool's effective mining power fraction from α to the upper bound $\frac{\alpha}{1-\alpha}$ (achievable when $\gamma = 1$).

Selfish mining attack and eclipse attack have only happened to smaller blockchains such as Monacoin [102], but never to Bitcoin or other mainstream blockchains. This is probably

due to two reasons. First, miners in established public blockchain networks actually care about the system's longevity and reputation, which can positively affect the exchange rate of the cryptocurrency and thus their mining revenue. Second, established blockchains tend to be better connected (reflected by the existence of dedicated mining pools and relay networks), which allows for an effective detection of any selfish mining and eclipse attack behavior.

Mining Pools and Centralization Risk: According to the incentive mechanism of Nakamoto consensus, the mining revenue of a miner is proportional to its computing power. Since bitcoins can be traded for fiat currencies at exchanges, higher-earning miners have the financial advantage to purchase more efficient mining hardware, which consumes less joules per hash operation. Furthermore, higher-earning miners are often backed by large organizations that can direct huge capital into the mining business. As a result, small individual miners are either forced out of the game, or alternatively join in mining pools for stabler income. All members in a mining pool are registered with a coordinator and work to extend a common chain, while transaction validation, packaging and block proposal can be performed independently. To incentivize pool participation, block rewards are redistributed among the pool through a reliable remuneration scheme so that every pool member routinely gets a fair share of the pool's mining rewards according to its registered computing power.

In fact, joining in a mining pool has become the dominant way of participation in major PoW-based blockchains. The measurement study by Gencer et al. [93] in 2018 shows that throughout a one-year observation period, over 50% of the gross mining power was controlled by eight mining pools in Bitcoin and five mining pools in Ethereum. Moreover, the empirical study by Kondor et al. [113] in 2014 shows that the wealth distribution among Bitcoin addresses has been converging to a stable exponential distribution, and the wealth accumulation of node is positively related to its ability to attract new connections, which is another advantage of established large miners.

2.4.4 Improvements to the Nakamoto Consensus Protocol

The GHOST Rule. The greedy heaviest-observed subtree (GHOST) block finalization rule was proposed by Sompolinsky et al. [193] for Bitcoin in 2015. According to the longest-chain rule, all unconfirmed blocks in a fork shall be orphaned, resulting in a waste of honest mining power which could otherwise have been used to contribute to the longest-chain's security. The longest-chain rule also limits the transaction capacity since the tight tradeoff between performance and security mandates that the block interval should be sufficiently long. The GHOST rule is an alternative to the longest-chain rule that the orphaned blocks also contribute to the main chain security, effectively reducing the impacts of forks, which allows for a shorter block interval and thus higher transaction capacity. GHOST requires that given a tree of blocks with the genesis block being the root, the longest chain within the heaviest subtree shall be used as the main chain. Similar to the Nakamoto consensus, the probabilistic finality of the heaviest subtree up to the current block height will hold as long as more than 50% of mining power are honest.

The simulation result in [193] shows that given the same block interval, applying GHOST rule leads to a slightly lower transaction throughput than that with the longest-chain rule when block interval is fixed, but near-perfectly prevents the security degradation when the block interval decreases, allowing for a higher transaction throughput. A variation of GHOST is implemented in the Ethereum blockchain, wherein the "uncle blocks" (i.e. blocks with a valid proof of work but orphaned out of the main chain) may get rewarded for their redundant mining effort. As a result, Ethereum adopts a much shorter block interval (10-15 seconds) and achieves up to 25 TPS throughput, in contrast to Bitcoin's 10-minute block interval and 7 TPS throughput.

Bitcoin-NG. Bitcoin-NG was proposed by Eyal et al. [83] in 2016 to scale up Bitcoin's transaction capacity. A variant of Bitcoin-NG called Waves-NG [220] is currently used in Wave Platform, a blockchain initiative. The key insight of Bitcoin-NG is decoupling block generation into two planes: leader-election and transaction serialization, which respectively correspond to two types of blocks: key blocks and micro blocks. The key blocks resemble Bitcoin's blocks, which contain a solution to a hash puzzle representing the proof of work and have an average block interval of 10 minutes, except for the actual transactions which are included in the micro blocks. Once a key block is mined, all subsequent micro blocks shall be generated by the current key block miner until the generation of the next key block. The generation of micro blocks is deterministic and does not contain proof of work. As a result, the micro block frequency is in the control of the key block miner (up to a maximum) to accommodate as many transactions as possible. The blockchain data structure of Bitcoin-NG is shown in Fig. 2.6.

The longest-chain rule is still applied to finalize and resolve forks of key blocks. As for the micro blocks, since they are batch-generated by key block miners, Bitcoin-NG relies on a combination of a heaviest-chain extension rule and a longest-chain extension rule to finalize and resolve forks of micro blocks. To encourage honest participation and discourage the current key block miner from double-spending and other malfeasance, 60% of the transaction fees collected from micro blocks by the current key block miner are redistributed to the miner of the next key block.

As for Bitcoin-NG's performance, since key blocks do not carry transactions, the transaction throughput entirely depends on the micro block size and frequency. The micro block frequency needs to be controlled, for an excessive amount of them may exhaust the network bandwidth and cause frequent key block forks. Hypothetically, if we kept the key block frequency at 10 minute and micro block size at 1MB, set the micro block frequency to

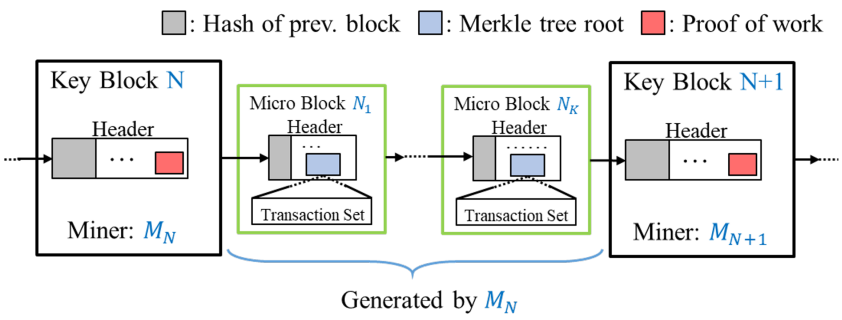


Figure 2.6: Bitcoin-NG blockchain data structure.

12 seconds (the minimum practical block interval under current Bitcoin network condition [126]), Bitcoin-NG would achieve up to 200 TPS throughput.

On the downside, due to the determinism in micro block generation, the key block miner may become a target of denial-of-service or corruption attacks. A compromised key block miner may enclose transactions selectively or finalize contradicting transactions, the inconsistency caused by which can cost the network more than one key block cycle to remedy.

2.4.5 Hybrid PoW-BFT Consensus Protocols

The limited transaction capacity and tight tradeoff between performance and security of Nakamoto consensus are much warranted by its probabilistic finality and decentralized ideal. In contrast, BFT consensus assumes fixed participants with revealed identities and achieves deterministic finality, allowing much shorter block intervals and thus much higher transaction throughput. In response, hybrid PoW-BFT protocols have been proposed to get the best of two worlds. Here we introduce four popular proposals.

PeerConsensus. Proposed by Decker et al. [67] in 2014, PeerConsensus uses a PoW-based blockchain to throttle and certify new identities joining the network, while being

agnostic to any application built upon it. The number of identities a player may control is proportional to its share of computation power, which provides Sybil resistance. With the identities established by the blockchain, the application can employ an efficient BFT protocol such as PBFT and SGMP [177] for committing transactions. The transaction fees collected are distributed to all identities equally. As a result, PeerConsensus effectively decouples participation management from transaction processing, allowing the latter to scale up throughput. On the downside, since the transaction history is not recorded in blockchain, PeerConsensus cannot control the malleability of transactions.

SCP. The scalable consensus protocol (SCP), proposed by Luu et al. [136] in 2015, incorporates BFT and sharding into blockchain consensus. The key idea of SCP is to partition the network into sub-committees (i.e. shards) with a PoW mechanism, so that each sub-committee controls a limited amount of computation power and the number of sub-committees is proportional to the network's gross computation power. This is aimed to limit the size of a sub-committee, which operates independently and curates a local blockchain using a BFT consensus protocol. A dedicated finalization committee is responsible for combining the outputs of all sub-committees into the global blockchain. A block in the global chain stores the hash and transaction Merkle tree root of every block proposed by every sub-committee. To ensure consensus safety, SCP requires each sub-committee as well as the whole network to maintain a two-thirds majority of honest computation power. However, the use of sharding and a dedicated finalization committee assumes the preexistence of network coordination, which to some extent counters the decentralized ideal of public blockchains.

In fact, using sharding to scale up blockchain transaction throughput has been extensively studied in the developer communities. Interested readers are referred to [79] for sharding roadmap by the Ethereum community.

ByzCoin. ByzCoin was proposed by Kogias et al. [112] in 2016 as a blockchain consensus protocol that leverages PoW for consensus group membership management and BFT for transaction finalization. ByzCoin takes inspiration from Bitcoin-NG's ledger structure but features a subtle difference. Instead of a linear structure, ByzCoin's ledger consists of two parallel blockchains: a keyblock chain and a microblock chain. Keyblocks are mined via PoW as in Nakamoto consensus and used for maintaining a consensus group from recent keyblock miners according to a sliding-share-window mechanism. Specifically, when a miner finds a new keyblock, it is credited one share in the consensus group and the share window moves one share forward. Only miners with shares in the window can participate in the subsequent consensus. Old shares expire once being left out of the window and so does the share owner's eligibility of consensus participation. The window length is subject to designer's choice on the consensus group size as well as the overall consensus participation fairness.

A microblock is produced by the current consensus group via an adapted PBFT protocol based on collective signing (CoSi) [202]. Compared to the original PBFT, the CoSi-based PBFT reduces the communication complexity from $O(N^2)$ to $O(N \log N)$ and thus scales better to large consensus groups. As for transaction finalization, the current keyblock miner packs up new transactions into a microblock and acts as the leader to trigger the CoSi-based PBFT. In the end, the microblock will be finalized and contain the collective signature of the consensus group and the hash pointer to the preceding keyblock, which also contains the collective signature.

ByzCoin configures that the sliding-share-window mechanism replaces one member of the consensus group at a new keyblock. This yields a slightly tighter fault-tolerance threshold than that of classical BFT consensus: $N \geq 3f+2$ is needed anytime, where N is the consensus group size and f the Byzantine population. Meanwhile, ByzCoin's PoW-based keyblock chain is still susceptible to forks. A fork can split the consensus group and potentially make

the PBFT consensus stall, which can further aggravated by the presence of selfish miners. In response, ByzCoin relies on a deterministic prioritization function tweaked with high output entropy to resolve forks timely and reduce the impact of selfish miners.

Pass and Shi's Hybrid Consensus. As a concurrent work to ByzCoin, the hybrid consensus protocol proposed by Pass and Shi [164] adopts a sliding-window idea similar to ByzCoin's but less susceptible to forks. That is, assuming the window size λ, the consensus group is populated by the last λ miners of the "stable part" of the blockchain, which is the main chain truncated off $\Theta(\lambda)$ blocks. This protocol keeps the PBFT consensus off-chain; only the consensus epoch number and a digest of the transaction log are attached to the new block. Moreover, this protocol advocates using FruitChain [165] as the underlying PoW blockchain, which was proposed by the same authors and allegedly achieves better ledger tamper-resistance than Nakamoto's blockchain.

A Short Summary. Due to the scalability concern that BFT protocol's communication overhead would be overwhelmingly high if the consensus group grew out of control, the above hybrid PoW-BFT protocols share a common trait that the PoW mechanism is used for maintaining a stable consensus group for each BFT protocol instance. Since the PoW-based participation control does not involve actual authorization and is open to any one with computation power, we consider it a form of soft permission control for public blockchains. Moreover, novel signature schemes such as CoSi [202] and aggregated signature gossip [135] can help reduce communication complexity in a sparsely-connected peer-to-peer network and allow for a larger number of consensus participants.

There are more ways to hybridize PoW and BFT in addition to the above proposals. Moreover, independently proposed cryptographic techniques are often complementary to the

hybrid design. This observation also applies to general hybrid PoX-BFT schemes.

2.5 Proof-of-Stake Based Consensus Protocols

Proof-of-Stake (PoS) originates from the Bitcoin community as an energy efficient alternative to PoW mining. In the simplest terms, a stake refers to the coins or network tokens owned by a participant that can be invested in the blockchain consensus process. From the security point of view, PoS leverages token ownership for Sybil attack mitigation. Compared to a PoW miner whose chance to propose a block is proportional to its brute-force computation power, the chance to propose a block for a PoS miner is proportional to its stake value. From the economics perspective, PoS moves a miner's opportunity cost from outside the system (waste of electricity) to inside the system (loss of capital and investment gain) [170]. Because of the lack of real mining, we often refer to a PoS miner as a validator, minter, or a stakeholder for PoS's close resemblance to investing in capital markets.

We identify four classes of PoS protocols: chain-based PoS, committee-based PoS, BFT-based PoS, and delegated PoS (DPoS). Chain-based PoS inherits many of the components of the Nakamoto consensus protocol such as information propagation, block validation, and block finalization (i.e. longest-chain rule), except that the block generation mechanism is replaced with PoS. Committee-based PoS leverages a multiparty computation (MPC) scheme to determine a committee to orderly generate blocks. BFT-based PoS combines staking with BFT consensus which guarantees deterministic finality of blocks. DPoS employs a social voting mechanism that elects a fixed-size group of delegates for transaction validation and blockchain consensus on behalf of the voters. Popular PoS initiatives are listed in Fig. 2.7.

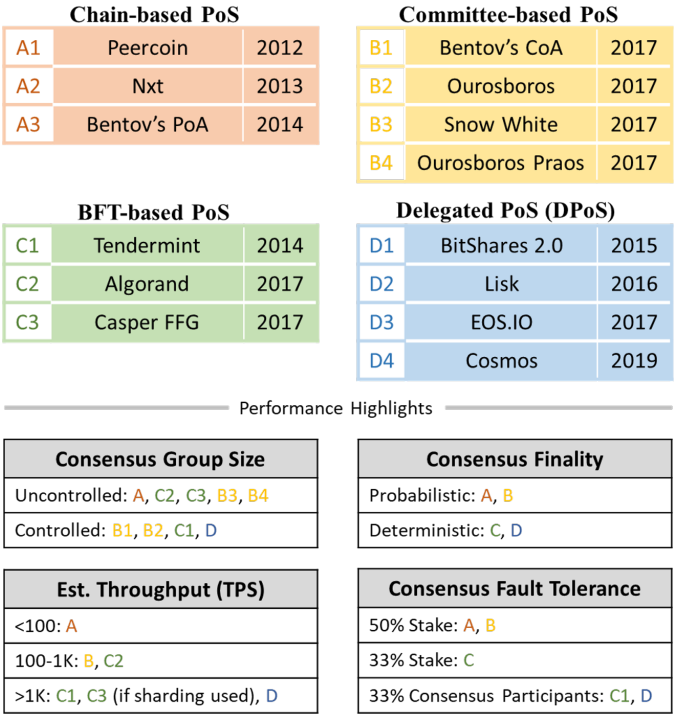


Figure 2.7: Popular PoS initiatives classified under four classes and performance highlights.

2.5.1 Chain-based PoS

Chain-based PoS is an early PoS scheme proposed by Bitcoin developers as an alternative block generation mechanism to PoW. It is within the framework of Nakamoto consensus in that the gossiping-style message passing, block validation rule, longest-chain rule, and probabilistic finality are preserved. Early full-fledged chain-based PoS blockchain systems include Peercoin and Nxt.

The general procedure of a chain-based PoS minter can be summarized by Algorithm 4. Unlike PoW, PoS does not hinge on wasteful hashing to generate blocks. A minter can solve the hashing puzzle only once for a clock tick. Since the hashing puzzle difficulty decreases with the minter's stake value, the expected number of hashing attempts for a minter to solve the puzzle can be significantly reduced if her stake value is high. Therefore, PoS avoids

the brute-force hashing competition that would occur had PoW been used, thus achieving a significant reduction of energy usage.

Algorithm 4: Chain-based PoS general procedure (Peercoin, Nxt)

```
Join the network by connecting to known peers;
Deposit in the stake pool;
Start BlockGen();
/* Main loop                                                    */
while running do
    (Same with Nakamoto's protocol except that block validation should include PoS check.)
end
/* PoS-based block generation                                   */
Function BlockGen():
    Pack up transactions and prepare a block header context C containing the transaction Merkle tree
      root and other essential blockchain information;
    /* PoS hashing puzzle                                       */
    Set up a clock (whose tick interval is a constant) and check for the following condition per clock
      tick:
                        Hash(C|clock_time) < target × stake_value
      wherein more preceding zero bits in target indicates a higher mining difficulty per unit of stake
      value;
    return new block;
end
```

Peercoin and Nxt. Both Peercoin [111] and Nxt [56] generally follow Algorithm 4. Their major difference lies in how the stake is valued. Stake value is initially proportional to stake quantity. To ensure the profitability of small stakeholders, a stake valuation scheme can be used to adjust the value of an unused stake as time passes. Peercoin uses the coin age metric for stake valuation, which lets the value of a stake appreciate linearly with time since the deposit. At the end of a block cycle, the value of the winner's stake returns to its base value. To avoid stakeholders from locking in a future block by deliberately waiting long, stake appreciation only continues for 90 days and stays flat since then. As a result, the chances of small stakeholders to generate a block are supplemented with time value that encourages them to stay participated even if they have not generated a block for a long time.

In comparison, Nxt does not appreciate stake value continuously across block cycles.

This is because the latter's coin age metric may lower the attack cost—attackers can just invest a small amount of stakes and keep attempting to generate blocks until they succeed, which is especially unfair to big stakeholders who perform honestly. Instead, Nxt requires the stake value appreciate only within one block cycle and be reset to the base value once the block cycle ends. Without coin age, Nxt addresses the monopolization problem from the incentive angle. First, stakes in Nxt are not actively managed by stakeholders; they are essentially the account balances—the more tokens held in its account, the higher the chance the stakeholder will win the right to generate a block. Second, total token supply is determined at the beginning and block rewards only come from transaction fees, which aligns a stakeholder's revenue with its validation effort. Therefore, all stakeholders have the incentive to honestly validate transactions since it is the only way to accumulate wealth.

Bentov's PoA. Compared to Peercoin and Nxt, Bentov's proof of activity (PoA) [21] is a hybrid PoW-PoS adaptation of the Bitcoin protocol that utilizes an algorithm called follow-the-satoshi (FTS) to involve staking. FTS works as follows: 1) Use a pseudo-random function (PRF) to locate an atomic piece of token (eg. satoshi in Bitcoin, wei in Ethereum) in the token universe; 2) If the atomic piece belongs to stakeholder A, then output A. With the input being a sequence of random seeds, FTS outputs a pseudo-random sequence of stakeholders such that the chance for any stakeholder to be in it is proportional to the volume of tokens owned by the stakeholder.

Bentov's PoA works as follows. At the beginning of block cycle k, an empty block header EB_k is generated according to the PoW rule and propagated across the network. After receiving EB_k, a stakeholder computes the N-tuple vector seed S as follows:

$$S_j = hash\Big(hash(EB_k)|hash(B_{k-1})|SF_j\Big) \quad \text{for } j = 1, ..., N$$

B_{k-1} is the previous block, SF is a N-tuple of fixed suffix values. N is a predefined value that should not be too large. Then S is used as the input for FTS to compute the pseudo-random sequence of stakeholders $pSeq$, which is also a N-tuple. Every stakeholder in $pSeq$ needs to sign the block and broadcast the signature; the last stakeholder in $pSeq$ wraps up the block by including transactions and the N signatures and broadcasts the final block B_k to the network. All stakeholders in $pSeq$ shares the reward of B_k with the PoW miner of EB_k. To avoid name conflict with proof of authority, we also refer to Bentov's PoA as PoAct.

Security Analysis. Chain-based PoS can tolerate up to 50% of all stakes being maliciously controlled. And since every token can be staked, the fault tolerance further generalizes to 50% of all tokens in the network. If colluding attackers control more than 50% of stakes, they can grow their malicious chain faster than the others and carry out a double-spending attack, which is analogous to the 51% attack in PoW blockchains. However, from the economic perspective, PoS attackers have lower incentives to do so because of the capital loss risk. As staking is recorded in the form of transaction scripts, the blockchain users can retrieve the staking records from which the consensus protocol can legally issue punishment to violators, such as nullifying stakes and disbarring the violators from participating in the future staking process.

2.5.2 Committee-based PoS

Chain-based PoS still relies on the hashing puzzle to generate blocks. As an alternative mechanism, committee-based PoS adopts a more orderly regime: determining a committee of stakeholders based on their stakes and allowing the committee to generate blocks in turns. A secure multiparty computation (MPC) scheme is often used to derive such a committee in the distributed network. MPC is a genre of distributed computing in which multiple parties

beginning with individual inputs shall output the same result [176]. The MPC process in the committee-based PoS essentially realizes the functionality that takes in the current blockchain state which includes the stake values from all stakeholders, and outputs a pseudo-random sequence of stakeholders (we call it the leader sequence) which will subsequently populate the block-proposing committee. This leader sequence should be the same for all stakeholders and those with higher stake values may take up more spots in the sequence. A general procedure for a stakeholder of committee-based PoS is shown in Algorithm 5. In this algorithm, the CommitteeElect() functionality can also be implemented in a privacy-preserving way with verifiable random function (VRF) [146] in that only the stakeholder itself knows if it gets elected into the committee. Well-known committee-based PoS schemes include Bentov's chain of activity (CoA), Ouroboros, Snow White, and Ouroboros Praos. These protocols and Algorand (see §2.5.3) were developed concurrently by academics around 2017 and share many common traits.

Bentov's CoA. Bentov's CoA [22] was proposed in 2016 partially based on Bentov's PoA. It follows the main routine in Algorithm 5 in that each nominated stakeholder gets to generate its own block. CoA first leverages a MPC process to generate a string S of N random bytes. Then S is fed to a FTS algorithm that outputs a pseudo-random sequence BlockGenSeq. All parties should output the same BlockGenSeq, which is then used to coordinate the generation of the next N blocks.

Ouroboros. Ouroboros was developed by Kiayias et al. [110] in 2017 and has been used as the consensus protocol for cryptocurrency Cardano. Ouroboros divides the physical time into fixed-time epochs and each epoch is subdivided into N slots, each can be used by only one slot leader to generate a block for the network. For each epoch, stakeholders with enough stake can become electors, who will collectively elect slot leaders (i.e. the committee) for the

Algorithm 5: Committee-based PoS general procedure

```
/* Joining network and staking                                          */
Join the network by connecting to known peers;
Deposit in the stake pool;
/* Main loop                                                            */
while running do
    /* Committee election                                               */
    if new block cycle then
        Participate in CommitteeElect();
        Check BlockGenSeq for my turns;
    end
    /* Block proposing & broadcast                                      */
    if my turn to generate block then
        Collect transactions and generate block;
        Write block to blockchain;
        Broadcast block to the network;
    end
    /* Longest-chain&validation rule                                    */
    if block is received & is valid & extends the longest chain then
        Write block into blockchain;
        Relay blocks to other committee members;
    end
end
/* PoS-based committee election                                         */
Function CommitteeElect():
    Fetch the current blockchain state and the stake information of all participants; use them as the
      MPC input;
    Participate in the MPC that produces BlockGenSeq, a pseudo-random sequence of block
      generation opportunities;
    return BlockGenSeq;
end
```

next epoch through a MPC procedure known as publicly verifiable secret sharing (PVSS), as is shown in Fig. 2.8. In the commit phase, elector E_i broadcasts a commitment message that includes a random secret. In the reveal phase, E_i broadcasts an opening message that reveals the previously sent secret. In the recovery phase every elector verifies that commitments and openings match and then form a seed string with the revealed secrets. All electors have the same seed string and shall obtain the same slot leader sequence after executing FTS. As we shall see next, Ouroboros' one-slot-one-leader arrangement entails stringent network synchrony, and the PVSS-based leader selection may expose the elected leaders to targeted

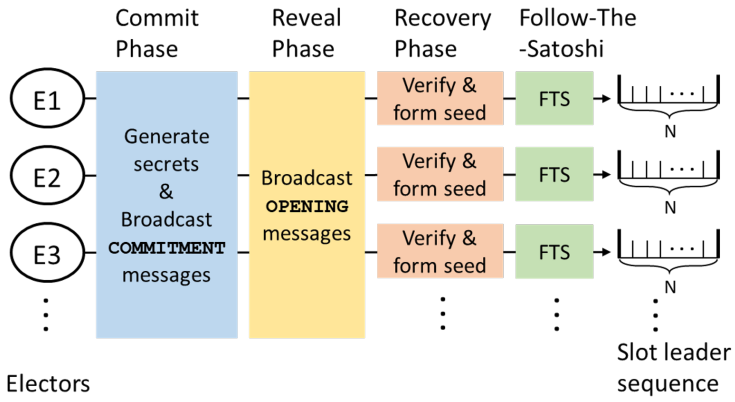


Figure 2.8: PVSS-based slot leader sequence generation in Ouroboros.

attacks.

Ouroboro Praos. Ouroboros Praos was proposed by David et al. [65] in 2017 to address two security concerns of Ouroboros. First, Ouroboros requires stringent network synchrony for slot leaders to use their allocated slots precisely, which is vulnerable to desynchronization attacks. In comparison, Ouroboros Praos is designed for partially synchronous networks wherein a maximum delay of Δ slots is allowed for message delivery, albeit Δ is unknown to electors. This is achieved by allocating empty slots that help electors to re-synchronize to the network and allowing a slot to have multiple slot leaders. Second, Ouroboros is susceptible to adaptive corruption against slot leaders. Since the leader sequence for the next epoch is known to all network participants, an attacker may target and try to corrupt slot leaders ahead of their block proposing. In comparison, Ouroboros Praos employs a locally executed verifiable random function (VRF) [146] that allows only the elector herself to know her block proposing slots for the next epoch, which is verified by the corresponding VRF proofs. Compared to Ouroboros' PVSS-based leader selection, the VRF scheme saves much of the communication overhead at the cost of local cryptographic computation. Similar schemes are also used in the concurrent developments such as Snow White and Algorand.

Because of its reduced synchrony requirement and the privacy-preserving nature of the VRF scheme, Ouroboros Praos does not limit the size of consensus participants and allows for a flexible committee. Ouroboros Praos also adopts key-evolving signatures (KES) to counter posterior corruption and provide forward security, which we will elaborate in §2.5.5.

Snow White. Snow White was developed by Daian et al. [63] in 2017. It is a PoS protocol specifically designed to accommodate the sporadic participation model in which nodes can switch online/offline arbitrarily. Similar to the committee-based PoS schemes above, Snow White employs a MPC procedure to decide the block proposing committee, each of which is issued an eligibility ticket privately. For each epoch, every stakeholder takes the current blockchain (including staking information) as input and output the committee for the next epoch. Specially, Snow White executes a modified version of the sleepy consensus protocol [166], an asynchronous consensus protocol that ensures the consensus safety in case of sporadic participation and committee reconfiguration. Compared to contemporary schemes (including Ouroboros, Praos and Algorand), this feature enables Snow White to work under harsh network conditions with frequent disconnections and volatile message delays. Moreover, Snow White uses a checkpointing scheme to finalize earlier history that protects the blockchain from posterior corruption attack [21] and adaptive key selection attack.

Security Analysis. In spite of having an orderly block proposing scheme, committee-based PoS still adheres to the longest chain rule for probabilistic finality. So long as fewer than 50% stakes are held by the malicious party, the honest parties can safely maintain the longest chain. Meanwhile, the expansion of the committee may lead to deterioration in network connectivity which can result in a significant drop in protocol performance and desynchronization of block proposal. The round-based committee election process with a predetermined round duration (eg. Ouroboros' fixed timeslot) also faces scalability problems, as

large committee sizes may lead to never ending consensus cycles due to the excessive communication overhead. To mitigate such risks, the duration of a communication round can be extended sufficiently to ensure that all broadcast messages are delivered before the participants proceed to the next round. This however leads to longer transaction conformation latency and lower throughput. A more straightforward approach is limiting the committee size by imposing a minimum stake requirement for the committee members. For example, Cardano, the cryptocurrency platform that deploys Ouroboros, mandates that every committee member should own no less than 2% of total tokens in circulation. This effectively limits the committee size to 50, safeguarding an efficient consensus process.

2.5.3 BFT-based PoS

Chain-based PoS and committee-based PoS largely follow the Nakamoto consensus framework in that the longest-chain rule is still used to provide probabilistic finality of blocks. In comparison, BFT-based PoS (or hybrid PoS-BFT) incorporates an extra layer of BFT consensus that provides fast and deterministic block finalization. Algorithm 6 shows the general procedure of BFT-based PoS at every participant. Block proposing can be done by any PoS mechanism (round-robin, committee-based, etc.) as long as it injects a stable flow of new blocks into the BFT consensus layer.

Aside from the general procedure, a checkpointing mechanism can be used to seal the finality of the blockchain (not shown in Algorithm 6). As a result, the longest-chain rule can be safely replaced by the most-recent-stable-checkpoint rule for determining the stable main chain. Popular BFT-based PoS blockchain protocols include Tendermint, Algorand, and Casper FFG. DPoS protocols such as EOSIO also use BFT consensus for block finalization within delegates.

Algorithm 6: BFT-based PoS general procedure

```
1  Join the network by connecting to known peers;
2  Start BlockGen();
   /* Main loop                                                    */
3  while running do
       /* Block proposing & broadcast                              */
4      if BlockGen() returns block then
5          Add block to its tempBlockSet;
6          Broadcast block to the network;
7      end
       /* Block validation                                         */
8      if block is received & is valid then
9          Add block to its tempBlockSet;
10         Relay block to the network;
11     end
       /* BFT consensus layer                                      */
12     if new consensus epoch then
13         Perform BlockFinBFT() on tempBlockSet;
14         Write the winning block to blockchain;
15         Clear tempBlockSet;
16     end
17 end
   /* PoS-based block generation                                   */
18 Function BlockGen():
19     Elect a block proposer, whose success rate is proportional to stake value;
20     Propose block;
21     return block;
22 end
   /* BFT-based block finalization                                 */
23 Function BlockFinBFT():
24     Participate in a BFT consensus that finalizes one winning block out of tempBlockSet;
25     return the winning block;
26 end
```

Tendermint. Tendermint was developed by Kwon et al. [36, 117] in 2014 and currently used in Cosmos Hub network [204]. It is the first public blockchain project to incorporate a BFT consensus layer and takes inspiration from the DLS protocol [74] and PBFT [49]. Tendermint works in consensus cycles. Each cycle involves a multi-round BFT consensus process to finalize one block. Each round consists of three phases: Propose, Prevote, Precommit. Specially, in the Propose phase a validator is designated by a deterministic algorithm as the block proposer in a round-robin fashion such that validators are chosen with frequency

proportional to the value of their deposited stakes. A validator continues iterating the three-phase rounds until one block receives more than 2/3 of Precommits. The validator then broadcasts Commit votes for the block and listens for other validators' Commit votes. When a block receives more than 2/3 of Commit votes, it will be finalized in blockchain. As long as more than 2/3 of validators of each round are honest, Tendermint can achieve consensus safety. On the other hand, because Tendermint selects block proposers deterministically, the future block proposers are susceptible to targeted attacks (eg. DDoS). Tendermint addresses this risk by deploying sentry nodes which act as proxies of block proposers and never reveal the IP addresses of the latter [205]. Notably, since Tendermint decouples the PoS mechanism from the BFT layer, a validator's stake value does not add weight to its consensus votes. For this reason, Tendermint is often considered an early effort on applying BFT consensus to blockchain.

Tendermint's also features an equal-sharing-style incentive mechanism instead of winner-takes-all. For every block height, the block reward is distributed among the block proposer and validators from whom the proposer received Commit votes. However, fairness may be impaired if Commit votes are not delivered in time before the next cycle, as is demonstrated by Amoussou-Guenou et al. [5].

Algorand. Algorand is a cryptocurrency system developed by Gilad et al. [96] at MIT CSAIL in 2017. It employs committee-based PoS for block proposing and Byzantine agreement for block finalization. First, similar to Ouroboros Praos' block proposer election mechanism, the election of Algorand's block proposing committee is done by a VRF scheme called cryptographic sortition which sorts candidates according to the amount of coins they own. Only those with rankings above a threshold are admitted into the committee for the next block cycle. Every individual user can check privately if it is in the committee. At each user

i, cryptographic sortition also outputs an eligibility proof signed by the user's private key $\sigma^{ep}_{sk_i}$ showing that it is truly a committee member. $\sigma^{ep}_{sk_i}$ is broadcast to the network along with the new block proposed by user i. Upon receiving the block, other users can verify the proof via the user i's public key pk_i.

On top of the cryptographic sortition-based block proposing scheme, Algorand relies on a Byzantine agreement protocol called $BA\star$ for block finalization. $BA\star$ reduces the consensus problem to binary Byzantine agreement: either agreeing on a proposed block or an empty block. In the ideal case where strong network synchrony is assumed, the committee follows $BA\star$ to exchange votes on proposed blocks so that they will decide a final block, or an empty block if no blocks pass the eligibility proof check. In a weakly synchronous network where block propagation and message exchange among committee members can suffer from uncertain delays, $BA\star$ outputs tentative blocks if none of the proposed blocks can be finalized, which results in a fork. To resolve the forks of tentative blocks, Algorand periodically runs a recovery protocol to accept a tentative block if there is any. Specially, the recovery protocol needs to be invoked by a synchronized committee. Therefore, according to the authors [96], weak synchronony is sufficient for consensus safety during $BA\star$'s routine operation while strong synchrony is required for consensus liveness when kicking off the recovery protocol for resolving forks.

Casper FFG. Casper FFG was first envisioned by Zamfir [242] in 2015 and formally presented by Buterin [39] in 2017. It is a light-weight PoS consensus layer built on top of Ethereum's current PoW-based block proposing mechanism (Ethash). Casper FFG slightly deviates from Algorithm 6 for that it directly incorporates PoS into block finalization. Algorithm 7 shows the general procedure of Casper FFG for each validator. Newly generated and received blocks are attached to the BlockTree, which is similar to the tree data structure

Algorithm 7: Casper FFG

```
Deposit in the stake pool;
/* Main loop                                                              */
while running do
    (Block proposing and block validation are the same as in Algorithm 6, except that blocks are
     attached to BlockTree rather than stored in a temporary set.)
    /* BFT consensus layer                                                */
    if new consensus epoch then
        Identify valid checkpoint blocks and attach them to CheckPointTree;
        Participate in CheckPointVote() w.r.t. CheckPointTree, which returns CP_s, CP_t;
        Mark CP_s finalized and CP_t justified;
    end
end
/* Staked checkpoint voting                                               */
Function CheckpointVote():
    Broadcast a vote for a source-target checkpoint pair in CheckPointTree;
    Check received votes against the slashing rules and then evaluate them by signer's deposited stake;
    if pair ⟨CP_s, CP_t⟩'s votes cover more than 2/3 of total deposited stakes then
        return CP_s, CP_t;
    end
end
```

used by the GHOST rule. The actual consensus subject, however, is the CheckPointTree, which is a subtree of BlockTree. Specifically, for every consensus epoch (100 in BlockTree's height or 1 in CheckPointTree's height), every validator broadcasts to peers a vote for a block as the checkpoint. The height of the block in BlockTree must be divisible by 100. The vote consists of a justified source checkpoint CP_s and its height $h(s)$, a target checkpoint CP_t and its height $h(t)$ $(h(s) < h(t))$, and the validator's signature S. All votes are broadcast to the network and are weighted by the signer's stake value. If the source-target checkpoint pair $\langle CP_s, CP_t \rangle$ is voted by validators who possess more than 2/3 of total deposited stakes, then CP_t is justified and CP_s is finalized. All blocks between CP_s and CP_t are finalized as well. Casper FFG relies on two so called Casper Commandments for ensuring consensus safety: 1) validator must not cast two distinct votes for the same checkpoint height, and 2) validator must not cast a new vote whose source-target span is within that of its existing vote. Violators are subject to slashing rules including forfeiting stakes and temporarily ban-

ning from staking. Since every vote is signed with the validator's private key and received by peer validators, Casper FFG can conveniently detect violators and enforce the slashing rules.

The current smart contract implementation of Casper FFG is documented in EIP 1011 [180]. A stakeholder becomes a participating validator by depositing a stake in the dedicated smart contract, which encodes Casper FFG and can be accessed via Ethereum transactions. Notably, Casper FFG is the preamble project of Casper Correct-by-Construction (Casper CBC), the PoS protocol family that will be used by Ethereum 2.0 to complete the transition to pure PoS [78].

To further improve performance and scalability, Ethereum 2.0 also plans to combine PoS with sharding [79]. In a nutshell, all Ethereum 2.0 participants are divided into shards. Each shard runs a blockchain instance via a consensus scheme not limiting to PoS. On the top level, the main chain, known as the "beacon chain", will be maintained by a group of known validators via a Casper CBC protocol. Each validator is randomly assigned to a shard as the shard manager and periodically commits a digest of the shard chain to the main chain. The parallelism of sharding and the energy efficiency of PoS can theoretically scale up both transaction throughput and network size. Nonetheless, the sharding scheme is still an ongoing work and faces several challenges before being harmonized with Casper CBC. They include the increased take-over risk related to small shard size, the difficulty in coordinating inter-shard communication and token transfer.

Security Analysis. BFT-based PoS's consensus fault tolerance varies among the three above-mentioned implementations. In Tendermint, although block proposers are determined based on PoS, all validators have the equal weight in the consensus process. Therefore Tendermint tolerates up to 1/3 of Byzantine validators. In comparison, Algorand and Casper

FFG tolerate up to 1/3 of maliciously-possessed stakes. In Algorand, if an attacker owns more than 1/3 of total tokens, then chance is high that more than 1/3 of the elected committee members is compromised by the attacker, leading to consensus failure of $BA\star$. In Casper FFG, if an attacker owns more than 1/3 of total deposited stakes and dominates the communication within the network, the compromised validators can vote on conflicting checkpoints without getting punished. Since a typical BFT consensus protocol can incorporate a checkpointing mechanism to ensure deterministic finality of blocks, costless simulation attacks can be naturally avoided (to introduce in §2.5.5).

2.5.4 Delegated PoS

Delegated PoS (DPoS) can be seen as a democratic form of committee-based PoS in that the committee (consensus group) is chosen via public stake delegation. It is currently used by BitShares (2015) [86], Lisk (2016) [132], EOSIO (2017) [27], and Cosmos Hub [204]. DPoS was designed to control the size of the consensus group so that the messaging overhead of the consensus protocol remains manageable. Members of the consensus group are also called delegates. The election of delegates is called the delegation process, and a general example is shown in Fig. 2.9. Small stakeholders vote for delegates with their stakes and the most voted delegates form the consensus group. In the actual case, the delegation process and the soliciting of votes may involve outside incentives. And the delegation process represents an interesting socioeconomic phenomenon.

Generally, an aspirating delegate needs to attract enough votes from normal token holders. This is often accomplished by offering a popular application and building up reputation through propaganda campaigns. By casting a vote to a delegate via a blockchain transaction, a token holder entrusts the delegate with its own stake. As a result, the delegate harvests

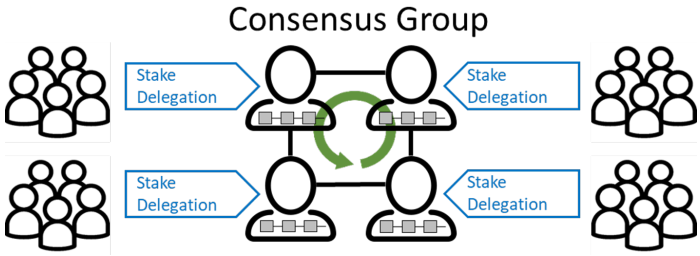


Figure 2.9: Illustration of the delegation process in DPoS.

the stake voting power from her voters and acts as their proxy in the consensus process. A token holder can switch vote to another delegate via another delegation transaction. Take EOSIO for example, anyone can be a delegate and solicit votes, but only those who ascend to top 21 can join the consensus group, among whom the right of block proposal is equally shared. EOSIO employs a pipelined PBFT-style consensus scheme to finalize the proposed blocks across the 21 delegates [124]. Specially, the physical time is divided into slots and the 21 delegates take turn to propose a block in a round-robin fashion. At each slot when a delegate proposes a block, the consensus scheme goes through pre-commitment and commitment phases and decides on the fate of the proposed block. Because of the small consensus group size and orderly PBFT-style procedure, every new valid transaction can be near-instantly propagated to the consensus group and finalized in the blockchain.

To enforce transaction validation and consensus safety, DPoS's incentive mechanism is designed to encourage honest delegation and consensus participation. Every delegate receives daily vote-reward proportional to the votes she has. Once ascending to the consensus group, delegates also receive block rewards for validation work.

Security Analysis. Assuming BFT is used by the consensus group for block finalization, which is recommended since the group size is limited, DPoS can tolerate 1/3 of delegates being malicious. For example, EOSIO can tolerate at most 6 out of 21 delegates being

malicious. In the real world they may not wish to misbehave or collude at all, since all delegates have revealed their identities to voters and would be scrutinized for any misconduct.

2.5.5 Vulnerabilities of PoS

Although heralded as the most promising mechanism to replace PoW, PoS still faces several vulnerabilities.

Costless Simulation: This is a major vulnerability of non-BFT-based PoS schemes, especially chain-based PoS in which PoS is used to simulate the would-be PoW process. Costless simulation literally means any player can simulate any segment of blockchain history at the cost of no real work but speculation, as PoS does not incur intensive computation while the blockchain records all staking history. This may give attackers shortcuts to fabricate an alternative blockchain. The four subsequent vulnerabilities, namely nothing-at-stake, posterior corruption attack, long-range attack, and stake-grinding attack are all based on costless simulation.

Nothing at Stake: Nothing-at-stake attack is the first identified costless simulation risk that affects chain-based PoS. It is also known as "multi-bet" or "rational forking". Unlike a PoW miner, a PoS minter needs little extra effort to validate transactions and generate blocks on multiple competing chains simultaneously. This "multi-bet" strategy makes economical sense to PoS nodes because by doing so they can avoid the opportunity cost of sticking to any single chain. Consequently if a significantly fraction of nodes perform the "multi-bet" strategy, an attacker holding far less than 50% of tokens can mount a successful double spending attack [141]. Nothing-at-stake problem can be practically solved by penalizing whoever multi-bets, such as forfeiting part of or all their stakes. However, the penalties could still be reversed if the attacker eventually succeeds in growing a malicious chain.

Posterior Corruption: Dubbed by Bentov [21] as "bribing attack" in 2014, posterior corruption is another attack utilizing costless simulation against PoS. The key enabler of posterior corruption is the public availability of staking history on the blockchain, which includes stakeholder addresses and staking amounts. An attacker can attempt to corrupt the stakeholders who once possessed substantial stakes but little at present by promising them rewards after growing an alternative chain with altered transaction history (we call it a "malicious chain"). When there are enough stakeholders corrupted, the colluding group (attacker and corrupted once-rich stakeholders) could own a significant portion of tokens (possibly more than 50%) at some point in history, from which they are able to grow an malicious chain that will eventually surpass the current main chain. Since posterior corruption is only possible because the private/public keys are fixed for blockchain participants, key-evolving cryptography (KEC) [88] can be applied so that the past signatures cannot be forged by the future private keys. Ouroboros Praos [65] currently adopts KEC for this purpose. Alternatively, as is in Snow White [63] and Casper FFG [39], checkpointing can be used to finalize the ledger and eliminate the possibility of posterior corruption.

Long-range Attack: Coined by the Ethereum founder Vitalik Buterin [37], long-range attack can be viewed as the ultimate form of costless simulation. It foresees that a small group of colluding attackers can regrow a longer valid chain that starts not long after the genesis block. Because there were likely only a few stakeholders and a lack of competition at the nascent stage of the blockchain, the attackers can grow the malicious chain very fast and redo all the PoS blocks (i.e. by costless simulation) while claiming all the historical block rewards. If the blockchain network is not incentivized by block rewards, the attackers can still deploy a similar long-range scheme called stake-bleeding attack [91] as long as transaction fees exist. That is, the attackers can accumulate significant amount of wealth by collecting most of historical transaction fees through the malicious chain. Through either scheme,

once the malicious chain overtakes the main chain, it is released to public and becomes the new main chain. While zero transaction fees can counter stake-bleeding attacks, general long-range attacks (and costless simulation as a whole) can be resolved by checkpointing, a more radical measure. Checkpointing is widely used in BFT protocols to ensure the finality of system agreements and safely discard older records. For permissionless blockchains (the main venue for chain-based PoS), however, checkpointing can undermine decentralization, as the finality of checkpoints requires the endorsement from certain authoritative entities.

Stake-grinding Attack: Generally, the block generation competition in Nakamoto-style blockchain with proof-of-X block proposal is a pseudo-random process that a higher X (eg. work, stake) yields a higher probability of winning the competition. However, unlike PoW in which pseudo-randomness is guaranteed by the brute-force use of a cryptographic hash function, PoS's pseudo-randomness is influenced by extra blockchain information—the staking history. Malicious PoS minters may take advantage of costless simulation and other staking-related mechanisms to bias the randomness of PoS in their own favor, thus achieving higher winning probabilities compared to their stake amounts [110]. For example, in chain-based PoS blockchains such Peercoin, at a certain historical point of the blockchain, attackers may iterate through different block headers and increase the probability of generating a valid block with their stakes [80]. For committee-based and BFT-based PoS schemes that decouple the election of block proposers from block generation, stake-grinding attacks can be mitigated by ensuring the PoS pseudo-randomness with a secure block proposer election scheme that involves minimal local information. Examples include Ouroboros, Ouroboros Praos, and Algorand.

Centralization Risk: PoS faces a similar wealth centralization risk as with PoW. In PoS the minters can lawfully reinvest their profits into staking perpetually, which allows the one with a large sum of unused tokens become wealthier and eventually reach a monopoly

status. When a player owns more than 50% of tokens in circulation, the consensus process will be dominated by this player and the system integrity will not be guaranteed. Take Ethereum's Casper FFG for example, the proposed PoS scheme is built upon the current PoW system, of which the cryptocurrency ethers can be directly used for staking. This gives initial advantages to those who have already accumulated huge wealth during Ethereum's PoW operation. Potential countermeasures against monopolization in PoS mainly come from the incentive mechanism and economic perspective. In addition to the stake valuation scheme that improves the winning chances of small stakeholders (see §2.5.1), we can use off-chain factors to complicate the staking process (EOSIO for example) and impose taxation on the blocks generated by large stakeholders, to name a few.

2.6 Comparison and Discussion

Besides Nakamoto-stype and PoS, there exist many more consensus protocols proposed for blockchain systems. Readers are referred to our complete survey [233] for more details. Table 2.3, 2.4 compare the popular consensus protocols from three aspects: five-component framework composition, fault tolerance, and transaction processing capability. The latter includes maximum throughput and transaction confirmation latency. For some consensus protocols, not all the five components are specified in the white paper. Protocols designed for permissioned blockchains (eg. BFT-SMR, PoET, PoTS) do not need to specify incentive mechanism; protocols that were initially proposed to substitute PoW (eg. PoA, PoET, PoR) inherit other components from Nakamoto's protocol by default.

The fault tolerance capability of a consensus protocol largely depends on the block finalization mechanism. For example, the 50% bound (of computing power or stake value) applies to all protocols that inherit the probabilistic finality property from Bitcoin; proto-

Table 2.3: A Comparison of Blockchain Consensus Protocols

Protocol Class (Example Realizations)	Five-Component Framework					Fault Tolerance
	Block Proposal	Block Validation*	Information Propagation	Block Finalization	Incentive Mechanism	
BFT-SMR (PBFT [49], Bft-SMaRt [23] HoneyBadgerBFT [148], Hotstuff [237])	Client operation request	Signature check	Broadcast	Mutual agreement on the same state	(N/A)	33% servers
Nakamoto (Bitcoin [151], Litecoin [134])	PoW	PoW check	Gossiping	Longest-chain rule†	Block reward, transaction fee	50% computing power
Nakamoto-GHOST (Ethereum [227])	PoW (Ethash)	PoW check	Gossiping via secure channels	A variation of GHOST rule†	Block reward, transaction fee	50% computing power
Bitcoin-NG (Waves-NG [220])	PoW for key blocks	PoW check for key blocks	Gossiping	Longest-chain rule†	Block reward, transaction fee	50% computing power
Hybrid PoW-BFT (PeerConsensus [67], SCP [136], ByzCoin [112])	PoW-based BFT committee election	PoW check	Broadcast among BFT committee	Longest-chain rule† for chain (PBFT for transactions)	⊙	33% computing power
Chain-based PoS (Peercoin [111], Nxt [56], PoAct [21])	PoS	PoS check	Gossiping	Longest-chain rule†	Block reward, transaction fee	50% deposited stake value
Committee-based PoS (Ouroboros [110], Praos [65], CoA [22], Snow White [63])	PoS-based committee election	Proposer eligibility check	Broadcast among committee	Longest-chain rule†	Block reward	50% token wealth
BFT-based PoS —Tendermint [117] (Cosmos Hub [204])	PoS-based round robin	Proposer eligibility check	Broadcast among validators	BFT (adapted DLS)	Block reward	33% token wealth
BFT-based PoS —Algorand [96]	PoS-based committee election	Proposer eligibility check	Broadcast among committee	BFT (adapted Byzantine agreement)	Block reward	33% token wealth
BFT-based PoS —Casper FFG [39] (Ethereum 2.0)	PoW (Ethash)	PoW & Checkpoint tree check	Broadcast among validators	BFT (with staked votes)	Block reward	33% deposited stake value
DPoS (EOSIO [27], Lisk [132], BitShares [86])	PoS with stake delegation	Delegate eligibility check	Broadcast among delegates	BFT (suggested)	Block reward, vote reward	33% delegates
Proof of Authority (Rinkby, Kovan, POA Network [169])	PoA	Proposer identity check	⊙ Broadcast‡	⊙ HoneyBadger-BFT‡	⊙ Transaction fee‡	50% TEEs (33% if BFT used)
Proof of Elapsed Time (Hyperledger Sawtooth family [87])	PoET within TEE	TEE certificate check	⊙ Broadcast‡	⊙ PBFT‡	⊙	50% IDs (33% if BFT used)
RPCA (Ripple [187])	Any server can propose transactions	UNL membership check	Broadcast to UNL peers	Accepting > 80% voted transactions	Transaction fee	20% nodes in each UNL

*Block validation also includes validating all transactions inside the block, which is omitted here for space saving.
†: With probabilistic finality. ‡: Unspecified in the protocol white paper, but found in one or more blockchain realizations.
⊙: Unspecified in the protocol white paper, but the Bitcoin counterpart is usable.

Table 2.4: A Comparison of DAG-based Consensus Protocols

Protocol Class (Example Realizations)	Five-Component Framework					Fault Tolerance
	Block/TX Proposal	Block/TX Validation	Information Propagation	Block/TX Finalization	Incentive Mechanism	
BlockDAG: SPECTRE [195], PHANTOM [194]	PoW	PoW check	Gossiping	Pairwise ordering† / k-cluster-blocks ordering†	⊙	50% computing power
BlockDAG: Conflux [126]	PoW	PoW check	Gossiping	GHOST rule to finalize pivot chain†	⊙	50% computing power
TxDAG: Tangle (IOTA [173])	Approving two tips & PoW	Tip approval check & PoW check	Gossiping	Highest cumulative weight rule†	Eligibility for issuing new transactions	50% computing power
TxDAG: Byteball (Obyte [54])	Approving tips per main chain index	Tip approval check	Gossiping	Witnesses consensus on main chain	Commissions collected from storage fees	50% (or 6) witnesses
TxDAG: Nano [125]	Cross-chain sender-receiver & PoW	Sender account & PoW check	Gossiping	stake-weighted voting†	(Unspecified)	50% token wealth
TxDAG: Avalanche [179]	Approving one parent	Parent approval check	Gossiping	Confidence ordering to solve conflict†	(Unspecified)	50% participants

†: With probabilistic finality. ⊙: Unspecified in the protocol white paper, but the Bitcoin counterpart is usable.

cols with BFT-style block finalization typically achieve 33% fault tolerance and guarantee deterministic finality. In hybrid schemes, the 33% fault tolerance is usually coupled with the block proposal mechanism or a preexisting identity scheme. In Hybrid PoW-BFT and BFT-based PoS, each node's voting weight in the BFT consensus stage is augmented by its block proposing capability. This is achieved by either linking a player's opportunity of participation in BFT consensus to the PoX process (eg. Hybrid PoW-BFT, Tendermint), or directly weighting a player's vote in the BFT consensus by the player's stake value (eg. Algorand, Casper FFG). In other hybrid schemes such as PoA-BFT and PoET-BFT, the PoX process is purposed for identity management and thus the fault tolerance threshold is 33% of identities. Directed acyclic graph (DAG)-based protocols[2] generally achieves probabilistic finality (except Byteball) and 50% fault tolerance of either computing power, consensus participants, or token wealth. The statistics on transaction processing capability were fetched from either the simulation result in protocol white paper or documented experiment. The throughput

[2]See our extended survey [233] for more details on DAG-based blockchains.

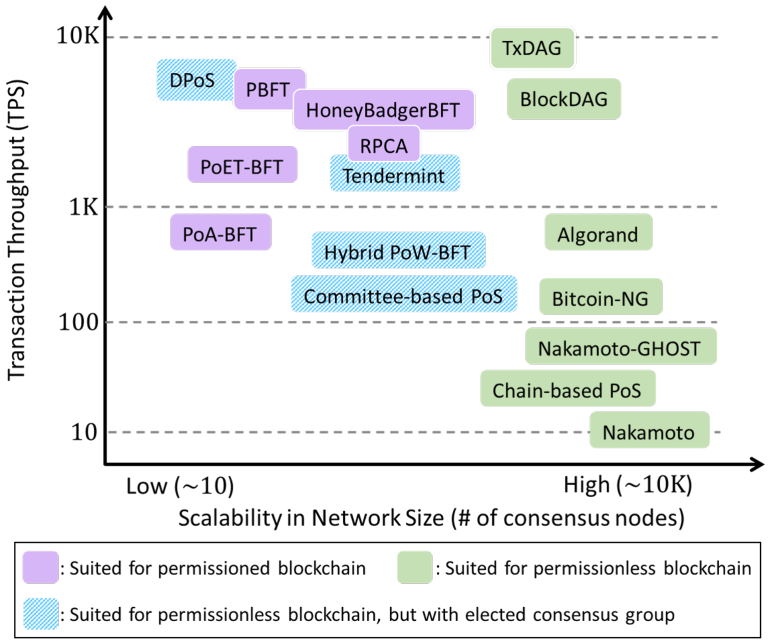


Figure 2.10: Comparing different blockchain consensus protocols on transaction throughput and scalability in network size.

metric measures the maximum TPS a protocol can handle, and is only precise to the scale of magnitude, i.e. sub-ten, tens, hundreds, or thousands. The confirmation latency metric estimates time for a submitted transaction to be finalized in the blockchain. We observe that protocols with probabilistic finality tend to have higher confirmation latency (typically more than 1 minute), and high throughput is often accompanied by low conformation latency.

We are also interested in another performance metric—scalability in network size, which measures how well a consensus protocol can maintain its transaction capacity when network size grows. Given the difficulty of performing large-scale experiments on actual blockchain networks, the scalability results are either fetched from the simulation study in protocol white paper or based on our speculation on the current network status. It is worth noting that transaction throughput and scalability in network size are often recognized as two integral parts in scalability evaluation of a blockchain system [61, 153, 214].

Fig. 2.10 illustrates performance of consensus protocols with respect to transaction throughput and scalability in network size and suitability for permissioned or permissionless blockchains. For protocols with BFT-style block finalization to achieve thousands of TPS, the network size should be small, typically around several hundred. Meanwhile, protocols that work for large-scale public networks are predominantly permissionless and employ a block finalization mechanism that only achieves probabilistic finality. Their transaction throughput is usually capped by a hundred TPS for security reasons. Generally, protocols illustrated farther to the top right in Fig. 2.10 tend to achieve better overall performance and scalability. However, the security implication of such superiority, exemplified by Bitcoin-NG and DAG-based protocols, is subject to scepticism and attracts ongoing research effort.

As for the suitability for permissioned or permissionless blockchains, we stress that protocols that need a stable consensus group with participation control and identification are suitable for permissioned blockchains. These protocols include PBFT, HoneyBadgerBFT, PoET-BFT, PoA-BFT, and RPCA. In comparison, protocols including Nakamoto, Nakamoto-GHOST, chain-based PoS, Bitcoin-NG, Algorand, and Tangle are specifically designed for large-scale permissionless blockchains. Meanwhile DPoS, Tendermint, hybrid PoW-BFT, committee-based PoS are designed for permissionless scenarios but rely on an election mechanism for maintaining a stable consensus group, of which the identities may be revealed for public supervision.

2.7 On Designing Blockchain Consensus Protocol

In this section we offer a succinct tutorial that hopefully helps protocol designers form reasonable objectives and avoid pitfalls.

2.7.1 The Paradigm Shift in Protocol Design

In the early days of blockchain development, consensus protocol designs were often coupled with designer's ingenuity and heuristics. Anticipating that the increasing number of participants would lead to more block contentions and blockchain forks in Bitcoin, Nakamoto designated the longest-chain rule for block finalization and cautiously fixed the average block interval to ten minutes for consensus security [151]. Litecoin [134] and Ethereum [227] adopted shorter block intervals for faster transaction settlement. Observing the energy inefficiency of Bitcoin and the financial value of unused tokens, Peercoin pioneered the PoS mechanism which could substitute PoW with far less energy consumption [111]. Yet its stake valuation scheme seemed to be heuristically designed. These early-day initiatives also commonly lacked sufficient security analysis, which were often scrutinized by researchers and practitioners [56, 82, 90].

Later blockchain consensus protocols, especially those proposed after 2016, have started to take inspirations from the established research of distributed computing, cryptography, and trusted computing. For example, HoneyBadgerBFT [148], BEAT [72], Algorand [96] extend reliable broadcast and Byzantine agreement to blockchain scenarios. Committee-based PoS protocols, such as Ouroboros [110], Snow White [63], Bentov's CoA [22], make use of MPC for the management of consensus committee and block proposal. PoET [87] and PoTS [6] leverage trusted execution environment (TEE) for secure block proposal and mining substitution. Furthermore, some proposals incorporate innovative cryptographic primitives for enhancing privacy of network participants. Zero-knowledge proof (ZKP) and ring signatures are used by Zerocash [184] and Monero [159] to construct privacy-preserving transactions that hide essential transaction values and sender-receiver addresses; Algorand and Ouroboros Praos [65] use cryptographic sortition and VRF for the PoS-based election of consensus committee without revealing participant's stake value. The privacy of consensus participants in

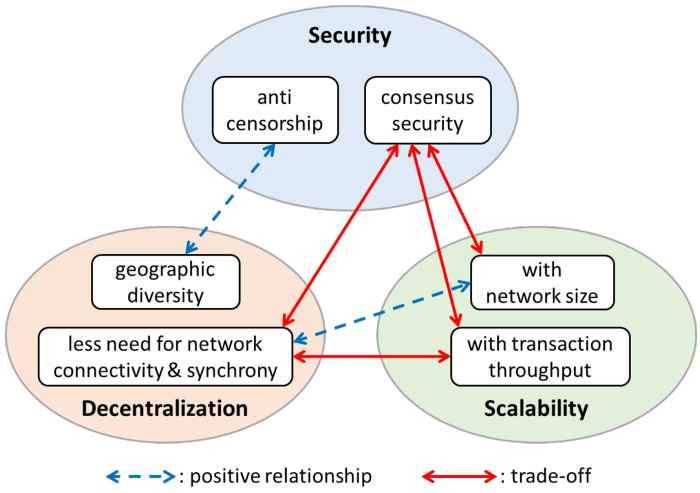


Figure 2.11: Illustration of the security-decentralization-scalability trilemma.

large-scale permissionless blockchains is still a fresh research topic.

Besides increased sophistication in design, new protocol proposals are also often accompanied by formal security analysis, via security frameworks such as the random oracle model [47] and universal composibility [45]. In summary, This paradigm shift has brought the design and analysis of blockchain consensus protocols into rigorous research, echoing Cachin et al. [41] that blockchain design should follow the rigor established by prior wisdom.

2.7.2 The Security–Decentralization–Scalability Trilemma

Observing the existence of various blockchain consensus protocols, we stress that the design of consensus protocol should seek a balance between three objectives: security, decentralization, and scalability. Security refers to the blockchain's consensus security in the presence of malicious players and anti-censorship capability. The two concepts correspond to the safety and liveness property in classical distributed consensus. Decentralization refers to the decentralization profile of the blockchain network, which is normally associated with geographic diversity, connectivity pattern, and synchrony of networked nodes. Scalability means two-

fold—the system needs to remain secure and efficient with rising transaction throughput as well as larger network size. Scalability is often considered as a broader concept of system performance.

Fig. 2.11 sketches the relationships and trade-offs between the three objectives. For most blockchain consensus protocols especially those tailored for financial application, security is always the top priority. Meanwhile, security level can be practically traded for system's scalability. Lower security requirement gives rise to more scalable protocol design. For classical BFT protocols, lower fault tolerance threshold (the "f" parameter) leads to fewer consensus rounds [30, 121] or less communication per round [49], effectively alleviating message complexity caused by large network sizes. For many public blockchain networks that use Nakamoto consensus, the probabilistic finality and mandatory multi-block confirmation give rise to permissionless access and higher scalability in network size. Meanwhile, a shorter block interval yields higher transaction throughput but also leads to higher chance of 51%-attack [66]. For DAG-based protocols, the security implications and their scalability in both network size and throughput are not well studied and remain an ongoing research.

A protocol designer should also anticipate the security implications of a blockchain network's decentralization profile. A more decentralized network tends to be less synchronized, owing to the increased diversity of geographic locations and sparsity of connections. On one hand, the increased geographic diversity makes censorship by one suppressing regime more challenging, which effectively enhances the system's liveness. On the other hand, the reduced synchrony implies that inter-player communications are less bounded by delay requirements. This contributes to the heterogeneity of network connections and allows better connected players to gain unfair advantages, which potentially impairs consensus security of the network. Take protocols with the gossiping rule and longest-chain rule for example, the heterogeneity of network connections gives well-connected nodes a communication

advantage of disseminating new blocks faster in the network than the less-connected ones, effectively enhancing the former's winning chance in blockchain fork races. As a result, in a sparsely connected network but with one well-connected adversary, the adversary may need significantly less than 50% of mining power to commit a double-spending attack.

Last but not least, a protocol designer should be aware that the tradeoff between decentralization and scalability depends on the practical needs. For example, Bitcoin was designed to operate in a weakly synchronized network and there is no enforceable criterion for message delivery and timeout. As a result Nakamoto consensus protocol needs to rely on best-effort mining and long block intervals to achieve the security against double spending, at the price of low transaction throughput. In comparison, BFT-based protocols are designed for permissioned networks where every participant operates with revealed identity and the overall network is small and well-connected and with enforceable time-out mechanisms. This enables the network to achieve higher transaction throughput. On the flip side, the high dependence on network connectivity and synchony for throughput brings high messaging complexity and network management overhead. When the network grows in size and inevitably becomes sparse, message relaying can be a major burden for each participant which potentially leads to jammed communication channel and stalled consensus process. Therefore, under a certain security premise, increased need for network connectivity and synchrony leads to higher transaction throughput but lower scalability with network size.

2.7.3 Protocol Composability

The discussions above demonstrate that there is never a one-fits-all blockchain consensus protocol. The existence of hybrid consensus protocols highlights the possibility to cherry-pick individual protocol components to fulfill specific application needs. Sawtooth [87],

a Hyperledger project featuring the utilization of trusted hardware for PoET-based block proposal, has the flexibility of plugging in different block finalization schemes—Raft for crash fault tolerance and PBFT for Byzantine fault tolerance. Similarly, POA Network [169] uses PoA for block proposal and has opted for HoneyBadgerBFT for block finalization to achieve best performance under asynchronous network conditions.

Besides block proposal and finalization, the choice of information propagation mechanism largely depends on the underlying network topology. In a public blockchain network where gossiping is the default information propagation method, broadcast can be used among a small set of identified nodes for better efficiency [61]. The block validation mechanism is associated with the block proposal mechanism and takes the current blockchain as input. The incentive mechanism is aimed for promoting honest execution of the former tasks. It needs to account for off-chain factors as well, such as the long-term system sustainability, financial stability, and subtle security issues [37, 80, 91].

2.8 Conclusion

In this review we provided a summary of classical fault-tolerance consensus research, a five-component framework for a general blockchain consensus protocol, and a comprehensive review of blockchain consensus protocols that have gained great popularity and potential. We analyzed these protocols with respect to fault tolerance, performance, vulnerabilities and highlighted their use cases. Notably, many of these protocols are still under development and are subject to major changes at the time of writing. We hope the five-component framework, classification methodology, protocol abstractions, performance analyses, and discussion on protocol design provided in this survey can help researchers and developers grasp the fundamentals of blockchain consensus protocols and facilitate future protocol design.

Chapter 3

Modeling the Impact of Network Connectivity on Consensus Security of Proof-of-Work Blockchain

(Copyright notice[1])

3.1 Introduction

Decentralization is a foundational principle for blockchain technology and distributed ledger system. Envisioned by Nakamoto, the pseudonymous creator of Bitcoin [151], and later advocates, blockchain technology is secure-by-design and enables mutually distrustful parties to securely curate a shared blockchain through distributed consensus without relying on a central authority for bootstrapping the trust. Driven by incentives, consensus participants act in their self-interest to maximize rewards. Under such decentralized zero-trust setting, the security of distributed consensus relies on the premise of honest-majority, i.e. honest parties always control the majority of gross voting power in the consensus process, and in

[1]This chapter previously appeared as a conference paper published in INFOCOM 2020. ©2020 IEEE. Reprinted, with permission, from Yang Xiao, Ning Zhang, Wenjing Lou, and Y. Thomas Hou, "Modeling the Impact of Network Connectivity on Consensus Security of Proof-of-Work Blockchain," In IEEE INFOCOM 2020-IEEE Conference on Computer Communications, pages 1648–1657. IEEE, 2020 [232].

the case of proof-of-work (PoW) based blockchains, 50% of gross computing (or "mining") power [90]. This threshold is widely used for evaluating the risk of mining centralization in Bitcoin and Ethereum [17, 93, 94, 198].

However, the consensus security from honest-majority comes with two implicit assumptions. First, all nodes have the same communication capability, i.e. propagating information throughout the network equally fast. Second, during a blockchain fork race, wherein several blocks of the same height compete for one place in the blockchain, all competitors have the equal chance of being the winner. In practice, the quality of connections often differ significantly across different network regions, as has been demonstrated by various measurements [16, 93, 147, 157]. Those residing in a highly connected cluster can disseminate blocks faster than those in a less connected region. This communication advantage translates into a higher chance of dominating a fork race, and has nontrivial consequence in the security of distributed consensus. As a result of this network advantage, the adversary will no longer require 50% of gross mining power to undermine the consensus security.

Following this observation, various blockchain scaling proposals and security analyses [61, 192, 193] have identified the positive correlation between high blockchain fork rate and weak consensus security. These works generally adopt the honest-but-potentially-colluding threat model, in which any size of honest miners can join the collusion to compromise consensus security. Specifically, colluding miners seek to dominate the fork races with honest miners and achieve unfair mining gains. As a result, the colluding miners require less than 50% of gross mining power to break the consensus. However, these security analyses are largely qualitative and do not look into the impact of the actual network connectivity or information propagation dynamics.

The security impact of information propagation dynamics in Bitcoin was studied quantitatively at the macro level in [66]. It proposes a probabilistic model that estimates the

average fork rate of the Bitcoin blockchain based on the measurement of how an average block propagates in the network. The authors then regard fork rate as a security measure of the blockchain network. However, their model still assumes all miners have equal communication capability and equal chance of winning fork races, and does not consider the impact of heterogeneous network connectivity. It also does not provide a concrete case of how an adversary may exploit blockchain forks.

Another line of research focuses on adversarial strategies for selfish colluding parties [82, 95, 154, 183]. In selfish mining [82], an adversary with superior communication capability can achieve unfair mining gains by strategically withholding and releasing blocks. It proactively creates blockchain forks that nullify the efforts of honest nodes. Although these works take into account the difference in fork winning chance between the adversary and honest miners, their analyses treat the adversary's communication capability as a preexisting parameter (denoted by γ_{SM}) rather than deriving it from the actual network connectivity profile. How the expansion process of selfish mining pool in the network affects its communication capability and overall consensus security is also an important issue but overlooked in the literature.

Recognizing the lack of quantitative analysis on the security of distributed consensus from a network perspective, in this work we fill the gap by proposing a novel analytical model to assess the impact of network connectivity on the security of PoW-based blockchain consensus systems. An overview of our analytical model is given in Figure 3.1. The model captures network connectivity by a graph representation of the peer-to-peer network, and evaluates each node's communication capability from its network location and the adversary setting. The communication capability measures, combined with the consensus protocol specification and two other digests namely the information propagation dynamics and mining power distribution, are then used to quantitatively evaluate the security properties of the

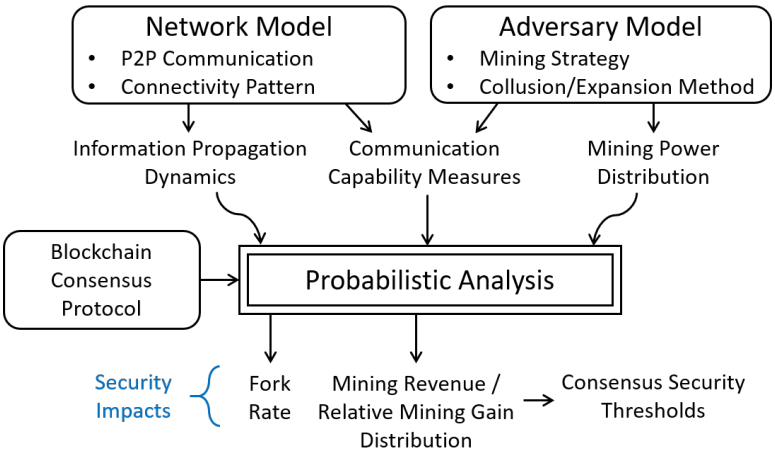


Figure 3.1: Analytical model for consensus security.

blockchain system.

The main contributions of this work include:

- We propose a novel analytical model to assess the impact of network connectivity on consensus security of PoW blockchain. In a nutshell, the model quantifies the communication capability of nodes involved in a fork race and derives the distribution of mining revenue, which is used for evaluating blockchain consensus security.

- We apply the analytical model to two adversarial scenarios, namely honest-but-potentially-colluding and selfish mining, and demonstrate that the lack of or excessively heterogeneous network connectivity leads to poor consensus security.

- We performed a thorough simulation experiment on PoW blockchain for each adversarial scenario. The simulation result matches the model prediction and validates our security analysis.

3.2 Background and Related Work

3.2.1 PoW Blockchain and Distributed Consensus

In public blockchain systems exemplified by Bitcoin, all networked miner nodes ("nodes" hereafter) work to curate a unified transaction history through distributed consensus. The transaction history is recorded in a chain of blocks in which every block contains a certain number of recently produced transactions. Every node seeks to generate the next block of the blockchain via a proof-of-work (PoW) process, namely, by finding an input to a cryptographic hash function that yields an output less than a target value. The input (i.e. the "proof") is attached in the block header. New blocks are disseminated immediately to the network via peer-to-peer gossiping. All nodes reach consensus on only one block at each blockchain height according to the "longest-chain rule": choosing the chain with the highest valid block. Generation of the next block is aimed at prolonging the longest chain, which shall always contain the most computational effort. Theoretically, as long as the majority computing power is controlled by honest nodes, the distributed consensus achieves probabilistic finality in that an accepted block could be discarded but with exponentially diminishing probability as the blockchain grows [14]. The above scheme is also known as Nakamoto Consensus, for its origin in the Bitcoin white paper [151].

In practical blockchain network, consensus security is complicated by blockchain forks. Blockchain fork is a scenario that multiple blocks of the same height are propagating in the network simultaneously. Under the assumption that all nodes are honest and follow the consensus rules, blockchain fork is caused by block propagation delays in that node j may generate a competing block before being aware of the existence of node i's block of the same height. To resolve blockchain fork, the longest-chain rule dictates that whichever fork branch gets appended with a new block should be chosen; blocks in other branches are then

discarded. In the presence of forks, the honest-majority premise can still ensure consensus security, under an assumption that all competing blocks in a fork have the equal chance of being the winner [90].

3.2.2 Network Connectivity's Impact on Consensus Security

Due to heterogeneous connectivity of the underlying peer-to-peer network, the equal-chance fork winning assumption does not hold true. A well-connected node, say node i, tends to have superior communication capability that allows it to disseminate information faster than a less-connected node, say node j. If node i generates a new block, it takes a shorter time for node i to propagate this block across the whole network and the rest of the network has a lower chance of generating a competing block. If node j generates a competing block before node i's block reaches j, node i's communication advantage can still cause a larger share of the network to follow its block, which gives node i a higher chance of winning the fork eventually. As a result, in the long run, well-connected nodes yield higher mining revenue than what is expected from their share of computing power. This discrepancy between the long-term mining revenue and the actual computing power of a node implies the possibility that a group of well-connected nodes with a minority fraction of computing power can harvest more than 50% of gross mining revenue, which renders the security from the honest-majority premise vulnerable.

Besides exploiting naturally occurred forks, a well-connected adversary can achieve significantly higher mining gains by proactively creating forks. Selfish mining [82] is one prominent example. Unlike an honest miner who publishes new block immediately after generation, a selfish mining attacker withholds newly generated blocks in a private chain, and strategically releases the private chain to the network whenever he sees his private chain's lead over the

public chain decreases to a threshold. The blockchain forks caused by the attacker's strategic private chain releases nullify the mining effort of honest nodes and create opportunities for the attacker to profit from its communication advantage. The detailed selfish mining strategy and the communication advantage parameter γ_{SM} will be introduced in §3.5.

3.3 System Model

3.3.1 Network Model and Consensus Protocol

We consider a peer-to-peer network of N nodes represented by an undirected graph $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ and its adjacency matrix A. $\mathrm{A}_{ij} = 1$ indicates node i, j share a peer relationship and can communicate in one hop. The PoW process and consensus based on the longest-chain-rule are modeled as follows. To model the output randomness of the cryptographic hash function used for PoW, we assume each node i generates new blocks according to Poisson process of rate π_i per time unit δ. Block generation of the whole network is characterized by the merged Poisson process of rate $\pi = \sum_i \pi_i$. Our model does not adjust mining difficulty. We consider a fixed set of consensus participants with fixed block generation rate.

Once some node i generates $block_i(h)$ of blockchain height h, it disseminates $block_i(h)$ throughout the network via peer-to-peer gossiping. Other nodes decide on the acceptance of $block_i(h)$ according to the longest-chain rule. That is, if another node k sees $block_i(h)$ while its local blockchain has already accepted $block_j(h)$ from node j and $j \neq i$, it declares a fork at height h and stops propagating $block_i(h)$. Conversely, if another node l sees $block_i(h)$ before $block_j(h)$, it declares a fork at height h and stops propagating $block_j(h)$. Once the two competing blocks completely stop propagating and the network partitions into two factions each of which advocates one block, we call this situation a fork stalemate. And the two

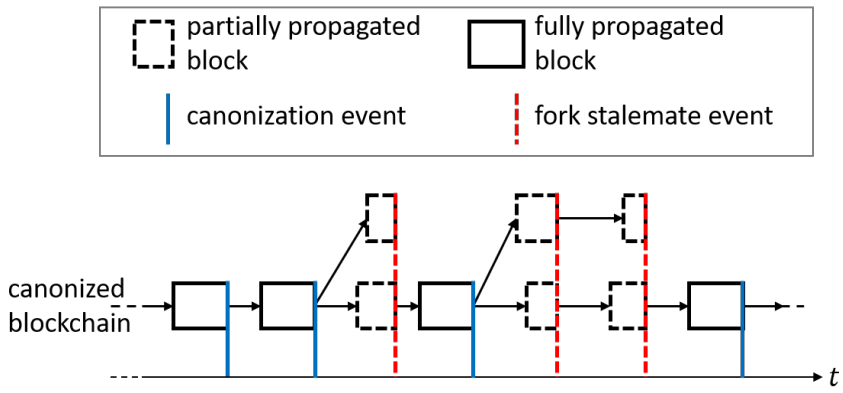


Figure 3.2: An example for blockchain canonization and fork stalemate events. Width of a block denotes its propagation period.

blocks are partially propagated. A fork stalemate can be resolved by a new block of height $h+1$ subscribing either $block_i(h)$ or $block_j(h)$ and being fully propagated in the network.

As for the finalization of blockchain, we consider the blockchain canonized by height h if a block of any origin $block_*(h)$ gets fully propagated in the network without encountering any competing block. We define the completion of $block_*(h)$'s propagation as a canonization event. Essentially, a canonization event at height h rejuvenates the PoW process as if the past forks and competitions never happened. Figure 3.2 shows an example for blockchain canonization and fork stalemate events. Note that the canonization concept is different from the probabilistic finality of Nakamoto Consensus, which considers consensus security a probabilistic measure. We will use canonization events as embedding points to estimate the mining revenue of each node in the §3.4.

3.3.2 Adversary Models

Honest-but-Potentially-Colluding. This adversarial scenario characterizes the practical case of the well-known 50% attack. All nodes operate honestly by default, but the top miners can potentially collude so that their combined mining revenue (MR) exceeds 50% of

Table 3.1: Summary of Notations

Network and Model Parameters	
G	The graph representation of the node network.
N	Number of nodes in G.
A	The adjacency matrix of G.
δ	Timeslot, also the time unit.
π	Block generation rate of the entire network (δ^{-1}).
π_i	Block generation rate of node i (δ^{-1}).
π_{SM}	Block generation rate of the selfish mining pool (δ^{-1})
Analyses	
$\mathrm{U}_i(t)$	Set of nodes unaware of node i's block at time t since its generation. $\lvert\mathrm{U}_i(t)\rvert$ is the cardinality of $\mathrm{U}_i(t)$.
$\lvert\mathrm{U}_i(t)\rvert_\pi$	Combined block generation rate of nodes in $\mathrm{U}_i(t)$.
$P_{NC,i}(t)$	Probability of the rest miners not proposing a competing block by time t of i's block's propagation.
$h(c)$	Blockchain height of the c^{th} canonization event
$\tau_{ij}(t)$	Minimum time for node i's block to reach j starting at time t from the generation of i's block.
$\omega_{i\succ j}(t)$	Node i's likelihood to win the fork race against node j if j publishes a competing block at time t from node i's block's generation. $\hat{\omega}_{i\succ j}(t)$ is an estimation.
γ_{SM}	Selfish mining pool's communication capability, i.e. the average fraction of honest mining power that will advocate the pool's block after it releases private chain.
MR_i	Mining revenue of node i as % of total canonized blocks.
RMG_i	Relative mining gain of node i. $RMG_i = \frac{MR_i - \pi_i/\pi}{\pi_i/\pi}$.
Security Metrics	
FR	Average fork rate of the whole network.
AT50	50%-attack threshold, i.e. minimum fraction of computing power controlled by adversarial nodes whose combined MR exceeds 50% of the total.
PRTH	Profitability threshold, i.e. fraction of computing power controlled by the selfish mining pool when it first achieves positive RMG during its expansion.

the total. In our analysis, a well-connected node may obtain positive relative mining gain (RMG) and collude with other well-connected nodes. The mining revenue of a node can be viewed as its "enhanced mining power" in contrast to its raw computing power. In this scenario we are interested in the 50%-attack threshold (AT50), i.e. the minimum fraction of computing power controlled by adversarial nodes whose combined MR exceeds 50% of the total.

Selfish Mining. This adversarial scenario assumes there are a pool of nodes in the network performing the selfish mining strategy described in [82]. We treat the selfish mining pool as a colluding consortium that expands among honest nodes. As the pool expands, it acquires the colluding nodes' computing power and external communication links. In this scenario, AT50 denotes the fraction of computing power controlled by the pool when the pool first achieves 50% of total MR during its expansion. We are also interested in the pool's profitability threshold (PRTH), which is the fraction of computing power controlled by the pool when it first achieves a positive RMG.

3.4 Analysis on Honest Mining

In this section we calculate the impact of network connectivity on blockchain fork rate and mining revenue distribution under the honest mining assumption. We then discuss the security provision under the honest-but-potentially-colluding adversarial scenario.

3.4.1 Fork Rate

Define M_i as the event that node i is the first to generate the next block at an arbitrary moment of no outstanding blockchain fork. Denote the time for node i to find a block by random variable T_i. Then $T_i \sim exponential(\pi_i)$ and

$$P(M_i) = P\{T_i < T_j, \forall j \neq i\} = \frac{\pi_i}{\pi} \tag{3.1}$$

which can be conveniently derived from properties of Poisson processes. To facilitate the ensuing analysis, we consider the physical time slotted into basic time units of δ.

Let the moment when event M_i happens be time 0. Denote $\mathrm{U}_i(t)$ the set of nodes unaware of node i's block at time t, and $|\mathrm{U}_i(t)|_\pi$ the combined block generation rate of $\mathrm{U}_i(t)$. We have $|\mathrm{U}_i(0)|_\pi = \pi - \pi_i$ and $|\mathrm{U}_i(t)|_\pi = 0$ when t exceeds the minimum time needed for i's block to reach all nodes. The probability that the rest of network does not generate a competing block by time t can be written as:

$$P_{NC,i}(t) = \prod_{s=\delta}^{t} \big(1 - |\mathrm{U}_i(s)|_\pi\big) \tag{3.2}$$

Since $(1 - |\mathrm{U}_i(s)|_M)_{s=\delta}^{t}$ is a positive increasing sequence bounded by 1, thus $(P_{NC,i}(t))_{t=\delta}^{\infty}$ is a convergent sequence.

Then by the law of total probability, the average blockchain fork rate of the whole network is obtained by weighing $(1 - \lim_{t\to\infty} P_{NC,i}(t))$ with $P(M_i), \forall i$:

$$\begin{aligned} FR &= \sum_i P(M_i)\big(1 - \lim_{t\to\infty} P_{NC,i}(t)\big) \\ &= \sum_i \frac{\pi_i}{\pi}\Big(1 - \prod_{s=\delta}^{\infty} \big(1 - |\mathrm{U}_i(s)|_\pi\big)\Big) \end{aligned} \tag{3.3}$$

If $\pi \ll 1$, N is large (e.g. π=1/600, $N\approx$10,000 in Bitcoin), mining power and network connectivity are evenly distributed, then $\forall i$ we have $\pi_i = \frac{\pi}{N}$, $|\mathrm{U}_i(s)|_\pi = \frac{\pi}{N}|\mathrm{U}_i(t)| = \pi \cdot \overline{ur}(t)$ wherein $\overline{ur}(t)$ denotes the average ratio of nodes uninformed of a new block at time t since its generation. Further assuming $\delta \to 0$, Eq. 3.3 reduces into the following form:

$$FR \approx 1 - \Big(1 - \pi\Big)^{\int_0^\infty \overline{ur}(t)dt} \tag{3.4}$$

which is consistent with the average fork rate obtained by [66]. The approximation $(1-ax) \approx (1-x)^a$ for small x is used.

3.4.2 Mining Revenue and Relative Mining Gain

Define a discrete-time random process $\{B_i(h)\}_{h=1,2,\dots}$ in which $B_i(h) = 1$ if node i is the block generator at height h in the canonized blockchain; 0 otherwise. The mining revenue MR_i and relative mining gain RMG_i of node i in the long term are defined as follows:

$$MR_i = \lim_{H\to\infty} \frac{1}{H} \sum_{h=1}^{H} B_i(h) \tag{3.5}$$

$$RMG_i = \frac{MR_i - \pi_i/\pi}{\pi_i/\pi} \tag{3.6}$$

Next we propose an estimation method for MR_i via probabilistic analysis. Define another discrete-time random process $\{W_i(c)\}_{c=1,2,\dots}$, which is embedded right after each blockchain canonization event. Therefore there is no outstanding fork nor propagating block in the network when random variables $W_i(c)|_{c=1,2,\dots}$ are evaluated. We further define $h(c)$ as the

blockchain height of the c^{th} canonization event and

$$W_i(c) = \begin{cases} 1 & \text{if } B_i(h(c)+1) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

Next we argue that the expectation of $W_i(c)$ at any epoch c, denoted $E[W_i]$, can be used to estimate MR_i in a conservative manner.

Theorem 3.1. $W_i(c)|_{c=1,2,...}$ are independent and identically distributed (i.i.d.) and their common expectation $E[W_i]$ satisfies the following relation with MR_i:

$$E[W_i] \begin{cases} \leq MR_i & \text{if } E[W_i] \geq \frac{\pi_i}{\pi} \\ > MR_i & \text{otherwise} \end{cases} \tag{3.8}$$

In other words, $E[W_i] - \frac{\pi_i}{\pi}$ is a conservative estimate of the mining gain/loss of node i. Moreover, the gap between $E[W_i]$ and MR_i tightens as the overall fork rate FR decreases.

Proof. Since $\{W_i(c)\}_{c=1,2,...}$ is embedded right after each blockchain canonization event when all previous forks are pruned and block propagation ceases, the competition for future blocks is oblivious of past block competitions. And block generation at each node is a memoryless process. Therefore, random variables $W_i(c)|_{c=1,2,...}$ are i.i.d and share a common expectation, denoted $E[W_i]$.

For an arbitrary canonization interval $c \to c+1$, we consider the blocks within it: those of height $h(c)+1, ..., h(c+1)$. First, $\frac{\pi_i}{\pi}$ evaluates the chance of i being the first to generate a block. $E[W_i] > \frac{\pi_i}{\pi}$ implies that i has a communication advantage over the network average which brings it positive mining gain. If $E[W_i] > \frac{\pi_i}{\pi}$ and i wins block $h(c)+1$, it will continue with an enhanced communication advantage for the current fork race and have a higher

chance of winning the subsequent blocks from $h(c)+2$ to $h(c+1)$. Conversely, if $E[W_i] < \frac{\pi_i}{\pi}$ and i does not win block $h(c)+1$, the chance for i to win any block from height $h(c)+2$ to $h(c+1)$ further decreases because of its aggravated communication disadvantage. Since $W_i(c)$ only considers the first block after canonization event c, using $E[W_i]$ to estimate MR_i implies that i would have an equal chance of winning any block from $h(c)+2$ to $h(c+1)$ just as winning $h(c)+1$. Therefore, $E[W_i]$ tends to underestimate (or overestimate) MR_i if $E[W_i] >$ (or $<$) $\frac{\pi_i}{\pi}$.

On the positive side, if the fork rate decreases, so is the number of blocks $h(c+1)-h(c)$ within the canonization interval $c \to c+1$. That is, there will be fewer blocks in a fork incident for $E[W_i]$ to under-/overestimate MR_i, and thus the former can achieve higher estimation accuracy. □

Next we derive $E[W_i]$. By the law of total expectation:

$$E[W_i] = P(M_i)E[W_i|M_i] + \sum_{j \neq i} P(M_j)E[W_i|M_j] \tag{3.9}$$

We separated the summation because the two conditional events $W_i|M_i$ and $W_i|M_j$ occur under different conditions.

$W_i|M_i$ consists of two subcases:

- No-fork win: No conflicting blocks are proposed by the rest of the network during the propagation of node i's block.
- Fork win: Conflicting blocks are proposed by the rest of the network during the propagation of node i's block, whereas node i's block still wins.

The probability of no-fork win equals to $P_{NC,i}(\infty)$, as is evaluated by Eq. (3.3). The

probability of fork win is slightly more complicated. During the propagation of node i's block, the number of conflicting blocks generated by the rest of network at time slot $(t, t+\delta]$ conforms to a Bernoulli distribution with rate $|\mathrm{U}_i(t)|_M$. If node j happens to generate a competing block during $(t, t+\delta]$, node i's block will need to win the support of the majority computing power of the network before it encounters node j's block in a stalemate. We denote the chance of node i winning the fork under this condition by $\omega_{i\succ j}(t)$. Therefore:

$$\begin{aligned} &E[W_i|M_i] \\ &= E[W_i,\ \text{no-fork win}|M_i] + E[W_i,\ \text{fork win}|M_i] \\ &= \lim_{t\to\infty} P_{NC,i}(t) + \sum_{t=\delta}^{\infty} P_{NC,i}(t) \sum_{j\in \mathrm{U}_i(t)} \pi_j \omega_{i\succ j}(t) \end{aligned} \tag{3.10}$$

Notably, in the derivation above we only considered two-prong forks for simplifying the analysis; likelihood of three or more-prong forks is negligible compared to that of two-prong forks.

In contrast, the conditional event $W_i|M_j$ in Eq. (3.9) can only happen via a fork race. That is, node i needs to generate a competing block during the propagation of node j's block, and eventually wins the fork race. Similarly to Eq. (3.10), we have:

$$\begin{aligned} &E[W_i|M_j] = E[W_i,\ \text{fork win}|M_j] \\ &= \sum_{t=\delta}^{\infty} P_{NC,j}(t)\pi_i \mathbb{1}_{\{i\in \mathrm{U}_j(t)\}}\big(1-\omega_{j\succ i}(t)\big) \end{aligned} \tag{3.11}$$

$\mathbb{1}_{\{i\in \mathrm{U}_j(t)\}}$ is an indicator function, returning 1 if the condition holds true; 0 otherwise. The winning chance of node i under this circumstance is $1-\omega_{j\succ i}(t)$.

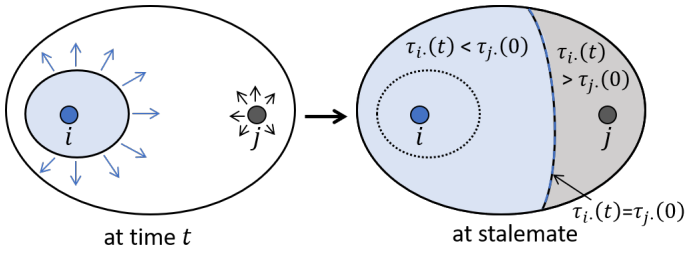


Figure 3.3: Explanation of Eq. (3.12). Light blue (grey) area denotes portion of the network that advocates i's (j's) block. $\hat{\omega}_{i\succ j}(t)$ is evaluated by the total computing power portion covered by light blue area at stalemate.

Evaluating $\omega_{i\succ j}(t)$. $\omega_{i\succ j}(t)$ essentially measures the communication advantage of node i over j when j generates a competing block which starts the fork race. For i to win the fork race against j, it has to have the majority of the network advocate its block before the two competing blocks end up in a stalemate. Let the moment when i publishes its block be time 0. Define $\tau_{ik}(t)$ as the minimum time for i's block to propagate to node k starting from time t. Then $\omega_{i\succ j}(t)$ can be estimated as follows:

$$\hat{\omega}_{i\succ j}(t) = \sum_k \frac{\pi_k}{\pi}\left(\mathbb{1}_{\{\tau_{ik}(t)<\tau_{jk}(0)\}} + \frac{1}{2}\mathbb{1}_{\{\tau_{ik}(t)=\tau_{jk}(0)\}}\right) \tag{3.12}$$

Figure 3.3 explains the calculation of $\hat{\omega}_{i\succ j}(t)$. As a result, we can finally obtain $E[W_i]$ by substituting Eq. (3.1), (3.10), (3.11), (3.12) into Eq. (3.9).

3.4.3 Security Analysis

We consider all nodes are honest-but-potentially-colluding. The fork rate FR provides an overall measure of how much mining power is wasted, while the 50%-attack threshold AT50 measures the system's security in the worst case scenario that the colluding group consists of the highest mining revenue earners. Next we analyze how network connectivity impacts FR and AT50.

Proposition 3.2. Lower overall network connectivity leads to higher FR and lower AT50, thus weaker consensus security.

Proof. We assume the block generation rate π_i is fixed for any node i. First, lower overall network connectivity means it takes longer for any node to disseminate a new block across the network. This can be caused by a protocol change that lowers the minimum peer number requirement. For the calculation in (3.2) (3.3), this leads to a higher $|\mathrm{U}_i(s)|_\pi$, lower $\lim_{t\to\infty} P_{NC,i}(t), \forall i$, and thus higher FR. Moreover, a lower $\lim_{t\to\infty} P_{NC,i}(t)$ means that more of MR_i is contributed by fork races and the distribution of mining revenue is deeper influenced by each node's communication capability. As a result, MR_i moves farther from $\frac{\pi_i}{\pi}$ and AT50 moves lower. □

Remark 3.3. For a certain network connectivity profile, higher block generation rate across all nodes (thus a higher π) would lead to a higher $|\mathrm{U}_i(s)|_\pi, \forall i$ and have the same impact of lower overall network connectivity.

Proposition 3.4. Higher heterogeneity of network connectivity also leads to lower AT50.

Proof. We still assume the block generation rate $\pi_i, \forall i$ is fixed. Higher heterogeneity of network connectivity means there is a greater divergence of communication capability among nodes. For instance, if node i resides in a highly-connected cluster in the network while node j resides in a sparsely-connected region, i will have a significant communication advantage over nodes in the sparse network region including j. As a result, i can disseminate a block to majority of the network much faster than j. $\hat{\omega}_{i\succ j}(t)$, as is evaluated by (3.12), will be close to 1 and $\hat{\omega}_{j\succ i}(t)$ will be much lower than 0.5. Therefore, i can harvest more mining revenue from fork races than j or other nodes in sparse network region. Consequently, $E[W_i]$ climbs higher above $\frac{\pi_i}{\pi}$ and $E[W_j]$ drops lower below $\frac{\pi_j}{\pi}$. This ultimately results in a more unequal MR distribution and hence lower AT50. □

3.5 Using Network Connectivity in Selfish Mining Analysis

In this section we evaluate the impact of network connectivity on selfish mining pool's communication capability and analyze its security implication under an expanding-consortium setting.

3.5.1 Selfish Mining Strategy

The core idea of selfish mining is to withhold newly generated blocks in a private chain, and release the private chain when the selfish mining pool sees the honest chain catch up close enough with the private chain. The detailed selfish mining strategy is illustrated in Figure 3.4, which we replicated from [82] and added with more description. Let α, β be the computing power share of the selfish mining pool and the honest nodes. Then $\alpha = \pi_{SM}/\pi$ and $\beta = 1 - \alpha$, where π_{SM} is the selfish mining pool's block generation rate. The state number denotes the private chain's lead over the honest chain. State transition is triggered by any block generation event. Transitions from state 1 to $0'$ and 2 to 0 are accompanied by the selfish mining pool's releasing the private chain. Any transition destined to state 0 marks a canonization event. γ_{SM} is defined as the long-term average fraction of honest computing power that will advocate the selfish mining pool's private chain when the pool and an honest miner release competing blocks simultaneously.

To incorporate network connectivity into the analysis, we model the selfish mining pool's network function as follows:

- Information exchanges within the selfish mining pool are instantaneous. The pool members are fully connected and synchronized. Any pool member who receives a new block from an honest node can make decision (changing state, switching chain,

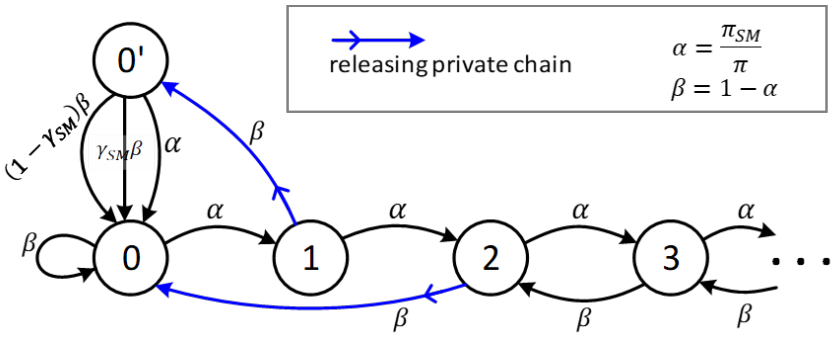


Figure 3.4: Selfish mining strategy in [82]. State number denotes the private chain's lead over the honest chain. State transition is triggered by block generation.

publishing the private chain) on behave of the entire pool.

- Once the selfish mining strategy determines to release the private chain, all pool members release the private chain to all peers simultaneously.
- Selfish mining pool members still relay blocks for honest miners, as long as the block does not trigger the pool to release its private chain. The reason is two-fold for the pool: to avoid suspicion of being a "blackhole" attacker, and to avoid network partitioning which would paralyze the blockchain system altogether.

Based on this model, we consider γ_{SM} the selfish mining pool's communication capability measure and evaluate it from the network connectivity profile.

3.5.2 Evaluating Communication Capability with Betweenness Centrality

Based on the assumption that nodes in the selfish mining pool are synchronous and can communicate with each other instantaneously, we treat these pool nodes as a fully-interconnected cluster and equivalently a super node denoted by smpool, which preserves the pool members' all external communication links to the remaining honest nodes. We show that a betweenness centrality measure of smpool within the network accurately evaluates γ_{SM}.

Theorem 3.5. γ_{SM} can be evaluated by the mining power weighted betweenness centrality measure of smpool:

$$\gamma_{SM} = \sum_{i \neq \text{smpool}} \frac{\pi_i}{\pi - \pi_{SM}} \sum_{j \neq i \neq \text{smpool}} \frac{\pi_j}{\pi - \pi_{SM} - \pi_i} \frac{\sigma(i, j|\text{smpool})}{\sigma(i, j)} \tag{3.13}$$

wherein $\sigma(i, j)$ is the number of shortest paths between i and j, and $\sigma(i, j|\text{smpool})$ is the number of such paths that pass through smpool.

Proof. Let i and j denote a pair of honest nodes, with i being the miner of a new block which triggers smpool to release its private chain according to the selfish-mine strategy. For j to switch to smpool's private chain instead of accepting i's new block, the highest block of smpool's private chain must be propagated to j before i's new block. In the graph model, this is necessitated by smpool residing on a shortest communication path between i and j. Therefore, $\frac{\sigma(i,j|\text{smpool})}{\sigma(i,j)}$ gives the likelihood that smpool delivers its private chain to j ahead of i's block. The weight $\frac{\pi_i}{\pi-\pi_{SM}} \frac{\pi_j}{\pi-\pi_{SM}-\pi_i}$ evaluates the pair (i, j)'s mining power contribution to γ_{SM} among all pairs of honest nodes. As a result, the mining power weighted betweenness centrality measure of smpool computes the average fraction of honest mining power that will advocate smpool's block after smpool releases its private chain, thus accurately evaluates γ_{SM}. □

Remark 3.6. Equation (3.13) can be conveniently computed with the Brandes algorithm [34]. If there are M honest nodes and they have equal mining power, i.e. $\pi_i = \frac{\pi-\pi_{SM}}{M}, \forall i \neq SM$, then the weight $\frac{\pi_i}{\pi-\pi_{SM}} \cdot \frac{\pi_j}{\pi-\pi_{SM}-\pi_i}$ becomes $\frac{1}{M(M-1)}$ and (3.13) reduces to the standard normalized betweenness centrality measure. With γ_{SM} obtained, the calculation of the selfish mining pool's mining revenue follows the procedure of [82]. Notably, the mining revenue of pool is proportional to γ_{SM}.

3.5.3 Security Analysis

We consider the selfish mining pool as an expanding consortium among the network of honest nodes. Under this setting, we discuss how network connectivity affects γ_{SM} and consensus security w.r.t. security thresholds AT50 and PRTH.

Proposition 3.7. Lower overall network connectivity leads to higher γ_{SM}, lower AT50, and lower PRTH, thus less secure against selfish mining.

Proof. Lower overall network connectivity leads to reduced communication capability of both the selfish mining pool and honest nodes. However, since the selfish mining pool consists of originally honest nodes and preserve all their external communication links, communication capability reduction of an average honest node will be more significant than that of smpool. Therefore, smpool will be residing in the shortest communication paths of more honest pairs if the overall network connectivity lowers, yielding a higher γ_{SM} for every α value. Consequently this yields lower AT50 and PRTH. □

Proposition 3.8. Compared to AT50, PRTH is more sensitive to network connectivity changes.

Proof. According to [82], PRTH is reached much earlier than AT50 for any γ_{SM}. Due to the gradualism of selfish mining pool's expansion, the rate of the selfish mining pool harvesting new external communication links initially increases, then gradually slows down as the pool takes in more nodes. Thus as α grows from 0 to 50%, γ_{SM} grows quickly at first then slow down as it comes closer to 1. Also the mining revenue of selfish mining pool is proportional to γ_{SM}. Therefore, a moderate reduction of overall network connectivity would lead to significant decrease in PRTH, but limited decrease in AT50. □

Remark 3.9. PRTH is also a more practical security measure than AT50 in the sense that once the pool size hits PRTH, joining the pool will be financially attractive to the remaining

Table 3.2: Honest Mining Experiment Result Corresponding to Figure 3.6

	Configuration		Metrics				
	Graph(N, D)	π	FR-SIM	FR-ANA	AT50-SIM	AT50-ANA	RMSE
a	$G_E(1000, 4)$	0.1	0.3148	0.3136	459/1000	470/1000	0.0325
b	$G_E(1000, 4)$	0.05	0.1773	0.1670	478/1000	484/1000	0.0205
c	$G_E(1000, 8)$	0.1	0.2409	0.2248	475/1000	479/1000	0.0187
d	$G_E(1000, 8)$	0.05	0.1315	0.1159	487/1000	489/1000	0.0123
e	$G_R(1000, 4)_{F10}$	0.1	0.3117	0.3099	457/1000	470/1000	0.0423
f	$G_R(1000, 4)_{F10}$	0.05	0.1758	0.1649	479/1000	485/1000	0.0232
g	$G_R(1000, 8)_{F10}$	0.1	0.2309	0.2124	480/1000	484/1000	0.0195
h	$G_R(1000, 8)_{F10}$	0.05	0.1250	0.1090	490/1000	491/1000	0.0113

honest nodes in the network.

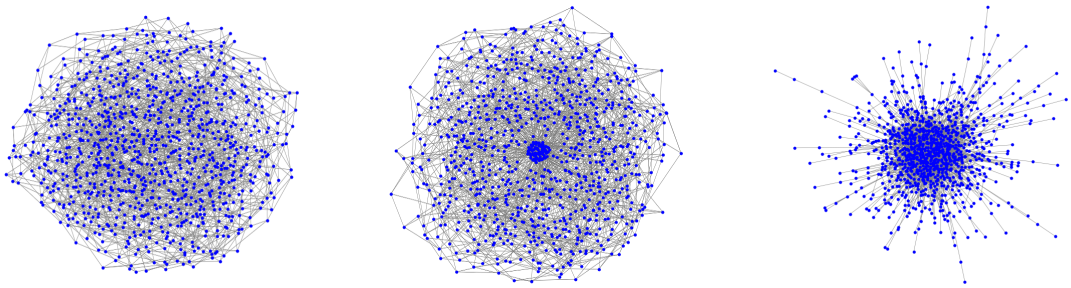
Remark 3.10. Selfish mining pool can take advantage of heterogeneity of network connectivity to achieve lower AT50 and PRTH. If the selfish mining pool is aware of the peer-to-peer network's topology, it can prioritize its expansion into well-connected regions of the network to maximize the growth of γ_{SM} and its mining revenue. As a result, heterogeneity of network connectivity can be exploited by selfish mining pool to achieve lower AT50 and PRTH.

3.6 Evaluation

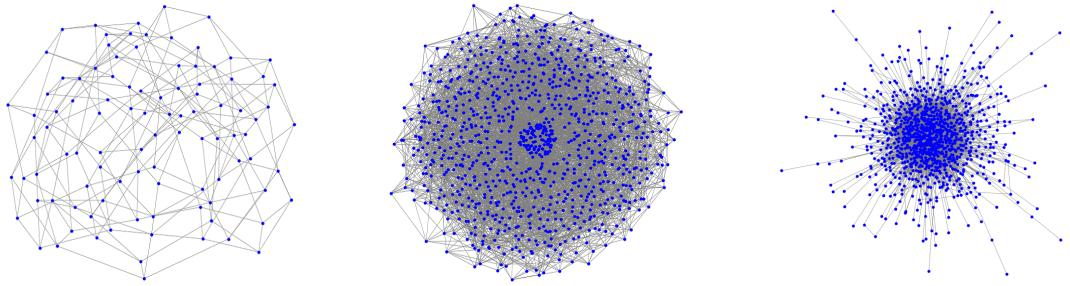
We conducted simulation experiments to validate our model and security analysis. The simulation program was written in Python and follows a time-driven fashion and takes the following as input: graph representation of the network, block generation rates of all nodes, adversarial setting (honest or selfish mining), and simulation time (slots).

3.6.1 Setup

We use three types of network graph for evaluation with the following notations:



(a) $G_R(1000, 4)$ (b) $G_R(1000, 4)_{F10}$ (c) $G_E(1000, 4)$



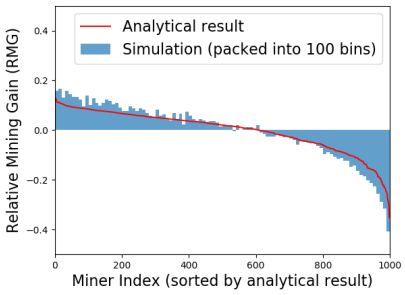
(d) $G_R(100, 4)$ (e) $G_R(1000, 8)_{F10}$ (f) $G_E(1000, 8)$

Figure 3.5: Visualization of six network graphs used in our experiments. Dot represents mining node, grey line segment represents communication link.

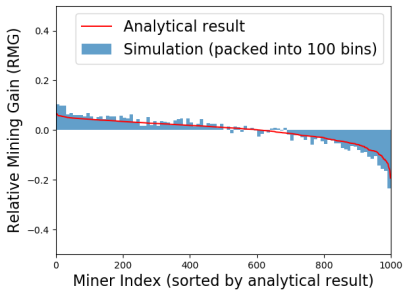
- $G_R(N, D)$: a regular graph with N nodes and degree D.
- $G_R(N, D)_{FX}$: a $G_R(N, D)$ but with the first $X\%$ of nodes being fully-interconnected.
- $G_E(N, D)$: a graph with N nodes and exponentially distributed node degrees with an average D.

The latter two graph types are designed to simulate different heterogeneous network connectivity profiles. The six network graphs used in our experiments are visualized in Figure 3.5 with Python package NetworkX [103]. To evaluate the impact of network connectivity and provide a fair ground for comparing security thresholds, we assign all nodes the same block generation rate: $\pi_i = \frac{\pi}{N}, \forall i$ while using π as a system input.

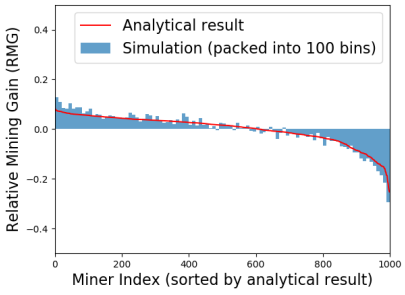
The following metrics are used for evaluation: FR, AT50 and PRTH as security metrics, rooted mean square error (RMSE) between analytical RMG distribution and simulated RMG



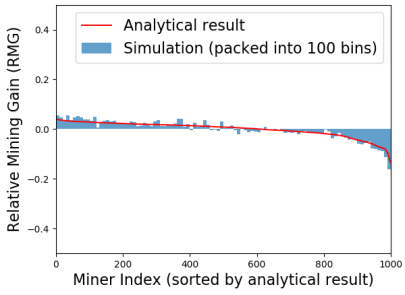
(a) $G_E(1000, 4), \pi = 0.1$



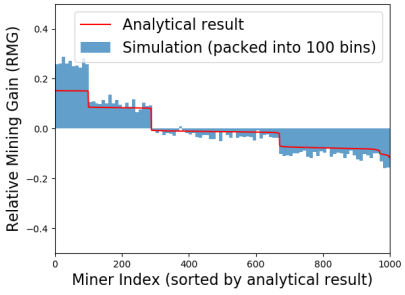
(b) $G_E(1000, 4), \pi = 0.05$



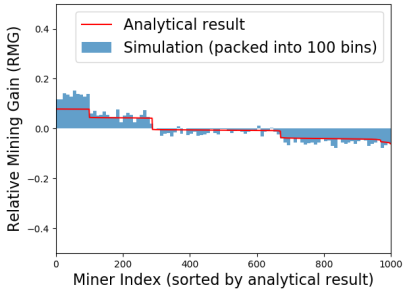
(c) $G_E(1000, 8), \pi = 0.1$



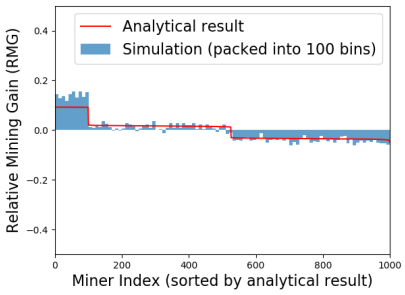
(d) $G_E(1000, 8), \pi = 0.05$



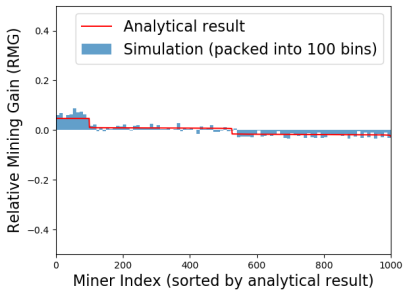
(e) $G_R(1000, 4)_{F10}, \pi = 0.1$



(f) $G_R(1000, 4)_{F10}, \pi = 0.05$



(g) $G_R(1000, 8)_{F10}, \pi = 0.1$



(h) $G_R(1000, 8)_{F10}, \pi = 0.05$

Figure 3.6: Relative mining gain (RMG) results of eight experiments.

distribution as the model accuracy metric.

3.6.2 Honest Mining Experiment

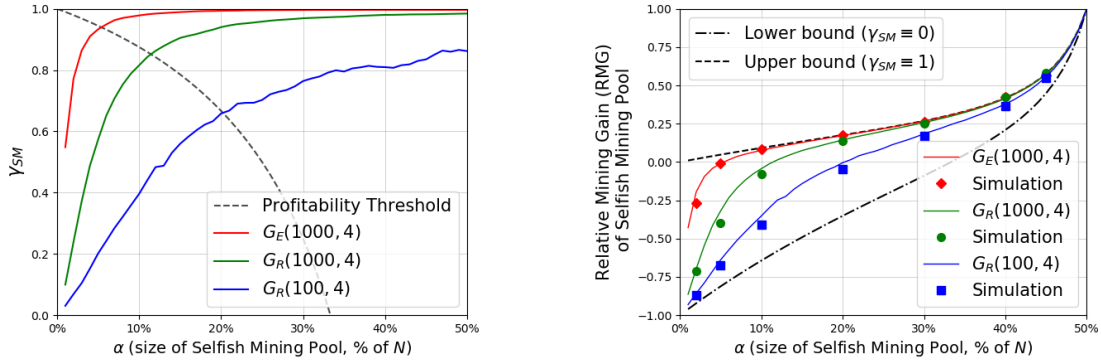
We performed eight experiments on four network graphs with different settings. Each experiment was run for 10 million timeslots. The configuration and evaluation results are shown in Table 3.2 and the RMG histograms are shown in Figure 3.6. We made the following observations:

- The analytical RMG result conservatively estimates the simulation result. The accuracy improves when π decreases or D increases. For instance, the obvious gap between analytical result and simulation in Figure 3.6e demonstrates the fully-interconnected top-10% nodes have a significantly higher block winning chance than that expected by $E[W_i]$. Nonetheless, as is shown in Table 3.2, for either graph type when π decreases from 0.1 to 0.05 or D increases from 4 to 8, the fork rate decreases and so does RMSE. This validates Proposition 3.1 that the estimation accuracy improves when fork rate drops.

- FR decreases and AT50 increases when π decreases or D increases. This validates the security analysis in 3.4.3 that higher overall network connectivity or lower block generation rate leads to stronger consensus security in the presence of potentially colluding nodes.

3.6.3 Selfish Mining Experiment

We switched the adversary setting from honest to selfish mining and performed three experiments. Each experiment targeted a certain network graph and consisted of seven simulations,



(a) Analytical γ_{SM} (b) Analytical RMG and simulation

Figure 3.7: Comparison of simulation and analytical result for selfish mining.

each took an α value from $\{2, 5, 10, 20, 30, 40, 45\}\%$ and ran for 10 million timeslots. Figure 3.7a shows the analytical result of γ_{SM}. The configuration and evaluation results are shown in Table 3.3 and Figure 3.7b. For graph $G_E(1000, 4)$ we configured the selfish mining pool to expand from the highest-degree node to lower-degree nodes in a descending order. This was purposed as an example of selfish mining pool's expansion strategy. RMSE here measures the averaged estimation accuracy of the analytical RMG over all the seven α values.

Table 3.3: Selfish Mining Experiment Result Corresponding to Figure 3.7

	Configuration		Metrics				
	Graph (N, D)	π	PRTH-SIM	PRTH-ANA	AT50-SIM	AT50-ANA	RMSE
1	$G_E(1000, 4)$	0.01	52/1000	55/1000	369/1000	369/1000	0.0290
2	$G_R(1000, 4)$	0.01	122/1000	114/1000	368/1000	370/1000	0.0338
3	$G_R(100, 4)$	0.01	22/100	21/100	38/100	38/100	0.0311

To estimate the thresholds AT50 and PRTH, for each of the three experiments we fitted the simulated RMG points using degree-7 polynomials and obtained AT50 and PRTH using the fitted curve. The following observations are made:

- The close match between analytical RMG results and simulation results in Figure 3.7b validates our betweenness centrality-based calculation of γ_{SM}.

- When N decreases from 1000 to 100, PRTH changes more dramatically than AT50. This validates our security analysis that PRTH is more sensitive to network connectivity changes. Particularly, the analytical curves of γ_{SM} in Figure 3.7a demonstrates that as the selfish mining pool size expands from $\alpha = 0$ to $\alpha = 50\%$, γ_{SM} grows quickly at first then gradually slows down when it comes closer to 1.

- As is shown in Figure 3.7a, γ_{SM} rapidly crosses above the PRTH curve and grows close to 1 when the selfish mining pool expands in $G_E(1000, 4)$. This demonstrates the feasibility for selfish mining pool to choose a particular expansion strategy to exploit the heterogeneity of network connectivity for faster revenue growth.

3.7 Discussion

On Potential Model Deficiency. In our model we only consider two-prong forks. That is, at most two blocks of the same height could be propagating in the network concurrently. Though the possibility of three-prong forks or more is significantly lower than two-prong forks, it could still affect the long-term mining revenue distribution, especially when the network is large and block propagation delays are high. To tackle this we would need a more delicate model that accounts the progress of all competing branches in a fork.

On Model Practicality. In practical blockchain networks, it can be challenging to monitor the block propagation progress (i.e. $U_i(t)$). To overcome this difficulty, a block propagation progress-agnostic model is needed to estimate the communication capability and forecast the revenue of a node via congregate network statistics. Furthermore, the permissionless network is structurally volatile and may conform to a scale-free expansion pattern [16]. It is possible to model structural changes of the network with a certain random process and

evaluate its impact on information propagation and consensus security. The impact of a selfish mining pool's internal communication routine is also a potential issue to explore.

3.8 Conclusion

We presented a modeling study on the impact of network connectivity on consensus security of PoW blockchain under two adversarial scenarios, namely honest-but-potentially-colluding and selfish mining. For the first scenario, we demonstrated the communication advantage of a node over its competitors in a fork race and provided a method to estimate its long-term mining revenue and relative mining gain. For the second scenario, we introduced a practical model for the selfish mining pool's network functions and showed that the communication capability of selfish mining pool, γ_{SM}, can be accurately evaluated by the mining power-weighted betweenness centrality measure. For both scenarios, we showed that low network connectivity and excessive heterogeneity of network connectivity lead to poor consensus security. Our modeling and analysis can serve as a paradigm for assessing the security impact of various factors in a distributed consensus system. In future work we will incorporate more realistic network settings into our model and extend the analysis to other blockchain systems.

Chapter 4

A Decentralized and Truth Discovering Data Feed for Blockchain DApps

(Copyright notice[1])

4.1 Introduction

Decentralized applications (DApps) are known for enabling autonomous and mutually distrustful parties to execute business logic without relying on a trusted intermediary or central authority. DApps are popularized by distributed ledger and blockchain technologies, with prime examples being the Ethereum smart contract [227] and Steemit social network [199]. A typical DApp runs atop a blockchain or distributed ledger system which provides networking utilities and instantiates a virtual operating system with a global state machine, whose consistency across decentralized participants is guaranteed by the native consensus mechanism. Take Ethereum for example, participants create or invoke a smart contract of certain DApp logic through blockchain transactions, the execution of which leads to changes in the contract state finalized by Ethereum's blockchain consensus.

When a DApp's business pertains to the real world, which is the most compelling case,

[1]This chapter was in submission to a conference for possible publication as of May 9, 2022. Reprinted/adapted, with permission, from Yang Xiao, Ning Zhang, Wenjing Lou, Y. Thomas Hou, "DecenTruth: A Decentralized and Truth Discovering Data Feed for Blockchain DApps," May 2022.

it needs to feed on data from external sources. In decentralized finance (DeFi) for instance, an investment DApp needs to periodically quote asset prices from different market sources to calculate an investment strategy; a DApp for mutual insurance needs recent information on the insured persons/objects to settle insurance claims [35]. The mechanism to fulfill such transportation of external data to DApp is known as data feed (or oracle when instantiated as a third-party service). Data from external sources, unlike the transactional data which are self-sufficient in the blockchain ledger, are beyond the jurisdiction of blockchain consensus. Meanwhile, due to the decentralized nature of DApp, it can be difficult to convince the participants to trust a third-party authority for vetting the external data or the trustworthiness of data sources. Such lack of trust on external data, known as the external data challenge, is often seen as a major hindrance to the wider adoption of DApps.

Existing solutions have tackled the external data challenge from the source authenticity (i.e., data are from legitimate sources) and data integrity (i.e., no tampering during transmission) perspectives [20, 35, 100, 244, 245]. These solutions generally involves building a secure channel between a DApp and an external data source to ensure secure data transmission after a successful mutual authentication. They follow either the third-party model which provides authenticated oracle service to DApp clients [35, 244, 245] or the first-party model which allows DApp participants to directly collect inputs from sources and commit to the DApp [20, 100]. These solutions generally assume individual external sources will provide truthful data inputs (i.e., being close or equal to the ground truths). The fundamental challenge of source trustworthiness remains unsolved, leading to the outstanding challenge on external data truthfulness.

4.1.1 The Data Truthfulness Challenge

An external data source, though being a legitimate entity, may submit fraudulent or deceitful data to a DApp's data feed mechanism. When a DApp feeds on data from multiple external sources, the sources may provide conflicting responses on the same object. For DApps that rely on external data for making critical decisions (e.g., DeFi apps), such problematic data can bring catastrophic outcomes to the participants. Philosophically, trusting individual external sources for proving truthful data contradicts DApp's defining principle of decentralization, as such trust imposes a single-point risk in the data plane. Unfortunately, it cannot be addressed by any internal consistency measure or data authenticity/integrity mechanism alone.

In order to solve the data truthfulness challenge, a data feed system entails data-oriented solutions that can extract the truthful information (as close to the ground truths as possible) from multi-sourced data with varying quality and be resilient against adversely affected sources or system participants. Truth discovery (TD), emerged a data mining line of research [130, 137, 238], poses an ideal data-oriented solution that can jointly estimate the ground truths of data objects and source reliability degrees from potentially conflicting multi-sourced inputs. Unreliable sources are assigned low reliability degrees that will penalize them in the weighted aggregation step. Although existing blockchain data feed systems [3, 35] do incorporate lightweight aggregation mechanisms such as majority/median voting, they are not designed to achieve resilience against adversarial sources. Using more complex methods such as TD would incur prohibitive cost if deployed on-chain (e.g., inside an Ethereum smart contract). Meanwhile, TD as an established data mining method generally assumes the algorithm is executed (or eventually aggregated) by one trusted server. To instantiate TD for DApp data feeds, novel methods are needed to decentralize the TD workflow and react to potentially malicious sources.

4.1.2 A New Data Feed Model

Eyeing on the potential of TD in addressing the data truthfulness challenge, we propose the decentralized truth discovering data feed model to enable a DApp to reliably procure truthful data from multiple external sources without introducing a central point of trust. This model builds upon a network of data feed nodes, which can be either directly instantiated by a consortium of DApp participants (per the first-party model) or incorporated into the existing commercial oracles (per the third-party model). The nodes connect to each other to form an off-chain network overlaying on the existing connectivity (e.g., blockchain network). Each node collects inputs on a common list of DApp data objects from its external sources. The nodes go through an interactive process to agree on the same values for the data objects on a batch basis. These values, called the truth estimates, are committed as data feed to the DApp by each node. The model's off-chain approach essentially facilitates the usage of complex data-plane algorithms, such as TD, without adding to the DApp's on-chain cost.

For producing the same truth estimates across all nodes with high accuracy (i.e., close to the ground truths) while preserving decentralization, the model entails the combined use of data-plane and consensus mechanisms to solve following challenges. First, sources may arbitrarily deviate their inputs from the ground truths, due to abnormality or adversarial influence which can also be adaptive. Second, the off-chain network of nodes inherits the threat profile from the corresponding decentralized DApp participants (or commercial oracles), which in the worst case may operate in an asynchronous, Byzantine-ridden situation—communication between nodes is subject to indefinite delay and some nodes may turn Byzantine and alter their communicated messages arbitrarily.

4.1.3 DecenTruth

We introduce the DecenTruth architecture to instantiate the decentralized truth discovering data feed model and address the above challenges. DecenTruth harmonizes techniques from two lines of research: TD and asynchronous Byzantine fault tolerant (BFT) consensus. We take inspiration from the research of online incremental TD [163, 219, 243] and design a novel composite batch incremental TD process (CBI-TD) as the data-plane solution. Nodes are able to consistently produce truth estimates from a common subset of local truth proposals for every batch of objects, and perform reliability tracking on their local sources and peer nodes for achieving Byzantine resilience. We devise a consensus protocol called weight-prioritized agreement on a common subset (WP-ACS) that enables nodes to propose its local truth estimates and jointly decide on the aforementioned common subset of proposals amid the harsh asynchronous network condition. Priority is given to the proposals from nodes with higher historical weights computed by CBI-TD. The combination of the two techniques realizes a decentralized data feed with strong guarantees on fault tolerance, resilience, and data-plane accuracy.

In summary, we make the following contributions:

- Observing the external data truthfulness challenge facing DApps and the deficiency of existing solutions, we formulate a decentralized truth discovering data feed model to allow a DApp to obtain truthful data from inputs provided by potentially untrustworthy external sources, while preserving the decentralization property.
- We introduce the DecenTruth architecture to realize the aforementioned model while addressing practical challenges in security and resilience. DecenTruth is comprised of two carefully designed components, CBI-TD and WP-ACS, realizing online incremental truth discovery and asynchronous BFT consensus.

- We show that DecenTruth achieves Byzantine resilience under a practical adversary model, including adaptive Byzantine corruptions on sources and nodes and existence of an adversarial network scheduler.

- We implemented DecenTruth and evaluated its performance in an emulated DApp data feed scenario. The result shows that our system presents a practical approach towards the external data truthfulness challenge with effective Byzantine resilience and long-term estimation accuracy comparable to that of an ideally centralized TD method given a certain level of source participation.

4.2 Background and Related Work

4.2.1 Solutions to the External Data Challenge

Existing work has provided data feed mechanisms to address the authenticity and integrity part of the external data challenge, which predominantly involves building a secure channel between DApps and external data sources. Town Crier [244] is a third-party data feed service for Ethereum smart contract. It builds on a smart-contract front end and a trusted computing back end, realizing a secure channel for transferring data from external HTTPS-enabled websites to client smart contracts. Proofs of data authenticity and channel integrity are facilitated by the remote attestation functionality of Intel SGX, a trusted computing platform. DECO [245] realizes the authenticated data feed functionality similar to that of Town Crier but without trusted computing hardware. It is built upon the participation of independent oracles and zero-knowledge proofs after a multiparty authentication process. The third-party model can also be realized in a distributed fashion, where data collectors (i.e., oracles) form an oracle network and provide data feed service to DApps collectively. Chainlink [35] is

currently the most popular commercial oracle service provider that comprises of a list of reputed oracle nodes and incorporates Town Crier and DECO functionalities. BandChain [174] is another popular commercial oracle service for cross-chain data. Instead of building a third-party data feed service, PDFS [100] and API3 Airnode [20] instantiate the data feed function in each DApp participant. API3 Airnode stipulates that data reliability can be guaranteed by its derivative business logic, for instance, an insurance service to DApp clients on the quality of data provided by external sources. This first-party approach incorporates data sources directly into the DApp ecosystem and saves the need of bootstrapping trust on commercial oracles.

Solving the data truthfulness challenge, however, is beyond the capability of secure channel solutions and needs data-plane mechanisms. In current data feed solutions, Astraea [3] and [44] use smart contract to realize an on-chain stake-and-vote mechanism, where data feed nodes vote a binary decision on external data items; Chainlink [35] and BandChain [174] also use lightweight on-chain aggregation mechanisms on multi-sourced data, such as taking the average or median. These on-chain mechanisms tend to bring computation workload linear to the size of data objects to entire blockchain platform. Usage of more complex aggregation algorithms is cautioned due to the prohibitive on-chain cost (e.g., Ethereum smart contract).

4.2.2 Truth Discovery

The truth discovery (TD) paradigm aims for uncovering the ground truths of data objects from noisy and potentially conflicting inputs provided by different sources [130]. TD aims to jointly estimate the ground truths of objects and the reliability degrees of sources so that the estimated truths approximate the unknown ground truths as close as possible. Sources

that consistently provide inputs close to the ground truths will be hopeful assigned high reliability degrees, which in turn contribute to an accurate estimate of the truths.

When we instantiate TD for DApp data feed, early TD formulations cannot be employed directly since they predominantly work on a static and monolithic dataset and tend to overlook the evolving adversarial attributes in the system such as Byzantine behaviors of sources. Moreover, the entire TD task is usually performed by one trusted central server, which does not fit into DApp's decentralized low-trust model.

Online TD on Streaming Data. This line of research aims to instantiate TD on streaming input in a data-driven fashion, in which data are continuously generated by sources and fed to the TD algorithm. They commonly adopt lightweight mechanisms to handle streaming inputs in an online or recursive fashion [215, 252]. Li et al. [129] propose an online incremental TD scheme which is able to estimate the truths and source reliability degrees with consistent accuracy when reliability of sources evolve over time. These online TD solutions provide valuable lessons on achieving good estimation accuracy while preserving computation efficiency.

Distributed TD. This line of research aims to scale TD to larger data volume and source diversity. They provide TD formulations under a distributed setting in that truths are first discovered locally and then aggregated at the central server. To accommodate large data volumes in quantitative crowdsourcing tasks, Ouyang et al. [163] propose to decompose the original TD problem into several small-scale tasks that can be run in parallel before aggregation. Zhang et al. [243] design a heuristic distributed TD approach to deal with data sparsity, misinformation, and efficiency issues. Wang et al. [219] extend the TD problem to a two-stage distributed setting, wherein local TD servers handle local data sources while a

central server aggregate the local results. These parallel and distributed TD solutions still rely on a central server to perform task allocation and final truths aggregation. Realizing TD in an environment with no trusted central authority remains a fresh topic. Tian et al. [207] instantiate a TD mechanism using Ethereum smart contract. Fu et al. [89] propose a decentralized TD formulation based on maximum likelihood estimation and P2P gossiping; the inter-node consistency/agreement and adversarial influence on nodes are yet to be considered.

In our system, we adopt the online incremental TD approach using higher-order statistical metrics for capturing the presence of Byzantine sources. We also take inspiration from distributed TD in that inputs from data sources are first processed locally, reducing the computation workload at the global step.

4.3 Building Blocks

4.3.1 The Baseline TD Algorithm

Given a list of objects $\mathcal{O}$ and a collection of inputs $\{x_{s,o}\}_{s\in\mathcal{S},o\in\mathcal{O}}$ from source group $\mathcal{S}$ on each object in $\mathcal{O}$, TD aims to jointly estimate the truths behind the objects $\{\hat{x}_o\}_{o\in\mathcal{O}}$ and the reliability degrees of sources $\{r_s\}_{s\in\mathcal{S}}$ so that the estimates approximate the ground truths as close as possible. Under this principle, existing TD formulations have nuanced assumptions on issues including input data generation model, data format, source reliability consistency and dependence, as well as correlation between objects [130].

Prior wisdom has provided different mathematical formulations for the TD problem. The solutions commonly resemble an iterative procedure, in which truth aggregation (Eq. 4.1) and the source reliability degree estimation (Eq. 4.2) take place alternately until a certain

convergence criterion is met.

$$\hat{x}_o = \frac{\sum_{s \in \mathcal{S}} \mathbb{1}_{s,o} r_s x_{s,o}}{\sum_{s \in \mathcal{S}} \mathbb{1}_{s,o} r_s} \qquad \forall o \in \mathcal{O} \tag{4.1}$$

$$r_s = g\left(\sum_{o \in \mathcal{O}} d(x_{s,o}, \hat{x}_o) \right) \qquad \forall s \in \mathcal{S} \tag{4.2}$$

We define $\mathbb{1}_{s,o}$ as the indicator that returns 1 if source s provided input on object o (0 otherwise). $d(\cdot, \cdot)$ is a distance measure between input $x_{s,o}$ and the current truth estimate $\hat{x}_o$. $g(\cdot)$ is a monotonically decreasing function. Choices on $d(\cdot, \cdot)$ and $g(\cdot)$ vary among different solutions. Essentially, having source reliability in the estimation loop allows the TD algorithm to capture the consistency of a source's inputs on all objects and assign a higher reliability degree to sources with consistently accurate inputs.

4.3.2 BFT Consensus and ACS Protocol

The Byzantine fault tolerant consensus problem has been extensively studied in the distributed computing community. In the simplest form, it describes a network of N nodes working to consent on a common value, while up to F nodes may behave maliciously by sending arbitrary and conflicting values to other nodes [122]. This type of malicious behaviors is known as Byzantine fault. As is discussed in §2.2 of Chapter 2, this consensus goal is reachable only if $N \geq 3F + 1$ [168].

ACS Protocol. We are interested in one special type of BFT consensus problem called agreement on a common subset (ACS) which was first formulated in [19]. Here we provide a recap on ACS which is also introduced in §2.2.5 of Chapter 2. Asynchrony refers to the condition that messages within the network can have arbitrarily long delays, though the

eventual delivery is guaranteed. In such a network of N nodes, up to F nodes can be Byzantine and each node n has a proposal $\mathcal{P}_n$. The goal of ACS is to allow all correct nodes to agree on a common subset of proposals, denoted CSP. An ACS protocol can be composed from two BFT sub-protocols, namely reliable broadcast (RBC) and binary agreement (BA).

RBC is a consensus primitive that allows a node to safely disseminate its proposal to peer nodes [31]. When node n starts instance RBC[n] with proposal $\mathcal{P}_n$, the following properties are guaranteed: 1) Agreement: If any two correct nodes deliver $\mathcal{P}$ and $\mathcal{P}'$, then $\mathcal{P} = \mathcal{P}'$; 2) Validity: If the leader n is correct, then all correct nodes will eventually deliver $\mathcal{P}_n$.

BA is a lightweight consensus primitive that allows nodes to agree on a binary value $\in \{0, 1\}$. BA achieves following properties: 1) Termination: If all correct nodes receive input, then each of them should end up with a decision. 2) Agreement: If any correct node decides b, then all other correct nodes will decide b. 3) Validity: If any correct node decides b, then b must be the input of at least one node. If all correct nodes have the same input b, then b must be the final decision. To achieve termination in an asynchronous network, BA needs to make random decisions at times when seeing conflicting or insufficient information. In known BA implementations [43, 149], a cryptographic scheme called common coin is used to provide such randomness. When at least $F + 1$ nodes execute the ComCoin(k) protocol, all nodes will receive the same coin toss result $coin_k \in \{0, 1\}$ for object k.

An ACS protocol's agreement, termination, and validity properties follow from its composing RBC and BA protocols. BA instances essentially help to determine a subset of RBC instances that eventually complete. In the end, at least $N - F$ BA instances will return 1 and the corresponding proposals are included in the common subset CSP. In §4.5.3 we will use RBC, BA, and ComCoin to compose our customized WP-ACS scheme, a key component of DecenTruth.

4.4 The Decentralized Truth Discovering Data Feed Model

To tackle the data truthfulness challenge, we introduce a decentralized truth discovering data feed model that allows a DApp to procure truthful data from untrusted external sources while preserving decentralization.

4.4.1 Network and Task Model

Consider a network of N nodes tasked for discovering the true values of common objects for a DApp. The nodes can be either instantiated by a committee of DApp participants (i.e., first-party) or incorporated into the existing third-party oracles. We assume there exist S external sources that provide data inputs on the objects to the nodes. Each node $n \in [N]$ ($[N] := \{1, ..., N\}$) has access to a subset of the data sources denoted $\mathcal{S}_n \subset [S]$. In other words, $\mathcal{S}_n$ represents the "local sources" that report to node n. Nodes may communicate with each other via asynchronous but authenticated channels. That is, messages between any two nodes are guaranteed eventual delivery, but message delays are unpredictable. This asynchrony assumption captures the practical case of autonomous participation, wherein nodes are hardly coordinated and their intercommunication is subject to arbitrary delays in the worst case. Message authenticity and integrity are provided by TLS communication with a pre-established PKI.

The task of the nodes is to reach an agreement on the same truth estimate for a growing list of data objects. This process is executed in epochs, with each epoch e committed to a batch of objects $\mathcal{B}_e$ with fixed batch size B. This batch configuration accommodates the scenario that the DApp needs to periodically take action upon every cycle of updates. Specifically for each epoch e, node $n \in [N]$ receives inputs from its local sources, denoted $\{x^n_{s,i}\}_{s \in \mathcal{S}_n, i \in \mathcal{B}_e}$. $x^n_{s,i}$ is null if source s does not provide input on object i. At the end of an

epoch, all nodes need to reach consensus on the same truth estimates $\{\hat{x}_i\}_{i\in\mathcal{B}_e}$ for this batch. We note that input data can be of any type. In this work we focus on continuous data for a consistent narrative.

We reuse the investment DApp example in §4.1 to illustrate the task model. The DApp quotes hourly closing prices of a portfolio of stocks from different market sources. Batch size B is the number of stock symbols and an epoch is one hour. The price of a certain symbol-hour is an object. The quote on a price-hour provided by a source is an input. By the end of each hour, all nodes agree on the same quotes for the stock symbols, which are then committed to the DApp.

4.4.2 Threat Model

We assume the adversary can corrupt up to f_s fraction of sources and F nodes at any time, with $f_s < 0.5$ and $3F + 1 \leq N$. The adversary can exert Byzantine influence on corrupted sources and nodes. In the data plane, Byzantine sources may provide arbitrary values within the input space on any object to their corresponding nodes. Similarly, Byzantine nodes may send arbitrary or even conflicting information to peer nodes. In line with the asynchronous network assumption, a strong adversary may also introduce arbitrary delays to the communication link between any nodes. We further assume that adversarial corruptions are adaptive in that the adversary can corrupt different subsets of sources and nodes throughout time. The corrupted sources and nodes may act innocuously at first but suddenly turn Byzantine at some points. Lastly, compared to their Byzantine counterparts, honest sources will provide inputs on all objects with consistent reliability, measured by the standard deviation from the ground truths. Honest nodes will strictly follow the predefined protocol and do not disseminate conflicting information.

When deployed in the first-party manner, incentives for DApp participation and the data feed task are naturally aligned since the DApp participants directly manage their own data feed nodes, making the Byzantine threshold assumption a practical choice. When deployed to the third-party oracles, extra incentives or trust mechanisms would be needed to regulate the oracles to ensure the efficacy of the Byzantine threshold assumption. Detailed discussion is provided in §4.9.

4.4.3 Requirements

Four requirements need to be met for a concrete instantiation of the decentralized truth discovering data feed model:

- Accuracy: The final truth estimate $\hat{x}_i$ on any object i is close to the ground truth, as would be produced by a centralized estimator (e.g., a traditional TD algorithm).
- Resilience: The accuracy requirement holds in the long term under the aforementioned threat model. In the short term, accuracy can quickly recover from the sudden deterioration caused by adaptive Byzantine influence.
- Termination: If every honest node receives an input on object i from local sources, then every honest node eventually outputs a final truth estimate on object i.
- Agreement: If any honest node outputs the final truth estimate $\hat{x}_i$ on object i, then every honest node outputs $\hat{x}_i$ on object i.

The accuracy and resilience requirements reflect the data-plane performance. The termination and agreement requirements necessitate the eventual completion and consensus of the truth estimates on every batch of objects.

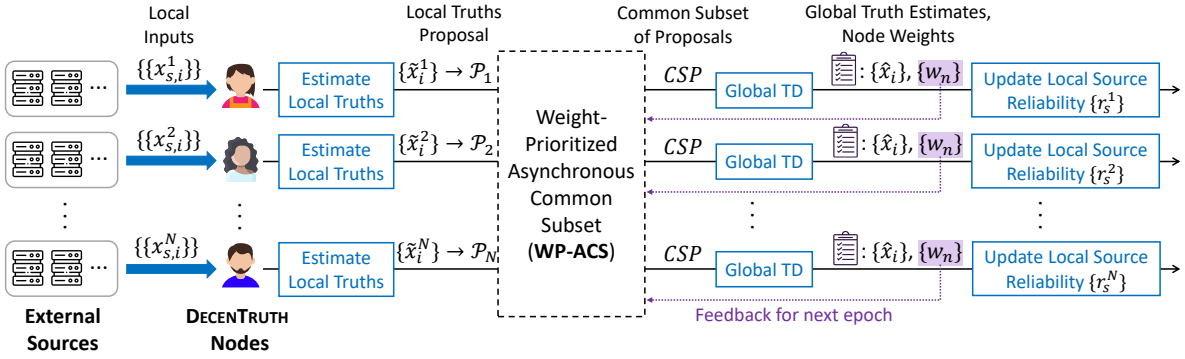


Figure 4.1: DecenTruth workflow for one epoch. Data-plane operations (i.e., CBI-TD) are highlighted in blue.

4.5 DecenTruth

4.5.1 Overview

We introduce the DecenTruth architecture to instantiate the decentralized truth discovering data feed model. Fig. 4.1 illustrates the DecenTruth workflow for one epoch. After receiving local inputs for the current epoch e, each node n computes a local estimate $\tilde{x}_i^n$ for every object $i \in \mathcal{B}_e$. These estimates constitute node n's local proposal, denoted $\mathcal{P}_n$. After going through a decentralized consensus process called weight-prioritized agreement on a common subset (WP-ACS), all nodes decide a common subset of proposals denoted CSP, which is subsequently fed to the global TD algorithm. Here "global" means the algorithm and its hyperparameters are pre-determined in all nodes. The global TD at all honest nodes will return the same result, i.e., the estimated truths and the node weights. At the end of an epoch, each node updates the reliability degrees of its local sources based on the newly obtained truth estimates. The updated source reliability degrees will be used for local truth estimation in the next epoch. The local truths estimation, source reliability updating, and global TD constitute a composite batch incremental TD (CBI-TD) process that keeps track of source reliability degrees and node weights and make online decisions on global truth

estimation. Notably, in this work we focus on the off-chain implementation. Realization of the final data feed commitment to DApp is an independent task and we defer it to future extension. Next we elaborate on the two major components of DecenTruth, i.e., CBI-TD and WP-ACS.

4.5.2 Component 1: CBI-TD

Each node n maintains four types of internal variables: reliability degree r_s^n, error measure ϵ_s^n and consistency measure κ_s^n of each local source $s \in \mathcal{S}_n$, and node weight w_k of every peer node $k \in [N]$ (all notations are associated to the current epoch e unless otherwise specified). Two sub-tasks are executed: local incremental TD and global TD.

Local Incremental TD is a cross-epoch procedure executed by every node. It is responsible for generating proposals for the global TD and keeping track of reliability degrees of local sources. We remark that previous solutions have demonstrated formulating an optimization problem on historical data to realize incremental TD [129, 252]. In comparison, our scheme faces the unique challenge of Byzantine sources and needs a new reliability evaluation method for Byzantine resilience.

The local incremental TD works as follows. For epoch $e = 1, 2, ...$, node $n \in [N]$ proceeds as follows. After receiving inputs $\{x_{s,i}^n\}_{s \in \mathcal{S}_n, i \in \mathcal{B}_e}$ from local sources $\mathcal{S}_n$, it estimates the local truths using the local source reliability degrees $\{r_s^n\}$ from the last epoch:

$$\tilde{x}_i^n = \frac{\sum_{s \in \mathcal{S}_n} \mathbb{1}_{s,i} r_s^n x_{s,i}^n}{\sum_{s \in \mathcal{S}_n} \mathbb{1}_{s,i} r_s^n} \qquad \forall i \in \mathcal{B}_e \tag{4.3}$$

Then node n compiles its proposal $\mathcal{P}_n = \{\tilde{x}_i^n\}_{i \in \mathcal{B}_e}$ and provides $\mathcal{P}_n$ to the consensus process and the ensuing global TD. After the global TD delivers the estimated global truths

$\{\hat{x}_i\}_{i\in\mathcal{B}_e}$, node n computes an error measure ϵ_s^n with mean square error on objects that s has participated, and a consistency measure $\kappa_s^n \in (0,1]$:

$$\epsilon_s^n = \frac{\sum_{i\in\mathcal{B}_e} \mathbb{1}_{s,i}(x_{s,i}^n - \hat{x}_i)^2}{\sum_{i\in\mathcal{B}_e} \mathbb{1}_{s,i}} \tag{4.4}$$

$$\kappa_s^n = \text{erfc}(\beta|\epsilon_s^n - \mathring{\epsilon}_s^n|) \cdot (1-\alpha) + \mathring{\kappa}_s^n \cdot \alpha \tag{4.5}$$

wherein $\mathring{\epsilon}_s^n$ and $\mathring{\kappa}_s^n$ refer to the corresponding measures from the last epoch. erfc() is the complementary error function which is widely used for evaluating statistical accuracy. The intuition behind using erfc() here is that it is a monotonic decreasing function and provides a convenient output range $(0,1]$ for input range $[0,\infty)$. It sharply penalizes input increase when input is close to 0, which helps stage a swift response to Byzantine source inputs. The weighted moving average calculation of κ_s^n is aimed for space efficiency, as it only needs the most recent error and consistency measures. $\alpha \in [0,1)$ is a user-defined decay factor: lower α enables swift reaction to short-term Byzantine behaviors while higher α helps establish long-term judgement on Byzantine sources. $\beta > 0$ is the scale factor that depends on the range of input value. α and β are design choices during implementation.

Finally node n updates its local source reliability degrees:

$$r_s^n = \frac{\kappa_s^n}{\epsilon_s^n} \qquad \forall s \in \mathcal{S}_n \tag{4.6}$$

While the error measure ϵ_s^n penalizes source s for generally inaccurate inputs, the consistency measure κ_s^n (when close to 0) penalizes source s specifically for Byzantine influence. We provide detailed analysis in §4.6.1 on how κ_s^n facilitates the Byzantine resilience of our system.

Global TD is executed by every node to obtain the global truth estimates $\{\hat{x}_i\}_{i\in\mathcal{B}_e}$ for

epoch e after obtaining the common subset of proposals $CSP = \{\mathcal{P}_k | k \in CS\}$. CS is the corresponding subset of node IDs and $\mathcal{P}_k = \{\tilde{x}_i^k\}_{i \in \mathcal{B}_e}$. As the problem of Byzantine proposals is addressed by the preceding WP-ACS consensus (to elaborate in §4.5.3), we adopt the conventional optimization-based approach for global TD. With node weights $\{w_k\}_{\in CS}$ and proposed values $\{\tilde{x}_i^k\}_{k \in CS, i \in \mathcal{B}_e}$ we formulate the following optimization problem:

$$\min_{\{w_k\},\{\hat{x}_i\}} \sum_{k \in CS} \sum_{i \in \mathcal{B}_e} w_k d(\tilde{x}_i^k, \hat{x}_i) \quad \text{s.t.} \sum_{k \in CS} \log w_k = 1 \tag{4.7}$$

$d()$ can be any distance function, including the square error we used for local TD. We keep the form $d()$ for generality.

This problem can be solved by coordinate descent in an iterative manner. First, we fix the truths estimates $\{\hat{x}_i\}$ and apply Lagrange multipliers method to Eq. (4.7) to get the best estimate of the weights $\{w_k\}$. We omit the derivation and directly give the result:

$$w_k = \frac{c}{\sum_{i \in \mathcal{B}_e} d(\tilde{x}_i^k, \hat{x}_i)} \qquad \forall k \in CS \tag{4.8}$$

wherein c is a constant. Next, we fix the weights and aggregate the truths:

$$\hat{x}_i = \frac{\sum_{n \in CS} w_n \tilde{x}_i^n}{\sum_{n \in CS} w_n} \qquad \forall i \in \mathcal{B}_e \tag{4.9}$$

Eq. (4.8) and Eq. (4.9) are executed alternately until a convergence criterion is met. After that $\{\hat{x}_i\}_{\mathcal{B}_e}$ are committed to the DApp as the global truths Moreover, throughout the iterations we keep the weights of nodes outside CS, i.e., $\{w_z\}_{z \notin CS}$, unchanged for consistency. Thus one more step is needed to normalize the weights of those in CS:

$$w_k \leftarrow \frac{w_k}{\sum_{l \in CS} w_l} \Big(1 - \sum_{z \notin CS} w_z\Big) \qquad \forall k \in CS \tag{4.10}$$

4.5.3 Component 2: WP-ACS

WP-ACS is the consensus component that enables nodes to agree on a common set of proposals CSP in the asynchronous network condition before proceeding to global TD. WP-ACS is designed to take advantage of the global TD component of our system for improving the quality of its output. In specific, WP-ACS relies on global TD for feedback on node weights from the last epoch. Proposals from nodes with lower weights should have a reduced probability of being included in CSP. The ensuing global TD in turn can benefit from the improved quality of CSP. This feedback mechanism is essential for DecenTruth to penalize the proposals made by malfunctioning/malicious nodes.

Algorithm 8: The WP-ACS Protocol (at TD node n)

Variables: $\mathcal{L}$, $dSet \leftarrow \{\}$, $coin$
Input: Local proposal $\mathcal{P}_n$, node weights $\{w_k\}_{k\in[N]}$
Output: Common subset of local proposals CSP and the proposal indices CS

$\mathcal{L} \leftarrow \{l | w_l$ is among the top $N-F$ of $\{w_k\}_{k\in[N]}\}$
Start RBC$[n]$ with $\mathcal{P}_n$ as input
if receiving the delivery of $\mathcal{P}_k$ from RBC$[k]$ and BA$[k]$ has not been provided input then
    if if $k \in \mathcal{L}$ then
        Provide input 1 to BA$[k]$
    else
        $dSet \leftarrow dSet \cup \{k\}$
if having provided inputs to every BA$[n], \forall n \in \mathcal{L}$ and $dSet$ is not empty then
    for $k \in dSet$ and BA$[k]$ has not been provided input do
        $coin_k \leftarrow$ ComCoin(k)
        Provide input $coin_k$ to BA$[k]$
    $dSet \leftarrow \{\}$
if having received outputs of value 1 from at least $N-F$ BA instances then
    Provide input 0 to each of the remaining BA instances that have not been provided input
if all N BA instances have output a value then
    $CS \leftarrow \{k |$ BA$[k]$ outputs $1\}$
    $CSP \leftarrow \{\mathcal{P}_k | k \in CS\}$ (wait for RBC$[k]$ to deliver $\mathcal{P}_k$ if not received yet)
    Return CS, CSP

Following this intuition, we devise the WP-ACS protocol as in Algorithm 8. We adopt

the RBC, BA, ComCoin primitives from [40, 43, 149] respectively, for their recognized performance. In the beginning, every node computes the priority list $\mathcal{L}$ using the node weights $\{w_n\}_{n\in[N]}$ from the last epoch: $\mathcal{L}$ contains the identifiers of the top $N-F$ nodes ranked by weights, and their proposals are tagged 'correct'. Nodes outside $\mathcal{L}$ are considered potentially Byzantine and their proposals are tagged 'questionable'. When a node receives the delivery of proposal $\mathcal{P}_k$ from RBC[k] and has not provided input to BA[k] yet, it provides input 1 to BA[k] if k is in $\mathcal{L}$. If k is outside $\mathcal{L}$, it is added to the deferred action set $dSet$. Only when all BA instances specified by $\mathcal{L}$ have been provided input, will the protocol provide $coin_k$ to BA[k] for $k \in dSet$. Here $coin_k \in \{0,1\}$ is the common coin received from ComCoin(k). When at least $N-F$ BA instances deliver 1, all the remaining BA instances are provided input 0 to facilitate a timely conclusion (per the "termination" requirement). When all N BA instances complete, CSP is assigned to the proposals whose BA output 1. At the end, CSP contains all the $N-F$ proposals identified in $\mathcal{L}$ plus possibly some of the F 'questionable' proposals.

The deferred coin-toss treatment is meant to leave a chance to the 'questionable' proposals that finish RBC early. We refrain from directly rejecting 'questionable' proposals, i.e., providing input 0 to their corresponding BA instances, due to the following consideration. If $\mathcal{P}_k$ falls into the questionable proposals not because node z is Byzantine but simply due to k's recent bad source inputs, $\mathcal{P}_k$ should be still given a chance, though reduced, of being considered in the ensuing global TD so that w_k can be reevaluated (i.e., having the node weight restored). This essentially provides honest nodes a recovery path from short-term deterioration in source data quality.

4.6 Analyses

4.6.1 Byzantine Resilience

We show DecenTruth's resilience against Byzantine sources and nodes under the previously defined threat model.

Proposition 4.1 (Byzantine source resilience). The CBI-TD component minimizes the impact of Byzantine sources on the truth estimates and can swiftly recover from the degradation in accuracy caused by suddenly turned Byzantine sources.

Proof. The resilience against suddenly turned Byzantine sources is achieved by incorporating the consistency measure (Eq. (4.5)) into source reliability calculation (Eq. (4.6)). When source $s \in \mathcal{S}_n$ becomes Byzantine, its inputs arbitrarily deviates from the ground truths. The consistency measure κ_s^n is a third-order statistic that evaluates the change in the reliability of source s across different epochs. A low κ_s^n value captures the fact that source s is no longer bound to the fundamental assumption of consistent reliability. As a result, when source s starts to behave Byzantine, node n will assign a near-to-zero κ_s^n in the subsequent epochs, resulting in a near-to-zero reliability degree r_s^n which minimizes the impact of source s' subsequent inputs. This quick resilient reaction as well as the high accuracy of truth estimates in the long term are built on the honest majority premise on all sources, which safeguards that at least half of the local truth proposals to WP-ACS are also statistically reliable (the worst case is when the Byzantine sources are concentrated on a half of the nodes). □

Proposition 4.2 (Byzantine node resilience, normal case). If the adversary randomly corrupts up to F nodes but does not control the network delays, then at least $\frac{6}{7}$ of the proposals consumed by global TD will be proposals from honest nodes in the long term.

Proof. First of all, the termination and agreement properties (§4.4.3) are provided by the WP-ACS consensus as $N \geq 3F+1$. Then, in the network without the adversary controlling the delays, all RBC instances shall finish in a random order. According to Algorithm 8, if a proposal $\mathcal{P}_k$ is marked 'questionable' (i.e., $k \notin \mathcal{L}$), its chance of being accepted into CSP is: $Pr(k \in CS | k \notin \mathcal{L}) = \frac{N-F}{2N}$ which represents that RBC[k] is among the first $N-f$ RBC instances to finish and $coin_k = 1$. Then the expected number of 'questionable' proposals in CSP is $\frac{F(N-F)}{2N}$. Meanwhile, since the $N-F$ 'correct' proposals are guaranteed in CSP and $N > 3F$, then the expected size of CSP is $\frac{(N-F)(2N+f)}{2N}$ and the expected ratio of 'correct' proposals in CSP is

$$\Gamma_c = \frac{2N}{2N+F} \in (\frac{6}{7}, 1] \tag{4.11}$$

For any Byzantine node z, if $\mathcal{P}_z$ is excluded from CSP for the current epoch, the chance of its acceptance for the next epoch will stay below $\frac{1}{7}$. On the other hand, if a Byzantine proposal $\mathcal{P}_z$ happens to be included in CSP, the global TD will nonetheless assign a low value to the node weight w_z as long as $\mathcal{P}_z$ deviates significantly from those of honest nodes. And its next-epoch proposal will be marked 'questionable'. When more epochs pass, the group of Byzantine nodes will converge with the group of 'questionable' nodes, resulting in a long-term CSP with at least $\frac{6}{7}$ honest proposals. □

Proposition 4.3 (Byzantine node resilience, severe case). If the adversary can corrupt any targeted node (up to F) and controls the network delays, i.e., as the network scheduler, then at least half of the proposals consumed by global TD will be proposals from honest nodes in the long term.

Proof. We consider the worst case where at some point the adversary can adaptively corrupts F nodes that all belong to the 'correct' category (corresponding to nodes within $\mathcal{L}$), and its network scheduling capability can delay the RBC instances for honest nodes' proposals

indefinitely. This results in a CSP with only $N - F$ proposals and the highest possible Byzantine proposal ratio in CSP: $\frac{F}{N-F}$, which is still less than $\frac{1}{2}$. This means the imminent global TD can still rely on the honest-majority proposals in CSP for truth aggregation and assign lower weights to the corrupted nodes. □

4.6.2 Communication and Computation Complexity

The communication overhead solely comes from the protocol messages of the consensus-plane component, i.e., WP-ACS. The communication complexity of WP-ACS follows from its composing primitives, namely RBC [40], BA [149] and ComCoin [43], yielding complexity of $O(N|\mathcal{P}| + \lambda N^2 \log N)$ bits per node per one execution. $|\mathcal{P}|$ is the size of a proposal in bits and λ is a security parameter of ComCoin.

The computation complexity is mainly contributed by the CBI-TD component. The local incremental TD and global TD yield $\Theta(\frac{S}{N})$ and $\Theta(N)$ cross-batch truth evaluations respective, resulting in $\Theta(\frac{S}{N} + N)$ complexity at each node for each epoch. In comparison, an ideally centralized TD mechanism would incur the computation complexity of $\Theta(S)$. This reflects the parallelism of our system in the data plane.

4.7 Implementation

We implemented the DecenTruth *node* program with approximately 2300 lines of Python code. The software architecture is illustrated in Fig. 4.2. It has data-plane interfaces for external data sources and the DApp instance. For source-to-node communication, we define a message format which contains a source identifier and a 15-byte payload field for source input. Inter-node communication in the consensus step is authenticated on top of TLS

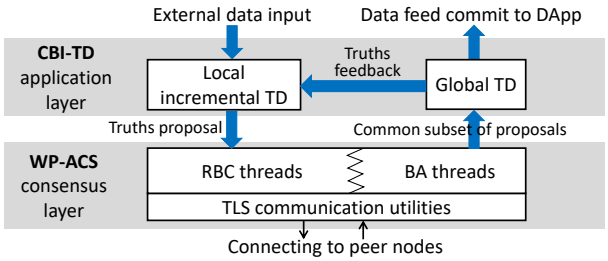


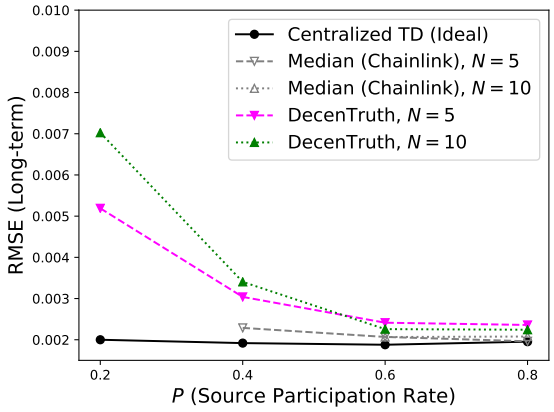
Figure 4.2: DecenTruth *node* software architecture.

communication utilities with a pre-established PKI. The message format includes a node identifier, protocol instance identifier, message type (specified by the RBC or BA procedures), and a $15B$-byte payload field for truths proposal (B is the batch size). For WP-ACS component we adopt the implementation of RBC and BA primitives from [148] while directly instantiating ComCoin with pre-distributed secret shares for reducing communication overhead. We additionally implemented an environment simulator *env* for evaluation purposes. When instantiated in an emulated network, *env* can simulate data inputs under Byzantine influence to each *node* instance and also act as the messaging intermediary for introducing packet delays.

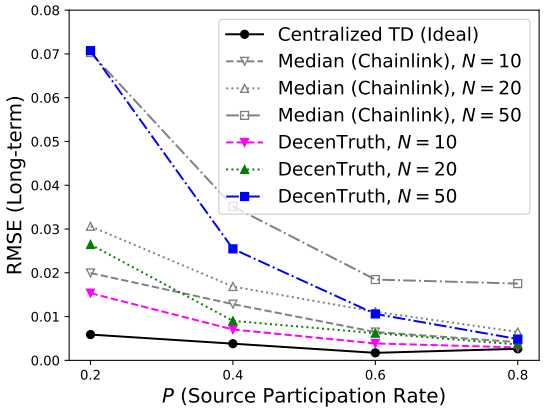
For performance comparison, we implemented a decentralized TD routine dubbed Median to emulate Chainlink's on-chain aggregation scheme that directly takes the median of all nodes' proposed local truths without having nodes go through consensus. We further implemented a centralized TD scheme following §4.3.1 as the ideal case.

4.8 Evaluation

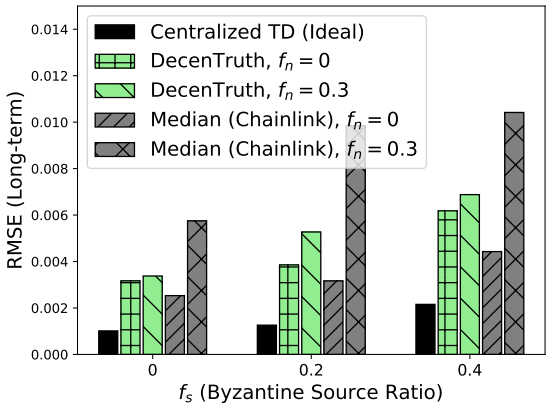
We evaluate DecenTruth's performance through network simulation experiments in a 72-core, 192GB-memory Linux machine. For each setting, $N \in \{5, 10, 20, 50\}$ *node* instances run in parallel along with one *env* instance. The maximum number of Byzantine nodes F, a



(a) With Nasdaq-100 dataset



(b) With synthetic dataset



(c) With synthetic dataset

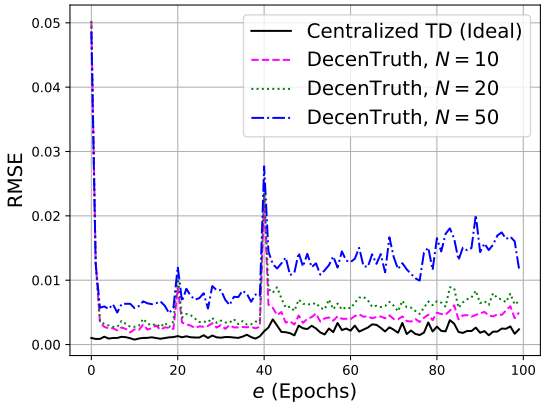
Figure 4.3: Long-term RMSE results. Other parameters: (a) $B = 100, f_s = 0, f_n = 0.3$ (b) $B = 100, f_s = 0.4, f_n = 0.3$. (c) $B = 100, N = 20, P = 0.5$.

design parameter, is fixed to $\lfloor \frac{N}{3} \rfloor$. Packet delay between any two nodes is randomly sampled from the exponential distribution of rate $\frac{1}{mdelay}$, simulating volatile communication delays, with $mdelay \in \{0.2, 0.4, 0.8, 1.6\}$ being the mean delay. *env* is pre-loaded with the datasets and randomly separates the data inputs into N source groups, with each group being local to a node. The inputs are provided to the corresponding nodes during runtime on a batch basis. To simulate a participation rate of $P \in (0, 1)$, every input is discarded with probability $1 - P$. For the weighted moving average calculation for source consistency (Eq. 4.5), we fix the decay factor $\alpha = 0.9$ and scale factor $\beta = 100$. For performance comparison, we instantiated the Median nodes and the ideally centralized TD routine in the same machine that hosted DecenTruth nodes.

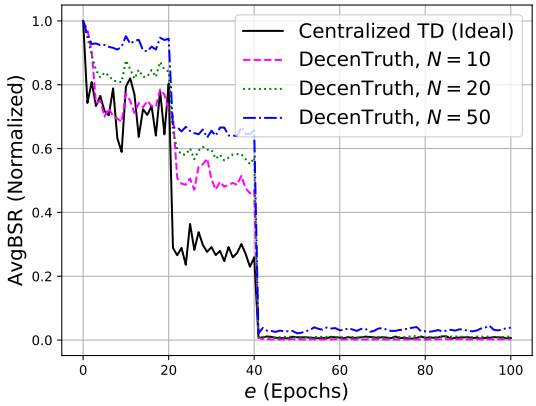
We use two datasets for evaluating data-plane performance:

- Nasdaq-100 Dataset: Stock [128] is a widely used real-world dataset in the TD line of research. It contains the trading data (last price, volume, daily change, etc.) of 1000 stock symbols in the 21 trading days of July 2011 from 55 web sources along with "goal standards" (i.e., ground truths) from the Nasdaq exchange. We extracted the daily last prices of the Nasdaq-100 symbols, resulting in a TD-compatible dataset containing inputs from 55 sources on 2,100 objects. We use it for evaluating the long-term accuracy performance of our system.

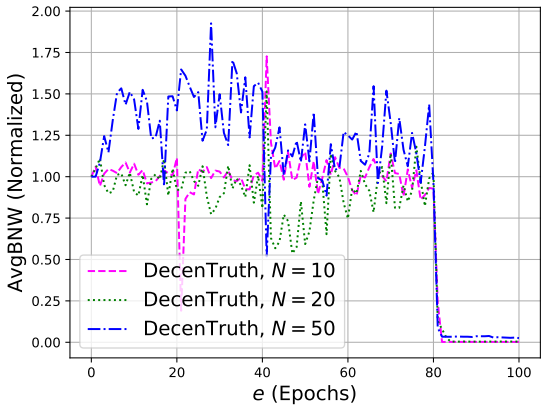
- Synthetic Dataset: We generated a synthetic dataset containing inputs from 1,000 sources on 10,000 objects, emulating stock quotes with normalized value. Each input $x_{s,i}$ from source s on object i is randomly sampled from $N(\dot{x}_i, \delta_s)$ bounded by $[0, 1]$, with $\dot{x}_i$ being the ground truth. δ_s is a predetermined value randomly sampled from $U(0, 0.5)$, representing the intrinsic unpredictability of source s. Building on top of the large number of data objects and sources in this dataset, we are able to evaluate



(a) RMSE per batch



(b) Avg. reliability of Byzantine sources



(c) Avg. weight of Byzantine nodes

Figure 4.4: Temporal data-plane performance showing DecenTruth's reaction to Byzantine influence ($B = 100, P = 0.5, f_s = 0.4, f_n = 0.3$). The synthetic dataset was used. 200 ($= 1000 f_s \cdot 0.5$) sources turned Byzantine at epoch 20 and 40; $N f_n$ nodes turned Byzantine at epoch 80.

the performance of DecenTruth upon malicious modification by Byzantine sources at different settings.

Byzantine Mutinies. To evaluate our system's Byzantine resilience, we impose Byzantine influence on the synthetic dataset. Byzantine sources or nodes may arbitrarily deviate their output data from the input values, but still within the $[0, 1]$ range. For each experiment run iterating through 10,000 objects, three "Byzantine mutinies" take place sequentially. f_s ratio of all sources eventually turn Byzantine with the first half turn at epoch $\lfloor \frac{2000}{B} \rfloor$ (1st

mutiny) and the second half turn at epoch $\lfloor \frac{4000}{B} \rfloor$ (2nd mutiny). f_n ratio of all N nodes turn Byzantine at epoch $\lfloor \frac{8000}{B} \rfloor$ (3rd mutiny). B is the batch size and $\lfloor \frac{10000}{B} \rfloor$ is the total epoch number. For fair comparison across different evaluation settings, the to-be-Byzantine nodes and sources under each (N, P) are pre-selected.

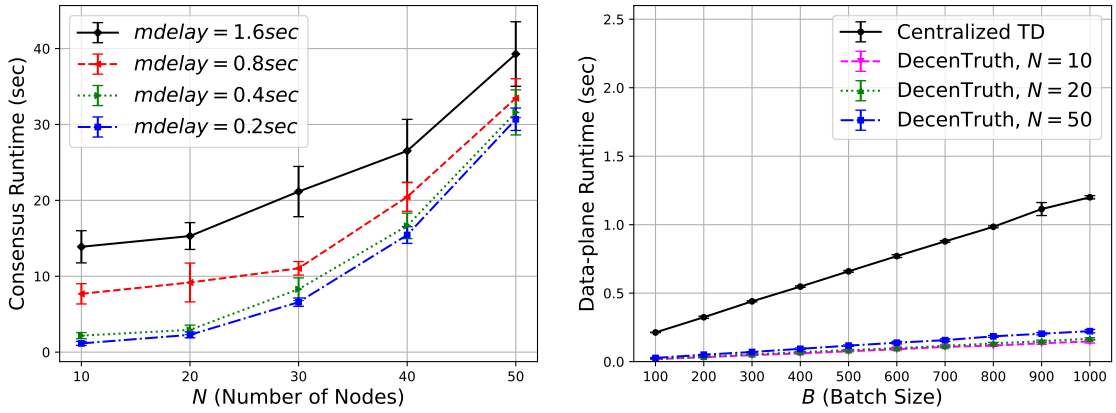
4.8.1 Data-plane Performance and Byzantine Resilience

We evaluate the TD accuracy of DecenTruth and its resilience against Byzantine mutinies. For each evaluation run, we collected the last 10% batches of final estimated truths and computed their RMSE against the ground truths. The RMSE here is a long-term measure which quantifies the system's overall TD (in)accuracy. The Nasdaq-100 result in Fig. 4.3a shows much lower RMSE than that in Fig 4.3b where the synthetic dataset was used. This is because we introduced Byzantine mutinies to sources (up to 40%) in the synthetic dataset while leaving Nasdaq-100 dataset the way it was. In Fig. 4.3a, when source participation rate is low (i.e., $P = 0.2$), the Median method fails to converge while DecenTruth still does. When Byzantine sources are present, as in Fig 4.3b, DecenTruth outperforms Median for every N. Both Fig 4.3a and 4.3b show that larger N leads to higher RMSE of DecenTruth, and lower participation rate further increases the gap from the ideally centralized TD scheme. Here we give an intuitive explanation. When N is large, each node has fewer local sources. This is fine when source participation rate is high, as local inputs of each node still cover most objects in one batch. However, if source participation rate is low, each node receives a small amount of inputs and a significant portion of objects are not covered by any local input at all. As a result, the proposals received by global TD will find less common ground in approaching the ground truths. Fig. 4.3c shows the influence of Byzantine sources and nodes at a fixed N. We observe that higher Byzantine source ratio (f_s) and actual Byzantine node ratio (f_n) both contribute to higher long-term RMSE, but limited in scale compared to

the impact of low source participation, as we observed in Fig 4.3b. Fig. 4.3c also shows that when $f_n = 0.3$, DecenTruth outperforms Median for every f_s, demonstrating its advantage in dealing with Byzantine sources or nodes.

To provide insight on DecenTruth's resilience to adaptive Byzantine influence and how the data-plane performance converges in real time, we measured the temporal RMSE value per batch, average Byzantine source reliability degree (AvgBSR), average Byzantine node weight (AvgBNW) throughout epochs for a single run. Fig. 4.4a demonstrates the quick convergence in accuracy at the start and the two Byzantine source mutinies (at epoch 20 and 40) and the weak influence of Byzantine node mutiny (at epoch 80). Fig. 4.4b and 4.4c further demonstrate DecenTruth's capability in detecting Byzantine behaviors and assigning low reliability degrees to Byzantine sources and low weights to Byzantine nodes.

Fig. 4.4a also implies that for larger N, the same ratio of Byzantine sources causes deeper accuracy degradation, as is indicated by the blue curve's much steeper increase in RMSE after each Byzantine mutiny. This is because when the reliability degrees of Byzantine sources are penalized, i.e., swiftly reduced to near-zero after each mutiny (see Fig. 4.4b), impact of Byzantine sources on the system's TD accuracy is the same as those who do not participate at all, which is verified by the decreasing trend of curves in Fig. 4.3a and 4.3b. This implies that higher Byzantine source and node ratio have the equivalent impact of reduced overall participation rate. Therefore, we argue that in practical scenarios, DecenTruth achieves Byzantine resilience as well as long-term TD accuracy comparable to those of a centralized scheme if source participation rate is maintained at a high level.



(a) Consensus runtime (b) Data-plane runtime

Figure 4.5: System runtimes for one batch. The synthetic dataset was used. (a) $B = 100, P = 0.5, f_s = 0.4, f_n = 0.3$. (b) $P = 0.5, f_s = 0.4, f_n = 0.3$.

4.8.2 System Runtime

We evaluate the system runtime of DecenTruth under different networking scenarios with the same TD setup. Batch size B is fixed to 100 so that each local truths proposal contains a 1500-byte payload. Fig. 4.5a shows the consensus runtime results under different network size N and *mdelay*. It is observed the consensus runtime grows linearly in *mdelay* and quadratically in N. And N tends to have a bigger performance impact as it increases. We speculate that the reason is two-fold: larger N leads to higher processing overhead for cryptographic routines at each node, and the total processing capacity of our simulation environment is limited. When N is large, the processing overhead can easily overshadow the communication overhead. In practical cases, the node population can be controlled to ensure bounded consensus delays. For instance, Chainlink relies on a fixed set of 21 oracle nodes for providing the data feed service [35].

We recorded the total runtime caused by TD computation (i.e., local estimate, global TD, and local update) at each node. Fig. 4.5b shows average TD runtime and sample standard deviation in one batch, under settings with different network size N and batch

size B. It is observed that TD runtime is linear to B while the centralized scheme yields significantly longer runtime. This result validates our analysis on computation complexity in §4.6 that the parallelism of our system and thus boosted efficiency in the data plane. It also observed that the runtime increases with N for a given B, which is intuitive as when N increases, so does the workload of global TD, which iterates through the data inputs multiple times in contrast to the single-pass nature of local incremental TD.

4.9 Discussion and Future Extensions

Consensus Efficiency. The WP-ACS consensus introduces significantly larger delays comparing to the data plane. This is mainly contributed by the cryptographic overhead in WP-ACS' RBC and BA instances, which are designed under the harsh asynchronous network condition. To achieve fast consensus, the population of consensus participants (i.e., DecenTruth nodes) should be limited, resulting in a scalability deficiency. A potential workaround is to instate a connectivity watcher in the data feed's overlay network for spotting the window for employing more efficient, partially synchronous consensus protocols [49, 237]. In future we will also explore lightweight cryptography for realizing the RBC and BA primitives which compose a more efficient WP-ACS protocol.

Privacy Extension. A DApp can be subject to privacy protection requirements for data sources when the data feeds contain sensitive information about the sources. The current design of DecenTruth provides partial protection on data source privacy, as the raw data provided by local sources to a node are not revealed to peer nodes. Nonetheless, it is still possible for an attacker who controls a significant portion of nodes to infer certain source inputs by exploiting the correlation between different local truths proposals on common objects.

Previous work [131, 200] has demonstrated incorporating differential privacy mechanisms into TD scenarios to enhance source privacy. In future work we will explore instantiating local differential privacy [76] in our system to enable nodes and sources to control their privacy exposure.

Compatibility with Existing Solutions. DecenTruth can be directly deployed to existing first-party data feeds (e.g., PDFS [100], API3 Airnode [20]) for enhancing data-plane performance, since the interests for high-quality data feed and correct DApp operation are naturally aligned for honest DApp participants. When extending to the third-party oracle model (e.g., Chainlink [35], BandChain [174]), Sybil attacks become a legitimate concern since the commercial oracles are generally not stakeholders of a DApp and DApp participants may not have faith in that a 2/3 of oracles would run DecenTruth correctly. To overcome such trust barrier, aside from the existing oracle reputation mechanisms (e.g., used in Chainlink), we could instantiate the data-plane component, i.e., CBI-TD, in a hardware-based trusted execution environment (TEE), so that its authenticity and integrity is attestable by DApp participants and peer oracles without assuming trust on the oracles themselves. This in part resembles Town Crier's approach [244] which uses TEE for data integrity. We will explore this approach in future extensions.

4.10 Conclusion

Eyeing on the external data truthfulness challenge on facing blockchain DApps, we identify the need for data-oriented solutions and propose a decentralized truth discovering data feed model to enable a DApp to obtain truthful data on common data objects from untrusted external sources, while preserving the decentralized property of DApp operation. As a con-

crete instantiation of this model, we introduce the DecenTruth architecture that harmonizes techniques from two domains, namely truth discovery (TD) and asynchronous BFT consensus, while addressing challenges in system security and resilience under a practical threat model. We implemented DecenTruth and evaluated its off-chain performance in an emulated DApp data feed scenario. The result demonstrates the Byzantine resilience of DecenTruth as well as its long-term TD accuracy comparable to those of an ideally centralized TD method under when a certain source participation level is met. We hope DecenTruth can provide a practical direction towards trustworthy data aggregation in the decentralized, distrustful blockchain world.

Chapter 5

Private Data Usage Control with Blockchain and Off-Chain Execution

(Copyright notice[1])

5.1 Introduction

The recent emergence of big data analytics and artificial intelligence has made life-impacting changes in many sectors of society. One of the fundamental enabling components for the recent advancements in artificial intelligence is the abundance of data. However, as more information on individuals is collected, shared, and analyzed, there is an increasing concern on the privacy implication. In the 2018 Facebook-Cambridge Analytica data scandal, an API, originally designed to allow a third party app to access the personality profile of limited participating users, was misused by Cambridge Analytica to collect information on 87 million of Facebook profiles without the consent of the users. These illicitly harvested private data were later used to create personalized psychology profiles for political purposes [144]. With

[1]This chapter previously appeared as a conference paper published in ESORICS 2020. ©2020 Springer Nature Switzerland AG. Reprinted by permission from Springer Nature Customer Service Centre GmbH: Virginia Tech Doctoral Dissertations, May 2022. Original publication: Yang Xiao, Ning Zhang, Jin Li, Wenjing Lou and Y. Thomas Hou, "PrivacyGuard: Enforcing Private Data Usage Control with Blockchain and Attested Off-Chain Contract Execution," In European Symposium on Research in Computer Security (ESORICS), pages 610-629. Springer, Cham, 2020 [231].

increasing exposure to the privacy risks of big data, many now consider the involuntary collection of personal information a step backward in the fundamental civil right of privacy [81], or even in humanity [150, 182]. Yet, driven by economic incentives, the collection and analysis of the personal data continue to grow at an amazing pace.

Individuals share personal information with people or organizations within a particular community for specific purposes; this is often referred to as the context of privacy [158]. For example, individuals may share their medical status with healthcare professionals, product preferences with retailers, and real-time whereabouts with their loved ones. When information shared within one context is exposed in another unintended one, people may feel a sense of privacy violation [152]. The purposes and values of those contexts are also undermined. The contextual nature of privacy implies that privacy protection techniques need to address at least two aspects: 1) what kind of information can be exposed to whom, under what conditions; and 2) what is the "intended purpose" or "expected use" of this information.

Much research has been done to address the first privacy aspect, focusing on data access control [25, 98, 216, 241] and data anonymization [73, 127, 138, 201]. Only recently, there have been a few works that attempted to address the second aspect of privacy from the architecture perspective [29, 64, 75, 188, 254, 255]. In fact, many believe that the prevention of this kind of "second-hand" data (mis)use can only be enforced by legal methods [62]. Under the current practice, once an authorized user gains access to the data, there is little control over how this user would use the data. Whether he would use the data for purposes not consented by the original data owner, or pass the data to another party (i.e., data monetization) is entirely up to this new "data owner", and is no longer enforceable by the original data owner.

Building upon our previous work [249], we present the design, implementation and evaluation of PrivacyGuard in this work. PrivacyGuard empowers individuals with full control

over the access and usage of their private data in a data market. The data owner is not only able to control who can have access to their private data, but also ensured that the data are used only for the intended purpose. To realize this envisioned functionalities of PrivacyGuard, three key requirements need to be met. First, users should be able to define their own data access and usage policy in terms of to whom they will share the data, at what price, and for what purpose. Second, data usage should be recorded in a platform that offers non-repudiation. Third, the actual usage of data should have a verifiable proof to show its compliance to the policy.

Blockchain, the technology behind Bitcoin [151] and Ethereum [227], has exhibited great potential in providing security and privacy services. Smart contract is a program that realizes a global state machine atop the blockchain and has its correct execution enforced by the blockchain's consensus protocol. PrivacyGuard enables individual users to control the access and usage of their data via smart contract and leverages the blockchain ledger for transparent and tamper-proof recording of data usage.

While smart contract and blockchain appear to be the perfect solution, there are fundamental limitations if applied directly. First, data used by smart contracts are uploaded in the form of blockchain transaction payload, which is not designed to hold arbitrarily large amount of data due to communication burden and scalability concerns [61, 151]. Second, smart contracts are small programs that have to be executed by all participants in the network, which raises serious computational efficiency concerns. For the same reason, existing platforms such as Ethereum are not purposed to handle complex contract programs [229]. Last but not least, data used by smart contracts are available to every participant on blockchain by design, which conflicts with the confidentiality requirement of user data. Existing secure computation techniques for preserving confidentiality and utility of data, such as functional encryption [28], can nonetheless be prohibitively expensive for the network.

To tackle data and computation scalability problems, PrivacyGuard splits the private data usage enforcement problem into two domains: the control plane and the data plane. In the control plane, individual users publish the availability as well as the usage policy of their private data as smart contracts on blockchain. Data consumers interact with the smart contract to obtain authorization to use the data. Crucially, the actual data of the users are never exposed on the blockchain. Instead, they are stored in the cloud in encrypted forms. Computation on those private user data as well as the provision of secret keys are accomplished off-chain in the data plane with a trusted execution environment (TEE) [9, 142] on the cloud.

When a data contract's execution is split into control and computation, where the computation actually takes place off-chain, several challenges occur. First, the correctness of the contract execution can no longer be guaranteed by the blockchain consensus. To this end, we propose "local consensus" for the contracting parties to establish trust on the off-chain computation via remote attestations. Second, the execution of contract is no longer atomic when the computation part is executed off-chain. We design a multi-step commitment protocol to ensure that result release and data transaction remain an atomic operation, where if the computation results were tampered with, the data transaction would abort gracefully. Lastly, private data are protected inside the TEE enclave and secrets are only provisioned when approved according to the contract binding.

We implemented a prototype of PrivacyGuard using Intel SGX as the TEE technology and Ethereum as the smart contract platform. We chose these two technologies for implementation due to their wide adoption. Our design generally applies to other types of trusted execution environments and blockchain smart contract platforms. The platform fulfills the goal of user-define data usage control at reasonable cost and we show that it is feasible to perform complex data operations with the security and privacy protection as specified by

the data contract.

To summarize, we make the following contributions in this work:

- We propose PrivacyGuard, a platform that combines blockchain smart contract and trusted execution environment to address one of the most pressing problems in big data analytics—trustworthy private data computation and usage control. PrivacyGuard essentially allows data owners to contribute their data into the data market and specify the context under which their data can be used.
- We propose a novel construction of off-chain contract execution environment to support the vision of PrivacyGuard, which is the key to improving the execution efficiency of smart contract technology and enabling trustworthy execution of complex contract program without solely relying on costly network consensus.
- We implemented a prototype of PrivacyGuard using Intel SGX and Ethereum smart contract and and deployed it in a simulated data market. Our evaluation shows that PrivacyGuard is capable of processing considerable volumes of data transactions on existing public blockchain infrastructure with reasonable cost.

5.2 Background and Related Work

5.2.1 Blockchain and Smart Contract

Blockchain is a recently emerged technology used in popular cryptocurrencies such as Bitcoin [151] and Ethereum [227]. It enables a wide range of distributed applications as a powerful primitive. With a blockchain in place, applications that could previously run only through a trusted intermediary can now operate in a fully decentralized fashion and achieve

the same functionality with comparable reliability. When the majority of the network's voting power (hashing power, stake value, etc.) are controlled by honest participants, the shared blockchain becomes a safe and timestamped record of the network's activities. The conceptual idea of programmable electronic "smart contracts" dates back nearly twenty years [203]. When implemented in the blockchain platform (eg. Ethereum), smart contracts are account-like entities that can receive transfers, make decisions, store data or even interact with other contracts. The blockchain and the smart contract platform however have several drawbacks in transaction capacity [61], computation cost [52, 109], as well as privacy of user and data [109, 114].

5.2.2 Trusted Execution Environment and Privacy-preserving Computation

Creating vulnerability-free software has long been considered a very challenging problem [196]. Researchers in the architecture community in both academia [58] and industry [9, 142] have embraced a new paradigm of limiting the trusted computing base (TCB) to only the hardware, realizing a trusted execution environment (TEE). The well-known Intel SGX [142] is an instruction set extension to provide TEE functionalities. Applications are executed in secure containers called enclaves. The hardware guarantees the integrity and confidentiality of the protected application, even if the platform software is compromised. TEE has recently been adapted as a powerful tool to support blockchain-based applications [52, 84, 106, 244].

Privacy-preserving computation has been an active area of research in the past decade [48, 106, 160, 186, 211, 249]. With the increasing reliance on rich data, there has been a significant amount of research on applying cryptographic techniques to perform privacy preserving computation and data access control [12, 28, 48, 167, 211, 212]. Recently, hardware-assisted TEE has been adapted in numerous works to achieve privacy-preserving computa-

tion [84, 105, 106, 160, 186, 253]. Specially, Ryoan [106] is closely related to PrivacyGuard in this work. It combines native client sandbox and Intel SGX to confine data processing module and provide confidentiality. However, Ryoan aims to achieve data confinement with a user-defined directed acyclic graph that specifies information flow. In comparison, PrivacyGuard allows data user and consumer to negotiate data usage using smart contract with non-repudiable usage recording.

5.2.3 Combining Blockchain and TEE

The idea of moving computation off-chain to improve the performance and security is mentioned in [29, 52, 53, 109, 191, 229]. Choudhuri et al. [53] combines blockchain with TEE to build one-time programs that resemble to smart contracts but only aim for a restricted functionality. Arbitrum [109] is a system that delegates the task of smart contract verification to a small subset of managers that are incentivized to execute the virtual machines honestly. Different from Arbitrum, PrivacyGuard focuses on moving computation off-chain using trusted execution and local consensus that involve only the contract participating parties. Ekiden [52] and the Intel Private Data Object (PDO) project [29] are two concurrently developed projects that are closely related to our work. Similar to PrivacyGuard, Ekiden harmonizes trusted computing and distributed ledger to enable confidential contract execution. Ekiden offloads computation from consensus nodes to a collection computing nodes in the aim of improving the ecosystem. In comparison, PrivacyGuard is designed to fit existing blockchain infrastructure. The Intel PDO project aims to combine Intel SGX and distributed ledger to allow distrustful parties to work on the data in a confidential manner. However, the system focuses heavily on a permissioned model with significant overhead for bootstrapping trust.

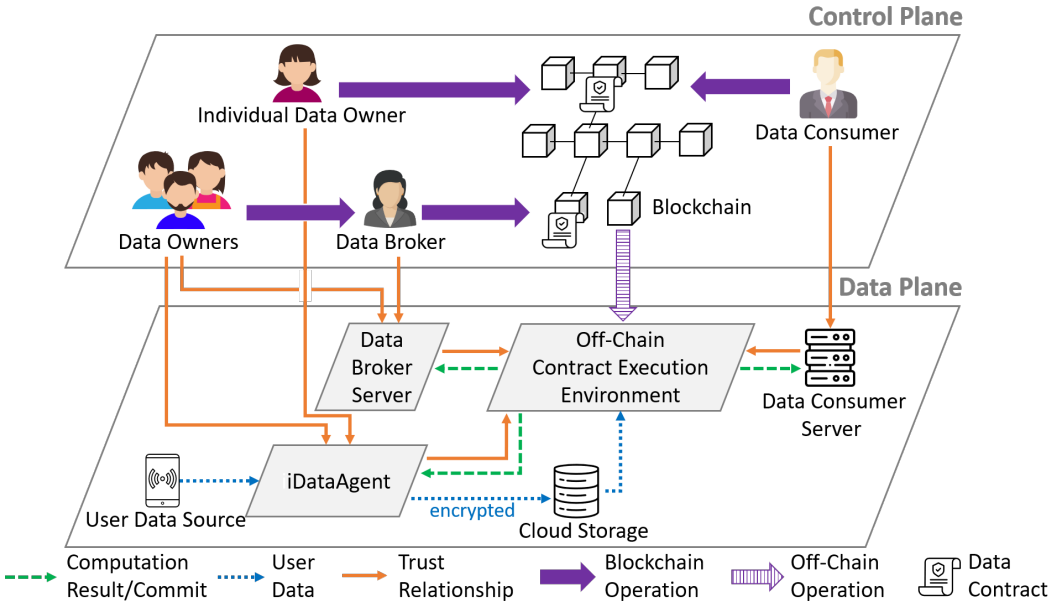


Figure 5.1: System architecture for PrivacyGuard framework.

5.3 PrivacyGuard Overview

5.3.1 System Goal and Architecture

The vision of PrivacyGuard is to not only protect data owner privacy but also promote a vibrant data sharing economy, in which data owners can confidently sell the right to use their data to data consumers for profits without worrying about data misuse. To realize this, there are three specific goals. First, data encryption/decryption are fully controlled by data owners. Untrusted parties (eg. cloud storage and data consumers) can not obtain or possess data owners' plaintext data. Second, Data owners are able to control who can access which data items under what conditions for what usage. The data usage records should be non-repudiable and auditable by data owners. Third, the security mechanism of our system should be able to capture user-defined policies and enforce the compliance of the policies during the execution of data access.

Fig. 5.1 shows the system architecture of PrivacyGuard. Although we have been using the term users to refer to both individuals and organizations, we differentiate two roles that an user in the data market can take. We refer to the individual or organization that owns the data as data owner (DO) and the entity that needs to access the data as data consumer (DC). Classified by the assigned responsibility, there are three main functional components in the PrivacyGuard framework:

- Data Market: Data market is an essential PrivacyGuard subsystem that supports the supply, demand and exchange of data on top of blockchain. For data access and usage control, DO can encode the terms and conditions pertaining to her personal data in a data contract, and publish it on a blockchain platform such as Ethereum. Data usage by DC is recorded via transactions that interact with the data contract.

- iDataAgent (iDA) and Data Broker (DB): iDA is a trusted entity representing an individual DO and responsible for key management for the DO. It also participates in contract execution by only provisioning the data key material to attested remote entities. Since it is often not realistic to expect individual DO to be connected all the time, iDA can also be instantiated as a trusted program in a TEE-enabled cloud server. To address the inherent transaction bandwidth limit of the blockchain network, DB is introduced to collectively represent a group of users.

- Off-chain Contract Execution Environment (CEE): This off-chain component executes data operations contracted between DO(s) and DC in a TEE enclave. The trusted execution guarantees correctness as if it was executed on-chain. The computation result is securely committed to DC while enforcing the contract obligation.

5.3.2 PrivacyGuard High-Level Workflow

The workflow of PrivacyGuard proceeds in three stages which can function concurrently. Stage 1 and 2 involve the supply side (DO, iDA, DB) that prepares the data items and usage contracts while stage 3 characterizes the regular operation.

Stage 1: Data Generation, Encryption, and Key Management: In this stage, a DO's data are generated by its data sources and collected by its iDA, who passes the encrypted data to the cloud storage. Keys for data en/decryption are generated by the DO via interface to iDA and managed by iDA. For a group of users with common data types, they can delegate their trust to a DB by remote-attesting the DB's enclave and provision data keys to the enclave.

Stage 2: Policy Generation with Smart Contract: In PrivacyGuard, individual DOs can define their own usage policies for their private data in DO contract (C_{DO}). The policies encoded usually includes the essential components for privacy context, such as data type, data range, operation, cost, consumer, expiration, etc. The operation, which specifies intended usage of the targeted data, can be an arbitrary attestable computer program. This paradigm grants DOs fine-grained control on the data usage policy and the opportunity to participate in the data market independently. However, it requires ample transaction processing capacity from the blockchain network that scales in the number of DOs. Alternatively, the DB-based paradigm uses DB as a trusted delegate for a large number of DOs. DB represents the DOs in the blockchain by curating a DB contract (C_{DB}) that accepts data registries from DOs and advertising their data in bundles. The encoding of C_{DO} and C_{DB} will be elaborated in §5.4.

Stage 3: Data Utilization and Contract Execution: DC invokes a C_{DO} (or C_{DB}) for permission to use certain private data of the targeted DO(s) for a specific operation, and

deposits payment onto the contract. If permission is granted on the blockchain, DC instructs CEE to load the enclave program for the contracted operation whose checksum is specified in the contract. Then both the DC and iDA (or DB) proceed to remote-attest the CEE enclave. This essentially allows the two parties to reach a "local consensus" on CEE's trustworthiness that enables the off-chain execution of the on-chain contract. When the attestations succeed, iDA (or DB) provisions data decryption keys to the CEE enclave to enable data operation within the enclave. When the operation finishes, the enclave releases the result in encrypted form and erases all the associated data and keying materials. To achieve a fair and atomic exchange that DC gets the decrypted result while DO(s) get the payment, we propose a commitment protocol for the two sides which ensures the atomic exchange only when they agree upon each other. The detailed design of the commitment protocol will be explained in § 5.5.

5.3.3 Threat Model and Assumptions

We assume all entities act based on self-interest and may not follow the protocol. However, to maintain a reasonable scope for the paper, we assume DO will not provide meaningless or falsified data intentionally. It is possible to encode rules in smart contract to penalize DOs for abusing the system with bad data. Furthermore, we assume the security systems, i.e. the blockchain and TEE, are trustworthy and are free of vulnerability. Specifically, in the control plane, we assume the blockchain infrastructure is secure that adversaries do not control enough resources to disrupt distributed consensus. We also assume smart contract implementations are free of software vulnerability. In the data plane, we assume the TEE is up to date, and particularly, Intel SGX, is secure against malicious attack from the operating system. We recognize that TEE implementations are not always perfect, and previous work has demonstrated side channel information leakage on the SGX platform alone [208, 209,

218, 235, 247, 248, 250], preventing such attacks is an important but orthogonal task. We also assume that all data operations requested by DC have been ratified by trusted sources and a cryptographic checksum of the program binary is sufficient for PrivacyGuard to check the data operation integrity.

5.4 Data Market of User-Defined Usage with Blockchain

The intuition behind the data market is to enable fair and transparent data transactions between DO and DC. In PrivacyGuard, DOs advertise private data items available for knowledge extraction on blockchain smart contracts. DC shops for a desirable data set and contract for his analytics. To start the data transaction, DC invokes the data contract and deposits a payment. The sales of knowledge extraction rights on private data are fulfilled that DO obtains the payment while the DC obtains the knowledge. The data transaction is then recorded in the blockchain with transparency. To enable user-defined access and usage control, the data contract, needs to encode DO's data usage policy including how data can be used by which DC at what cost. Next we present the our data contract design in PrivacyGuard in a constructive manner.

5.4.1 Encoding Data Usage Policy with Smart Contract

The Basic Data Usage Contract. In the conventional data sharing scenario, the data access policy often includes attributes such as type of the data, range or repository of the data, DO and DC credentials. For example, we assume patient X with public key pair (pk_X,sk_X) has three types of medical data: radiology data, blood test data and mental record data. X is only willing to share his radiology data (with descriptor $pData$) with urology specialist S

with public key pk_S. X can treat S as a DC and specify an access policy P in a data access contract: $\mathrm{C}_{\mathrm{X(DA)}} = \{P = \{pData, pk_S\}, Sig_{sk_X}(P)\}$. This encoding, however, specifies only data access but no obligation of the DC once access is granted. The DC could share the data with other parties against the original intention of the DO. To enable fine-grained control on how data is used, obligations need to be attached to the policy. For instance, if X only wants S to run a certain operation op on the data, then X can encode a new data usage contract in the following form: $\mathrm{C}_{\mathrm{X(DU)}} = \{P = \{pData, op, pk_S\}, Sig_{sk_X}(P)\}$.

Enabling Data Market Economy: A key feature of PrivacyGuard is to encourage DOs to share private data for public welfare as well as financial rewards without concerning privacy leakage or data misuse. Building on top of the success of cryptocurrency, the blockchain smart contract platform allows DO and DC to transact on the usage of data with financial value attached. DO can specify a price tag $\$pr$ (in cryptocurrency) in the policy. To further ensure a fair exchange that DC gets the knowledge and DO gets the payment, certain control logic should be instated in the form of smart contract functions. We call these functions and other contract metadata the contextual information, denoted ctx. Back to the previous example, we now have $\mathrm{C}_{\mathrm{X(DU)}} = \{P = \{pData, op, \$pr, pk_S\}, ctx, Sig_{sk_X}(P||ctx)\}$. In blockchain domain, the signature is conveniently fulfilled by X's signature in the contract creation transaction.

Transparent Tracking of Data Utilization: For the system to provide transparent data utilization tracking and policy compliance auditing, each data transaction needs to be recorded in a tamper-resistant and non-repudiable manner. In PrivacyGuard, contract functions (part of ctx, invoked via blockchain transactions) are used to facilitate the recording of data utilization. Since the blockchain ledger is publicly managed via global consensus and unforgeable, contract function invocations in blockchain transactions can provide non-repudiable records on data utilization.

Algorithm 9: Data Owner's Smart Contract C_{DO} Pseudocode

```
/* Contract creation by DO with a policy                                    */
Function Constructor()
    Parse policy as (dataset, price, operation, DCList, requestTimeout) ;
    pDS ← policy.dataset ;
    pPrice ← policy.price ;
    pOP ← policy.operation ;
    pDCL ← policy.DCList ;
    pRTO ← policy.requestTimeout ;
    R ← [ ]      // Usage records ;
    DO ← creator ;
/* Callable by DC                                                           */
Function Request(op, data, $f)
    if op = pOP and sender ∈ pDCL and data ⊂ pDS and f ≥ pPrice then
        Create a record entry R[idx] with index idx for this new data transaction ;
        R[idx].{data, DC, reqTime} ← {data, sender, sys.time} ;
        R[idx].status ← wait_computation ;
    else
        Return $f to sender and terminate ;
/* Callable by DC                                                           */
Function ComputationComplete(idx, K_resultHash)
    R[idx].krHash ← K_resultHash ;
    R[idx].status ← wait_complete ;
/* Callable by DO                                                           */
Function CompleteTransaction(idx, K_result)
    if Hash(K_result) = R[idx].krHash then
        Send $f to DO ;
        R[idx].kr ← K_result ;
        R[idx].status ← complete      // Data transaction complete ;
/* Callable by DC                                                           */
Function Cancel(idx)
    if sender = R[idx].DC and (sys.time − R[idx].reqTime) > pRTO then
        Return $f to R[idx].DC; ;
        R[idx].status = canceled ;
/* Callable by DO                                                           */
Function Revoke()
    if sender = DO then
        contract selfdestruct ;
```

Data Owner's Smart Contract C_{DO}. We design C_{DO} to capture the functionalities discussed above. The pseudo code of C_{DO} in shown in Algorithm 9. In addition to the policy variables, C_{DO} encodes functions for enforcing the control logic. Constructor initializes the policy at contract creation. Request takes a payment deposit from DC along with the requested operation *op*, the requested data descriptor D_{target}, and authorizes this data transaction. ComputationComplete is called by DC to signal the completion of the off-chain data execution. CompleteTransaction is called by DO to record the data usage and completes the transaction. The deposited payment is then redistributed to DO. We will cover more details on them along with the result commitment process in §5.5. Cancel is called by DC to abort the current transaction if the timeout passes. Lastly, Revoke invalidates the contract and can be called only by DO.

5.4.2 Using Data Broker to Address the On-Chain Scalability Challenge

While C_{DO} allows individual DOs to have fine-grained control over data usage policy and participate in data market independently, this paradigm puts heavy pressure on the blockchain transaction processing capability when the number of DOs is huge. In the meantime limited transaction throughput is a known problem for major public blockchains [38, 61, 83]. While there are many ongoing efforts to scale up transaction throughput [33, 171], we take a different but complementary approach to address this issue in PrivacyGuard's scenario. A trusted delegate, namely data broker (DB), is used to represent a group of users and curates a DB's contract (C_{DB}). C_{DB} allows individual DOs to register data entries and operations for DB to moderate. DB then participates in the data market on behalf of the registered DOs. We call this paradigm the DB-based system in our later implementation, in contrast to the iDA-based system.

Algorithm 10: Data Broker's Smart Contract C_{DB} Pseudocode

```
/* Contract creation by DB with config                                          */
Function Constructor()
    Parse config as (operationList, requestTimeout) ;
    cOPL ← config.operationList ;
    cRTO ← config.requestTimeout ;
    {DO, DS, R} ← {[[ ]], [ ], [ ]}      // DOs, data sources, data usage records ;
    DB ← creator ;
/* Callable by DO                                                               */
Function Register(op, DC, price)
    Create a DO entry DO[ido, op] with index ido for this new DO;
    DO[ido, op].{DO, DC, price} ← {sender, DC, price} ;
/* Callable by DB                                                               */
Function Confirm(cfDOs)
    for all {ido, op} that ido ∈ cfDOs and op ∈ cOPL and DO[ido, op] ≠ null do
        Append ido to DS[op].DOList ;
        Append DO[ido, op].DC to DS[op].DCList ;
        DS[op].price ← DS[op].price + DO[ido, op].price ;
/* Callable by DC                                                               */
Function Request(op, targetDOs, $f)
    if op ∈ cOPL and sender ∈ DS[op].DCList and targetDOs ⊂ DS[op].DOList and
      f ≥ DS[op].price then
        Create a record entry R[idx] with index idx for this new data transaction ;
        R[idx].{targetDOs, DC, reqTime} ← {targetDOs, sender, sys.time} ;
        R[idx].status ← wait_computation ;
    else
        Return $f to sender and terminate ;
/* Callable by DC                                                               */
Function ComputationComplete(idx, K_resultHash)
    (same as in C_DO, see Algorithm 9)
/* Callable by DB                                                               */
Function CompleteTransaction(idx, K_result)
    if Hash(K_result) = R[idx].krHash then
        for all ido ∈ DS[R[idx].op].DOList do
            Send $DO[ido, R[idx].op].price to DO[ido].DO;
        R[idx].kr ← K_result ;
        R[idx].status ← complete      // Data transaction complete ;
Function Cancel(idx)
    (same as in C_DO)
Function Revoke()
    (same as in C_DO, except callable by DB)
```

The pseudo code of C_{DB} is provided in Algorithm 10. C_{DB} emulates C_{DO} for most parts but with extra global variables for data source management and two more functions: Register and Confirm. When a DO wants to make use of the DB, she first invokes Register function to register her data with the C_{DB}. In the data plane, the DO needs to remotely attest the DB to establish trust, then provisions the data keys to the DB enclave. This, however, is not the end of data registration, because the data source and quality still need to be verified by the DB. Once verified, DB invokes Confirm function to complete the data registration. Furthermore, result commitment is also slightly different for C_{DB}. The CompleteTransaction function is now callable by DB and needs to distribute payments to all involved DOs.

5.5 Off-Chain Contract Execution

PrivacyGuard leverages blockchain smart contract to provide the control mechanisms for valued data exchanges. While the technology offers a distributed time-stamped ledger which is ideal in providing a transparent recording of data usage, smart contract suffers from several prohibiting drawbacks when it comes to confidential data computation purely on-chain. First, the smart contract invocation and the ensuing computation is executed and repeated by all nodes in the blockchain network. The cost to run complex algorithms on-chain can be prohibitive even assuming data storage is not an issue. Second, data has to be decrypted and stored on the chain, causing confidentiality problems.

To tackle this problem, we introduce the concept of off-chain contract execution in PrivacyGuard and introduce an entity called off-chain contract execution environment (CEE) to bring both the computation and data provisioning off-chain. Particularly, we decompose a data usage contract into two portions, the control part and the computation part. The control flow starts with invoking the contract and stops at the contracted computation task

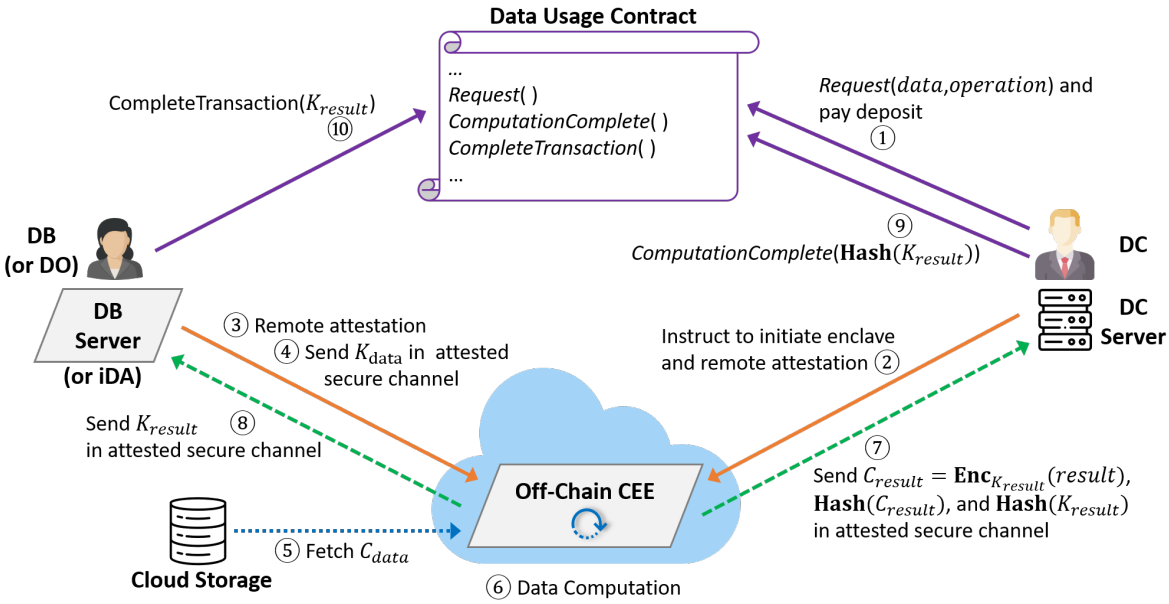


Figure 5.2: Off-chain Contract Execution and Result Commitment

which switched to off-chain. The control flow is resumed with another contract invocation when the off-chain computation task is finished. Accordingly, we propose a novel off-chain contract execution and result commitment protocol, as is shown in Fig. 5.2. Note that both DB and iDA can represent a DO. Here we resort to the DB-based paradigm for convenience of presentation. We defer the discussion on DB's role in the data plane to the end of this section. Next we elaborate on the important features of off-chain contract execution in a constructive manner.

5.5.1 Establishing Trust on the Execution of Contracted Operation through "Local Consensus"

The first challenge is the correct execution of the contracted task. As we have mentioned, when smart contracts are executed on blockchain platform, the correctness of the execution is guaranteed by the entire network through global consensus, which suffers from high on-

chain cost. Our observation is that the correctness of one particular computation instance only matters to the stakeholders of the data transaction, i.e. the DOs, DB, and DC. And we do not need the entire network to verify the correctness.

In the conventional setting of distributed computing, both the DC and DO would perform the data computation task and expect the same result from each other. However, it contradicts DO's goal of fine-grained control on data usage if the data are directly provided to DC. Instead, we rely on software remote attestation, which is a widely available primitive with TEEs [9, 142], for securely delegating the computation task to CEE. In this work we opt for Intel SGX [142]. First of all, the designated computation program should pre-ratified with its program (binary) hash published in the data contract along side "authorized operations". When instructed by DC for a specific computation task, CEE loads the corresponding enclave program for that task. Then the two transacting sides in the data plane, DB and DC, remotely attest the enclave program to verify its authenticity and integrity with the program hash in the contract. As a result, as shown in Fig. 5.2, the immediate steps after data transaction request is to have CEE load the enclave program and DC and DB remotely attest the CEE enclave. Once correctly CEE enclave is verified with attestation reports, both sides of the contract can then extend their trust to CEE, knowing the attested program will execute securely in the enclave till termination, and the computation result will be genuine even if an adversary compromises CEE's untrusted platform (i.e., "normal world" in TEE terminology, which includes the operating system and non-enclave programs). And finally the result produced by CEE will be the "local consensus" between the two sides.

5.5.2 Enforcing Data Obligation and Confidentiality

The local consensus mechanism guarantees the data intensive computation task can be offloaded to the off-chain entity CEE for execution while maintaining the correctness of computation. However, in order to achieve the privacy goals of PrivacyGuard, computation itself has to fulfill the data obligation, which we refer to as the obligations of DC for utilizing DO's data. More specifically, it follows the general requirement of secure computation, wherein only the computation result is accessible by the DC, not the plaintext source data. First, the computation process should not output any plaintext source data or any intermediate results that are derived from the source data. Second, at the end of the computation, all decrypted data and intermediate results should be sanitized. Despite recent breakthrough in fully homomorphic encryption, performing arbitrary computation over encrypted data remains impractical for generic computation. In PrivacyGuard, we make use of TEE enclaves to create the environment for confidential computing. As is illustrated by step 3 and 4 in Fig. 5.2, DO's data en/decryption key K_{data} can be provisioned to CEE's enclave only if the latter can be cryptographically verified via remote attestation and a secure channel is established. This comes as an integral part of the local consensus. The hardware of CEE, the processor specifically, enforces the isolation between the untrusted platform and the enclave. We require the enclave program to include steps to sanitize intermediate results and keying materials. Since memory contents are encrypted in Intel SGX, once the keying material is removed, the data can be considered effectively sanitized. This also ensures that the program inside the enclave will terminate once the contracted task is completed.

5.5.3 Ensuring Atomicity in Contract Execution and Result Commitment

The last challenge is ensuring the atomicity of the contract, which arises from the split of control between on-chain off-chain. Contract functions that were previously executed in a single block are now completed via multiple function invocations that are executed in multiple blocks. Furthermore, there is no guarantee on the execution time of the off-chain computation, because an adversary controlling the platform can interrupt the computation and cause delays. Specifically, under the threat model in §5.3.3, two issues need to be addressed.

The first issue is the contract function runtime. When the adversary has control of the off-chain computation platform of CEE, he can pause or delay the computation. For many data computations, the result can be time-sensitive. To tackle this problem, we add a timeout mechanism in the data contract to allow DC to cancel the request after timeout and have the deposit refunded (see Algorithm 9).

The second issue is the atomic completion of the contract. We want both the DOs to get the payment in the control plane while allowing DC to get the computation results in the data plane. This is particularly challenging due to the lack of availability guarantee on the CEE platform. When the platform is compromised, the adversary can intercept and modify any external I/O from the enclave, including both the network and storage. Our design for the atomic completion and result commitment can be observed from step 7 to 10 in Fig. 5.2. The key idea is that result release and contract completion should be done as a single message in the control plane. To prevent DC from getting the result without completing the payment to DOs, the result are encrypted into C_{result} with a random result key K_{result}, before being sent to DC in the attested secure channel. Since the platform can corrupt any output from CEE, the CEE enclave also sends DC the hash of the encrypted

result and key, i.e., Hash(C_{result}) and Hash(K_{result}), which will be later used by DC for integrity check on the result and the key. K_{result} is passed to DB in the attested secure channel.

To prevent DB from completing the transaction without releasing the correct result key, DC needs to initiate the commitment procedure in the control plane by invoking the contract function ComputationComplete with Hash(K_{result}), indicating it has the encrypted result and is ready to finish the data transaction if and only if the correct result key K_{result} is released. Upon observing the message from DC, DB then invokes the smart contract function CompleteTransaction with the result key K_{result}. Only when the hash of K_{result} matches the previously received Hash(K_{result}), will the contract write the data usage into records, release the payment to DOs, and finally conclude the data transaction. Note that our commitment protocol design does not need to protect the confidentiality of K_{result} (thus enabling the on-chain hash check). This is because the encrypted result C_{result} is passed directly from CEE enclave to DC via the attested secure channel. Finally, DC has the full discretion in deciding whether to publish the computation result afterwards.

5.5.4 Data Broker for Scalability in the Data Plane

In the iDA-based paradigm, when DC needs to use the data from a large number of DOs, the naive use of remote attestation on the CEE would require each iDA to individually attest and verify the CEE enclave, resulting in linearly growing computation overhead and network traffic. To address this challenge, in the DB-based paradigm, DB can be re-purposed as a trusted intermediary between the CEE and all relevant DOs in the data plane during the preparation stage, similar to its control plane role. Essentially, DB is also deployed on a TEE-enabled machine and instantiates an enclave for secure handling of DOs' data. The

enclave is attested to every new DO only once after the DO registers with DB. During the normal operation, DB attests CEE on behalf of all relevant DOs for each DC request, saving the need for individual DOs to attest CEE. To accommodate the extreme case when a large number of DOs registers with DB simultaneously, we will explore parallel remote attestation solutions in §5.6.1.

5.6 Implementation and Evaluation

We implemented a prototype of PrivacyGuard using Intel SGX as the TEE technology and Ethereum as the smart contract platform. Source code with documentation is available at https://github.com/yang-sec/PrivacyGuard. The on-chain components, namely the DO contract and the DB contract, were implemented in Solidity with 144 and 162 software lines of code (SLOC) respectively. The data usage price was set at 0.01 ethers per user data. The off-chain components include five PrivacyGuard applications, namely iDataAgent (iDA), Data Broker (DB), Data Owner (DO), Data Consumer (DC), Contract Execution Environment (CEE). They were implemented in C++ with Intel SGX SDK v2.3.1 on top of Ubuntu 16.04 LTS. The total SLOC for off-chain components is about 37,000.

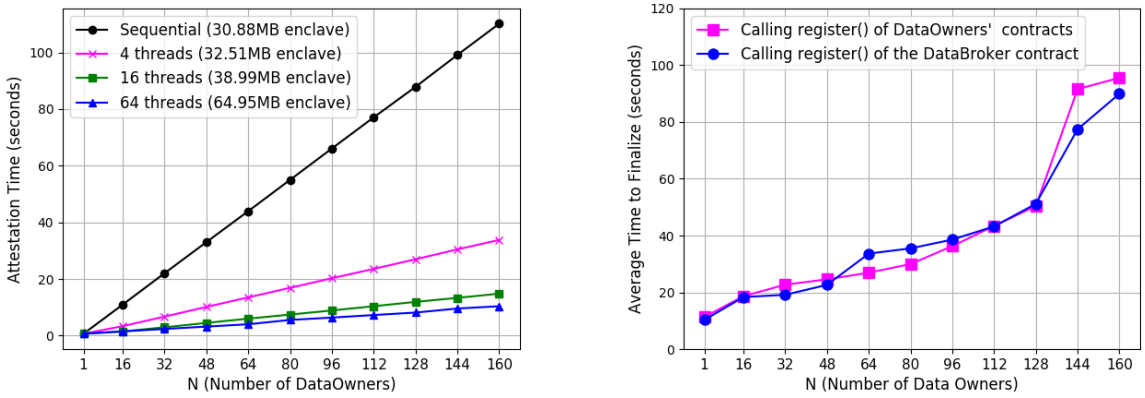
We deployed the contracts onto Ethereum Rinkeby testnet for evaluation, though our system is fully compatible with Ethereum mainnet. We used a fixed gas price of 10^{-9} ethers. PrivacyGuard applications were deployed in a LAN scenario. 1 DB, 1 iDataAgent, and 1 CEE ran on a SGX-enabled Linux machine with Intel Core i5-7260U CPU (2 cores 4 threads, 3.5 GHz). Up to 160 DOs and 1 DC ran on a Linux machine with AMD FX-8320 CPU (4 cores 8 threads, 3.5 GHz). We note that this setup aims for feasibility demonstration; in real-world deployment each application will most likely reside in a different machine. We used the adult dataset from UCI Machine Learning Repository [70] to simulate the data source. Each

DO randomly drew 500 data points from the dataset as its private data. We have tested the entire PrivacyGuard workflow in multiple runs and the data usage history has been recorded in the deployed contracts. Our evaluation focuses on the system's scalability and consists of three parts: control plane runtimes, control plane costs, and data plane runtimes.

5.6.1 Control Plane Runtimes

To accommodate the scenario where N DOs simultaneously attest the DB enclave in the DB-based system, we experimented with a parallel attestation scheme in DB that each of the N attestation instances is handled by one of the T software threads, which invokes a new attestation context of the enclave and a dedicated enclave thread control structure (TCS) (thus TCSNUM = T). The experiment was repeated under different T. To avoid congesting the Intel Attestation Service (IAS) which may violate the terms of service, we instead used a simulated IAS that responds to EPID signature revocation list request and attestation report request with 0.1s and 0.5s delays respectively. The result is shown in Fig. 5.3a. We observe that the parallel scheme is indeed a promising solution for scaling up attestation capacity, at the cost of enlarged enclave memory footprint. When $N = 160$, it takes the 64-thread DB about a tenth the attestation time of its sequential counterpart. We remark that efficient and scalable remote attestation is an interesting standalone topic to explore in future work.

To further evaluate the performance constraints imposed by the blockchain network, we measured the average transaction finalization delay in a congested environment. We set up 160 DOs to simultaneously send out a transaction calling the Register() function in the DB contract and their own DO contracts. we use receipt as the finalization response of the Ethereum transaction that makes the function call. The result is shown in Fig. 5.3b. As



(a) Attestation times of DB when N DOs simultaneously initiate attestation.

(b) Avg. transaction finalization delay when N DOs simultaneously call a contract function

Figure 5.3: Control plane runtimes

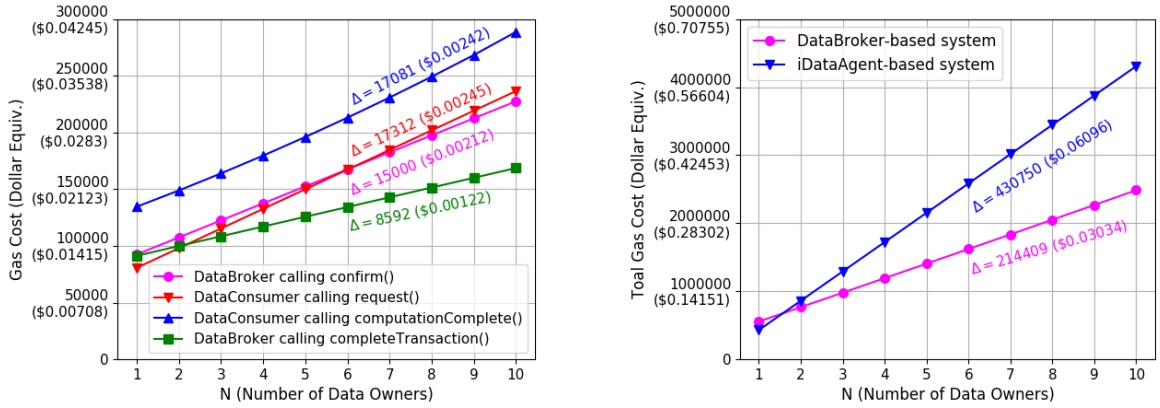
more DOs send transactions at the same time, the average time to finalize a transaction increases dramatically. A straightforward workaround is to require DOs to call Register() according to a time schedule that minimizes congestion.

5.6.2 Control Plane Cost

The monetary cost of the control plane mainly comes from the gas cost of operating smart contracts in Ethereum. At the beginning, every DO registers its data items on its own DO contract and the DB contract. DB fetches data from whoever registered with its contract and routinely confirms new registries. DC then requests for the data items from N DOs by sending a request transaction to the DB contract (or separate requests to all related DO contracts) with a sufficient deposit to cover the price before proceeding to attesting CEE. We repeated the experiment for $N = 1 \rightarrow 10$ and obtained the gas costs and dollar equivalents for each contract function call, based on the ether price on 03/31/2019, which was $141.51 (source: https://coinmarketcap.com/).

Table 5.1: Cost of the data contract's scale-independent functions

	DO Contract		DB Contract	
Function	Gas Cost	USD Equiv.	Gas Cost	USD Equiv.
constructor()	951747	0.13468	846794	0.11983
Register() (new)	156414	0.02213	125392	0.01774
Register() (update)	30121	0.00426	45177	0.00639
Cancel()	81998	0.01160	66954	0.00947



(a) Gas costs of the DB contract's scale-dependent function calls

(b) Total gas costs of a completion data transaction

Figure 5.4: Control plane cost.

We find that in both DB and DO contracts the costs of calling constructor() (contract creation), Register() and Cancel() do not depend on the number of registered DOs. We call these type of function calls scale-independent; otherwise scale-dependent. The costs of calling scale-independent functions and scale-dependent functions are shown in Table 5.1 and Fig. 5.4a. Notably, the costs of calling Request() and ComputationComplete() grow faster than the costs of calling Confirm() and CompleteTransaction(). This implies the total cost will increasingly shift to the DC side, which is a scalable trend for the system as the DC has incentives to pay for more data usage.

To evaluate the scalability gain brought by DB, we compare the case wherein individual DOs share data via their own DO contracts versus via the DB contract. In both cases,

the total amount of data requested by the DC and subsequently operated with by the CEE are the same. We summed the costs of all function calls except for the contract creation (calling constructor()) and extrapolated over different N. Fig. 5.4b shows that it costs the DB-based system much less to accommodate one extra DO ($0.0304) compared to the iDataAgent-based system ($0.06096). This result together with control plane runtimes (Fig. 5.3a) demonstrate DB's ability to provide PrivacyGuard with financial and performance scalability when facing a growing number of DOs.

5.6.3 Data Plane Runtimes

To evaluate CEE's performance in off-chain contract execution, we experimented with a demonstrative, reasonably complex computation task: training four parallel instances of a neural network classifier. Detailed hyperparameters can be found in our source code. The training functions were ported to the SGX enclave from the Fast Artificial Neural Network (FANN) Library (https://github.com/libfann/fann). To evaluate enclave overhead, we also implemented an untrusted version (executed outside enclave) of the computation task that ran on the same machine. We noticed that recent work showed Intel's Hyperthreading Technology (HTT) has flaws that may impair the security of SGX enclaves [209]. Therefore, we tested the computation task under different hardware options with respect to the usage of SGX enclave and HTT. The Intel CPU's TurboBoost feature was turned off to avoid unexpected performance gain.

The experiment results is shown in Fig. 5.5. We find that the overhead caused by disabling HTT is 48.84% for inside enclave and 17.99% for outside-enclave. This indicates disabling HTT will drag down in-enclave performance more significantly. The overheads caused by enclave are 196.55% and 274.13% for HTT-enabled and HTT-disabled respectively.

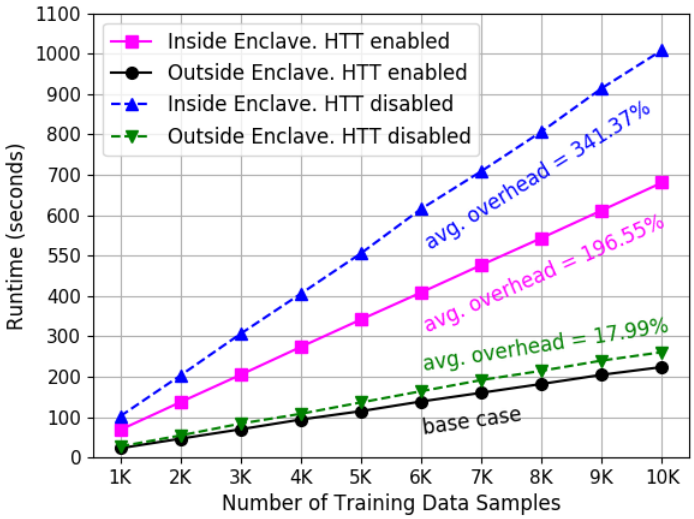


Figure 5.5: Runtimes of training an example neural network classifier under four hardware options.

We speculate that the big enclave overhead is related to the enclave's secure function calls and our imperfect porting of the training program. We leave the performance caveats of Intel SGX and possible solutions to future work.

5.7 Conclusion

In this work, we proposed PrivacyGuard, a platform that combines blockchain smart contract and TEE to enable transparent enforcement of private data computation and fine-grained usage control. Blockchain can not only be used as a tamper-proof distributed ledger that records data usage, but also facilitate financial transactions to incentivize data sharing. To enable complex and confidential operations on private data, PrivacyGuard splits smart contract functionalities into control operations and data operations. Remote attestation and TEE are used to achieve local consensus of the contract participants on the trustworthiness of the off-chain contract execution environment. Atomicity of the contract completion and

result release is facilitated by a commitment protocol. We implemented a prototype of PrivacyGuard platform and evaluated it in a simulated data market. The results show the reasonable control plane costs and feasibility of executing complex data operations in a confidential manner using the platform.

Chapter 6

Enabling Dynamic Spectrum Sharing in Low-trust Environment

(Copyright notice[1])

6.1 Introduction

Spectrum is the most important yet scarce resource for wireless communication and sensing. Spectrum regulators, such as the Federal Communications Commission (FCC) in the US, have opened up exclusively licensed or unlicensed bands for civilian use on a sharing basis. To protect the access rights of incumbent users and propose a dynamic spectrum sharing (DSS) process, the FCC has visioned for a spectrum access system (SAS) in its rulings on the 3.5GHz Citizens Broadband Radio Service band (CBRS) [206]. The current standardization effort on SAS for CBRS, led by the Wireless Innovation Forum (WInnForum), delegates the tasks of incumbents protection, user management, and coordination of spectrum sharing to individual SAS service providers. A SAS service provider generates spectrum access assignment in response to requests from its registered users [226] and different SAS service

[1]This chapter was in submission to a journal for possible publication as of May 9, 2022. Reprinted/adapted, with permission, from Yang Xiao, Shanghao Shi, Wenjing Lou, Chonggang Wang, Xu Li, Y. Thomas Hou, Jeffrey H. Reed, "BD-SAS: Enabling Dynamic Spectrum Sharing in Low-trust Environment," May 2022.

providers also need to periodically communicate with each other to synchronize service state in common service regions [223, 225].

6.1.1 Motivation for Decentralized SAS

As the NTIA and FCC plan to open up or re-purpose more wireless spectrum for shared use, such as 3.4-3.55 GHz, 3.7-4.2 GHz (C Band), 5.9 GHz (DSRC), and lower 37 GHz bands, the regulators are soliciting comments and potential DSS solutions for these bands [116]. The SAS model, which has seen an initial success in the CBRS band, poses a strong candidate for that purpose, as the existing SAS service providers can conveniently extend their spectrum management service to the other bands. However, the existing SAS model faces major challenges in terms of the centralization trust and fault tolerance.

First, it follows the centralized server-client paradigm and assumes that spectrum users can place absolute trust in the SAS service provider they subscribed to. This assumption, however, is questionable for future DSS scenes there can be an increasing number of SAS service providers servicing heterogeneous bands and users. There is no efficient mechanism to compel an individual SAS service provider to operate faithfully or follow the nuanced spectrum policies, except by the regulator's ex-parte enforcement. A malfunctioning or malicious SAS service provider would bring devastating results to its subscribed spectrum users. Second, according to the WInnForum specifications [225], SAS service providers periodically communicate with each other to synchronize service states on a request-response basis. The current inter-SAS synchronization occurs on a nightly basis [223] and it is defenseless against a malicious SAS service provider who disseminates false or tampered records, which could bring chaos to other SAS service providers. Third, as we anticipate more players to join the DSS ecosystem to use or provide SAS service with increasingly diverse value-added services

(e.g., license leasing and trading), the server-client paradigm would bring excessive (and undesirable) enforcement overhead to the regulators, who have long voiced support for a market-driven management approach [206, 240]. A SAS paradigm that is secure by design, resilient, and has a high level of self-governance is highly sought-after.

Prospect of Decentralization: From the security perspective, spectrum users must be guaranteed for reliable spectrum management service even if individual SAS servers are not trustworthy. From the performance perspective, there needs an efficient inter-SAS coordination mechanism that allow different SAS service providers to quickly synchronize service state without involving a central mediator. To this regard, we consider a decentralized, collectively managed SAS paradigm desirable in that spectrum access assignments are finalized via consensus by a group of SAS servers who can also synchronize their service states in a timely automated fashion. Users and SAS servers alike are encouraged to participate in the process honestly to realize high autonomy of spectrum management. The decentralized, self-governed paradigm provides a natural ground for innovative value-added services while minimizing the regulatory overhead. For practical purposes, this decentralized SAS model should be backward compatible with the existing SAS in terms of participants and task model, and also does not degrade SAS service quality, such as generating spectrum assignment in a responsive manner.

Blockchain as a Potential Solution: From its cryptocurrency original, blockchain emerged as novel technology for enabling fully decentralized payment networks. With a cryptography-hardened transactional model and consensus-based consistency mechanisms, blockchain enables trustworthy transaction processing and ledger keeping among mutually distrustful participants, even if a certain portion of them may behave maliciously. The decentralized and transparent nature, consensus-based security, immutable ledger keeping, and the support for self-executing smart contract have made blockchain an ideal technology to enable de-

centralized, automated spectrum management [222]. The FCC has indicated its interest in employing blockchain and distributed ledger technology for future spectrum sharing systems beyond CBRS [143]. Since then multiple blockchain-based SAS solutions have been proposed [8, 99, 246]. Still, there lacks a framework in the literature on how to decentralize the SAS amid malicious SAS servers while being backward compatible with the existing SAS.

6.1.2 Our Contribution

Built on our previous vision for decentralized SAS [189, 234], we formulate a decentralized SAS model encompassing a participation model and a task model. The participants, namely regulators, SAS servers, local witnesses, and spectrum users, get involved in four tasks—user registration, spectrum access assignment, record keeping, and regulation enforcement. We identify three key requirements that arise with decentralization: fault tolerance of all tasks when malicious SAS servers exist, responsiveness of spectrum access assignment, and backward compatibility with the existing SAS infrastructure which has been implemented for CBRS.

BD-SAS. We formally introduce the Blockchain-based Decentralized SAS architecture dubbed BD-SAS which extends from our proof-of-concept design in [234]. BD-SAS consists of two layers of smart contract-enabled blockchains: the G-Chain at the global scale for regulatory purposes and inter-SAS information exchange, and L-Chains at local regions for the actual spectrum access assignment. The G-Chain is a public venue for spectrum regulation publication as well as SAS server management and tracking of local SAS service. G-Chain participants include SAS servers, regulators, and local witnesses, who interact with a carefully designed G-Chain regulatory contract $\mathcal{C}_R$. An L-Chain is dedicated to spectrum access management for a specific geographical region and participants include SAS servers

who serve that region and witnesses who represent stable local spectrum users in that region. To enable automated spectrum assignment, a carefully designed spectrum access contract denoted $\mathcal{C}_{SA}$ is instantiated on the L-Chain and encodes the assignment function. The function is invoked at a spectrum user's request and outputs an assignment result indicating the allocated channel(s), location, and time to use. When finalized by the L-Chain consensus, the spectrum assignment records are open for auditing.

G-Chain is secure under the assumption that the majority of SAS servers are honest globally. To defend against powerful adversaries who can adaptively corrupt SAS servers to compromise a target L-Chain, we design a security mechanism called SAS server reshuffle to ensure that the SAS servers serving any L-Chain at any time are also majority-honest (i.e., THE same threat threshold as with G-Chain) with overwhelming probability. To minimize spectrum access assignment latency, i.e., for the requirement on responsiveness, we design the L-Chain workflow where the execution of spectrum access requests is separated from the total ordering of them, which are undertaken respectively by the SAS servers and witnesses. This separation scheme is essential to the responsiveness of the assignment process, as a user can start using the spectrum right after the SAS servers execute its request before the execution is finalized by local witnesses. And such response time is at most linear in the number of those SAS servers.

In summary, we make the following contributions in this paper:

- Visioning on the future spectrum sharing landscape, we propose a decentralized SAS model that allows spectrum users to access reliable SAS service without having to trust an individual SAS server. This model contains backward-compatible abstractions for system participants and tasks, and key security and performance requirements.

- We introduce the BD-SAS architecture to realize the above decentralized SAS model.

BD-SAS is the first blockchain-based SAS solution that achieves decentralization of SAS service, while being compatible with the existing SAS infrastructure and supporting automatic execution of key SAS functions.

- Within BD-SAS, we design a cross-chain security mechanism, called the SAS server reshuffle procedure, to defend BD-SAS against an adaptive adversary who can exert Byzantine influence on individual SAS servers. Moreover, we leverage the separation of consensus and execution to achieve higher efficiency in spectrum assignment.

- We implemented a BD-SAS prototype using Ethereum Rinkeby testnet (for emulating the G-Chain) and Hyperledger Fabric [7] (for realizing an L-Chain). Evaluation results demonstrate BD-SAS's capability of providing spectrum access assignment to users within five seconds under our most constraining network delay setting, showing BD-SAS's feasibility amid the tight timing regime of existing SAS operation.

6.2 Background and Related Work

6.2.1 The Existing Spectrum Access System

The SAS concept was coined by the FCC as a new DSS paradigm to open up previously under-utilized spectrum while protecting the rights of incumbent users. It was legalized in the FCC's 2015 ruling on the 3.5GHz CBRS band [206]. Based on a three-tiered access model, the SAS is designated by the FCC for managing the shared access to the CBRS band while protecting the preemptive right of Incumbent Access (IA) users and the licensed privilege of Priority Access (PA) users. The General Authorized Access (GAA) request spectrum access from the SAS but do not receive presumed interference protection.

Since 2016, the WInnForum has been leading the CBRS standardization effort, including specifications on SAS-CBSD interface [226] (CBSD stands for citizens broadband radio service devices), inter-SAS communication [225], security and PKI requirements [224]. Each SAS server, which is proprietary to a commercial administrator, maintains a database on the spectrum availability and receives CBSD registrations. The spectrum assignment process follows a server-client model and consists of three main request-response procedures—"Inquiry", "Access", and "Heartbeat". When a SAS server receives a access request from a registered CBSD, it responds with a "Grant" specifying the access assignment details, including allocated channel range, expiration time, and Heartbeat interval. Each CBSD needs to send Heartbeat requests to the SAS server periodically and receives Heartbeat responses. Each of Heartbeat response authorizes the CBSD to continue using the granted channels. To maximize spectrum utilization, the Grant expiration time is set to one minute by default [206] and the Heartbeat interval can be as tight as 30 seconds [97]. To facilitate coordination across different SAS servers, SAS servers periodically communicate with each other and synchronize service states and CBSD information [223, 225].

6.2.2 Blockchain for Spectrum Management

Prior wisdom has explored the prospective use of blockchain technology for spectrum management. It is identified in [189, 222, 251] that the key features of blockchain technology, namely decentralization, transparency, ledger immutability, and consensus-based security, are appealing to both spectrum users and regulators in the low-trust DSS scenarios. In particular, the FCC has been eyeing on blockchain's potential in enabling novel spectrum sharing and trading applications with minimal regulatory touch [143, 240]. Weiss et al. [222] provide qualitative analyses on the pros and cons of different blockchain technologies when applied to different spectrum sharing modes according to the authors' previous typol-

ogy [221]. Ariyarathna et al. [8] propose a digital-token-based spectrum access platform wherein a smart contract system is used by primary users as a trusted third-party service for advertising and leasing spectral tokens to secondary users. Grissa et al. [99] formulate a hierarchical blockchain framework called TrustSAS to enable privacy-preserving spectrum sharing among secondary users. Local blockchain networks are established among secondary users for spectrum query aggregation and response distribution while a global blockchain is used for general records keeping. Zhang et. al [246] propose a blockchain-enhanced spectrum sharing system the CBRS band where he PA users are responsible for establishing local blockchain networks to help a central regulator reduce its workload in spectrum sharing coordination.

The proposals mentioned above generally assume trust on either a third-party smart contract platform for spectrum allocation [8] or individual SAS servers for processing user inquiries [99, 246]. They are still susceptible to the security impact of malfunctioning or malicious SAS servers. In contrast, we consider a decentralized SAS model where users do not assume trust on individual SAS servers nor should SAS servers assume individual trust on each other.

6.3 System Model

In this section, we describe the participants, main tasks, and key requirements of a decentralized SAS model. Crucially, the participants and main tasks are designed to be compatible with the existing CBRS-based SAS implementation while general enough to accommodate for other spectrum bands that could potentially use a decentralized spectrum access solution.

6.3.1 Participants

There are four types of participants:

- Regulator is a government entity who publishes regulations on spectrum usage in its jurisdiction. Examples of regulators are the FCC and the NTIA in the US.
- SAS Server is a cloud server providing regulation-compliant spectrum allocation service to spectrum users.. Each SAS server is managed by a SAS administrator, the candidates of which include the current ones in CBRS (e.g., Google, Federated Wireless, CommScope, Sony) and other cloud service providers. SAS servers receive information about incumbent user appearance from federally authorized sensors
- Spectrum user operates a certified RF device and acts as a client to the SAS for spectrum access assignment. Typical spectrum users include private LTE base stations, 5G access points, and campus hot spots.
- Witness is an entity who participates in local spectrum management on behalf of its associated spectrum users. Candidates of witnesses include administrators or proxies (as in CBRS) of regulator spectrum users.

To ensure compatibility with both the existing and decentralized models, three more assumptions on the participants are needed. First, every SAS server operates independently when it comes to providing service to local spectrum users. Although in practice multiple SAS servers may be managed by one commercial entity, this independence can still be achieved at the local service level as we will elaborate in §6.4.3. Second, a spectrum user can be either standalone or associated to a witness, while a witness must be backed by one or more associated users, as is shown in Figure 6.1. This arrangement reflects flexibility in user participation, as a standalone user may not want to involve in SAS management other

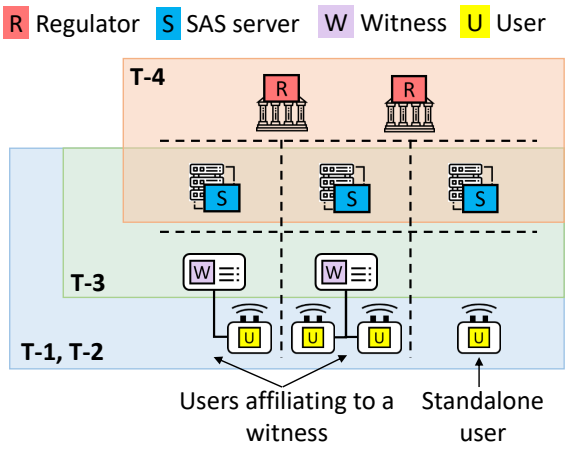


Figure 6.1: Task involvement of participants in the decentralized SAS model. Dashed lines represent trust boundaries.

than being a client. An individual user who wishes to get involved needs to operate her own witness. Third, we exclude federal incumbent users from the spectrum user category as they have preemptive access rights and do not need the spectrum allocation service from the SAS. The signalling mechanism of incumbent appearance is modelled as messages coming from a regulator node.

6.3.2 Main Tasks

Geographical Concepts. The global spectrum management applies to the entire spectrum jurisdiction, such as the territory of the US. In contrast, local spectrum access and management are specific to each region. All regions operate independently while being subject to global regulations. The region concept is inclusive of CBRS "zone" which typically refers to a US county [225] and a region may consist of one or more contiguous CBRS zones. Next, we define four main SAS tasks:

T-1. User Registration: The system accepts registrations from spectrum users. The registration process validates the user provided device information (including the device

certificate, region, and RF capability) and creates a unique user ID which will be used in later communications and tasks.

T-2. Access Assignment: The system generates spectrum access assignments (specifying location, channel range, expiration time, etc.) per user request. Calculation of the assignment follows a predefined interference model that computes over the current state of spectrum sharing. The system also allows a user to simply inquire the real-time spectrum availability (e.g., on location and channel range) before it decides operational parameters in its ensuing spectrum access request.

T-3. Record Keeping: The system keeps an irreversible record of spectrum access requests and assignments. In the decentralized SAS model considered in this paper, this task dictates that all SAS servers and witnesses who have participated in the access management for a specific spectrum region should agree on a unified record for that region. This record serves three purposes—for synchronizing SAS service states (required by the existing SAS [225]), enabling regulatory audits, and underpinning any compensation scheme.

T-4. Regulation enforcement: The system allows a regulator to publish regulations on the above tasks. A regulation can be either long-term or short-term. Long-term regulations include the addition of spectrum regions, designation of a universal interference model, publication of priority access licensees (as in the CBRS case), etc. Short-term regulations include notification of incumbent user appearance, addition/removal of SAS servers, etc.

The task involvement for participants is shown in Figure 6.1. Importantly, regulators do not involve in the day-to-day spectrum management business (T-1, T-2, T-3) while spectrum users act as clients for T-1 and T-2.

6.3.3 Key Requirements for Decentralization

In the current SAS model, T-1 and T-2 have been standardized as a server-client process [226] while T-3 and T-4 are trivial due to the trust on individual SAS servers. In comparison, the decentralized SAS model does not assume trust on any individual participant except the regulators. Spectrum users do not need to trust any individual SAS server, nor do SAS servers or witnesses trust each other. Moreover, the performance impact of decentralization should be kept in control. We identify the following key requirements:

R-1. Fault tolerance: All the four tasks should be executed correctly given that a certain portion of SAS servers are faulty or malicious (i.e., "Byzantine"). The detailed threat model is given in §6.3.4. In specific, for any spectrum region, users should get correct assignments (T-2) and SAS servers and witnesses should keep a unified assignment record (T-3) as long as fewer than half of SAS servers can be Byzantine.

R-2. Responsiveness: The generation of an actionable access assignment per user request (T-2) should be swift, and on par with the existing SAS' server-client model. Here "actionable" means the assignment is complete and valid before being written into the record. This implies that the assignment latency should be less than linear in the number of SAS servers involved in the assignment generation. Moreover, the system should accommodate the increasing number spectrum users without significantly compromising responsiveness.

R-3. Backward Compatibility: The system should be compatible with the existing SAS infrastructure and coexist with the server-client model when fulfilling T-1 and T-2. This offers flexibility to SAS servers amd spectrum users in their day-to-day operation, as they can gradually transition into the new decentralized model.

Table 6.1: List of system variables in BD-SAS

Symbol	Definition
$\mathcal{C}_R$	The Regulatory Contract on the G-Chain.
$\mathcal{C}_{SA}$	Spectrum Access Contract on an L-Chain.
$\mathcal{SG}_i$	Group of SAS servers serving the i^{th} L-Chain.
S_i	Size of $\mathcal{SG}_i$.
B	G-Chain block interval in seconds
T_i^E	i^{th} L-Chain's epoch length in B.
T_i^S	i^{th} L-Chain's shift length in B.
R_i	Number of epochs before a shift ends for the SAS server reshuffle procedure to start at the i^{th} L-Chain.

6.3.4 Threat Model

We assume individual SAS servers may suffer from Byzantine fault, i.e., arbitrarily deviating from its normal routine, due to server failure or adversarial corruption. We call unaffected SAS servers "honest". The adversarial corruption can be adaptive in that a SAS server may act honestly at first and turn malicious at an arbitrary point. In all cases, we assume the honest SAS servers always take up more than 50% of SAS servers at any time. We also assume witnesses have an honest majority in each local region.

6.4 The BD-SAS Architecture

6.4.1 BD-SAS Overview

We introduce the blockchain-based decentralized SAS architecture, dubbed BD-SAS, to realize the aforementioned decentralized SAS model. BD-SAS consists of two layers of blockchain networks: a single Global Chain (G-Chain) and region-specific Local Chains (L-Chains). We define curator as an entity who participates in the blockchain consensus and maintains a blockchain instance, and client as an entity who can create an account and issue transac-

tions but do not participate in consensus. Following this definition, the G-Chain is a public blockchain to which anyone can be a client; but only regulators and SAS servers can be the curators, with the former in charge of curator permission control. In comparison, the L-Chain of any region j is a fully permissioned blockchain, with the local witnesses and a group of SAS servers (denoted $\mathcal{SG}_j$) as curators, while local spectrum users act as clients. $\mathcal{SG}_j$ is responsible for user management and spectrum access assignment. Specially, we call the witnesses who founded the L-Chain and assumed long-term managerial responsibility the anchor witnesses, such the PA users in CBRS. Anchor witnesses also represent all other witnesses in the G-Chain. Anticipating the presence of an adaptive adversarial who can corrupt targeted SAS servers, $\mathcal{SG}_j$ needs to be periodically re-selected through a random reshuffle mechanism.

The data structures of the G-Chain and one L-Chain are shown in Figure 4.1. Both the G-Chain and L-Chain follow the account-balance model as in Ethereum [227] and support on-chain state machines that enable smart contract functionalities. The G-Chain's state contains the global account balances and a regulatory contract $\mathcal{C}_R$, which allows regulators to publish spectrum regulations and create new L-Chain profiles (T-4). $\mathcal{C}_R$ also encodes a record-keeping function that enables SAS servers to update local service states and claim service compensations (T-3) and a multiparty function realizing random reassignment of SAS servers. We will later show that the latter function is essential for our system in achieving resilience against adaptive SAS server corruptions. The SAS servers update the L-Chain state for every epoch, which spans over T_i^E consecutive G-Chain blocks. Every epoch start of the L-Chain is marked by a beacon block, whose header includes an extra hash pointer to the most recent G-Chain block. The SAS server group $\mathcal{SG}_j$ is randomly re-selected for every shift, which spaces over T_i^S consecutive G-Chain block cycles. T_i^E and T_i^S are design parameters and fixed during the i^{th} L-Chain's bootstrapping, and T_i^S is a multiple of T_i^E.

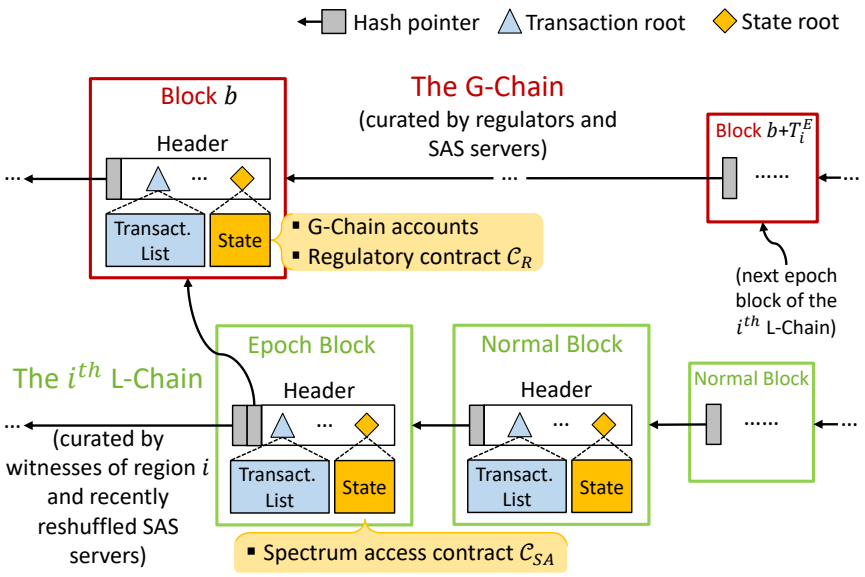


Figure 6.2: Ledger structures of the G-Chain and one L-Chain (each region has its own L-Chain) of BD-SAS.

An L-Chain's state contains a spectrum access contract $\mathcal{C}_{SA}$ that documents local witnesses, spectrum users, and assigned SAS servers. $\mathcal{C}_{SA}$ also encodes the functions for user registration (T-1) and spectrum access assignment (T-2). For every epoch, the assigned SAS servers are responsible for conveying recent regulations to the L-Chain at the epoch start (T-4) and updating the local service state to the G-Chain at the epoch end (T-3).

Next we focus on the practical design of G-Chain and L-Chain and elaborate on how the four tasks are fulfilled.

6.4.2 G-Chain Operation

The G-Chain can build on existing blockchain implementations through two paradigms: (1) reusing an existing public blockchain system such as Ethereum or its testnets or (2) bootstrapping from a new consortium blockchain network. The former paradigm builds on the reliability of the public blockchain's smart contract functionality, while the latter gives

freedom for choosing more efficient consensus protocols. We adopt the first paradigm for prototyping (see §6.7). At the starting phase, regulators and an initial group of SAS servers start with a G-Chain block, where a regulator submits a G-Chain transaction to create $\mathcal{C}_R$. New SAS servers join the G-Chain network by connecting to the URL endpoints of existing regulators and SAS servers, and passing TLS-based mutual authentication ased on an existing public key infrastructure (PKI), which is conveniently provided in the existing SAS ecosystem [224, 225]. SAS servers can synchronize to the G-Chain ledger and interact with $\mathcal{C}_R$ after passing the mutual authentication procedure with a regulator.

Encoding $\mathcal{C}_R$. The pseudocode of $\mathcal{C}_R$ is shown in Algorithm 11. Its public variables include the profiles of regulators, SAS servers, L-Chains, as well as interference model parameters and an incidence list for incumbent appearances. A regulator can update the above public variables by sending a transaction invoking the Publish function, which fulfills the regulation publication part of T-4. The ReshufflePropose and ReshuffleConfirm functions are parts of the SAS server reshuffle procedure as we elaborate in §6.4.3.

Local Update Procedure (T-3): When concluding an epoch, the SAS servers in $\mathcal{SG}_i$ needs to update their local service state onto the G-Chain, fulfilling the global part of T-3. To do so, they use a simple round-robin mechanism across epochs—each of the S_j SAS servers take turns to call the LocalUpdate function of $\mathcal{C}_R$ for every new epoch. The function updates the epoch, L-Chain state root *sroot*. Every anchor witness of L-Chain j check the new updated *sroot* with their local version, and calls LocalUpdate to provide its approval signature.

All G-Chain participants can send tokens to each other through transactions, akin to cryptocurrency payment. The anchor witnesses and SAS servers of an L-Chain can also use the local update procedure to claim compensation for their L-Chain spectrum access

Algorithm 11: Regulatory Contract $\mathcal{C}_R$ Pseudocode (essential functions only)

```
1  var Regulators[], Servers[]
2  var LC[].{desc, Anchors[], Witnesses[], SG[], shift, epoch, sroot} //L-Chains profiles
3  var iModel.{para1, para2, ...} //Interference model
4  var Incidents.{VacateOrders[], ...}
5  var Lottery[][].{hash, proof, shift, status}
6  Function Init() // Contract creation by a regulator
7      Initialize Regulators, Servers, iModel ;
8  Function Publish(regulation) // S represents the function caller
9      if S ∈ Regulators then
10         Update Servers[], LC[], iModel, or Incidents per regulation ;
11 Function ReshufflePropose(j, hash, proof) // j is the target L-Chain index
12     if S ∈ Servers and system.block − LC[j].shift ∈ [0, T_j^S − T_j^E) and
         Lottery[[j]S].shift ≤ LC[j].shift then
13         Lottery[j][S].{hash, proof, shift} ← {hash, proof, LC[j].shift + T_j^S} ;
14 Function ReshuffleConfirm(j, selectedSG[]) // Invoked by L-Chain j's witnesses
15     if S ∈ LC[j].Witnesses and system.block − LC[j].shift ∈ [T_j^S − T_j^E, T_j^S) then
16         if the majority of LC[j].Witnesses confirm selectedSG[] then
17             LC[j].SG[] ← selected[] ;
18             LC[j].shift ← LC[j].shift + T_j^S ;
19 Function LocalUpdate(j, epoch, sroot) // Invoked by L-Chain j's SAS server group
20     if S ∈ LC[j].SG and epoch ∈ [LC[j].epoch, system.block] then
21         if the majority of LC[j].SG update on the same LC[j].{epoch, sroot} then
22             LC[j].{epoch, sroot} ← {e, sroot} ;
```

assignment service (not shown in Algorithm 11). In this paper, we consider the compensation scheme an important but standalone task to be addressed in a separate work.

6.4.3 SAS Server Reshuffle Procedure

We require the SAS server group $\mathcal{SG}_j$ for each L-Chain j to be randomly replaced for every shift (i.e., T_j^S G-chain blocks). This is crucial to our system's resilience against adaptive corruptions on SAS servers at the local level, and to avoid that the SAS servers of one L-Chain belong to the same SAS administrator. This procedure takes advantage of the cryptographic primitive verifiable random function (VRF).

VRF was first introduced by Micali et al. [146] in 1999 for providing both unpredictability and verifiability of pseudo-random functions. It allows a function caller i to generate a random $hash$ and a proof π with its private key sk_i. The proof π allows others to verify the validity of the hash using the caller's public key pk_i. The randomness of $hash$ means it looks uniformly distributed to others without knowing π. Specifically, VRF generation and verification are as follows:

- $\langle hash, \pi \rangle \leftarrow \text{VRF}_{sk_i}(role||seed)$
- $decision \leftarrow \text{VerifyVRF}_{pk_i}(role||seed, hash, \pi)$

wherein $role$ is a descriptor, such as 'sas_server_#i', and $seed$ is a public random seed known to the system. $decision$ is a binary value taking true or false.

VRF-based SAS Server Reshuffling. Back to our system, SAS servers participate in the reshuffle procedure individually. If SAS server i is interested in joining $\mathcal{SG}_j$ (i.e., serving L-Chain j) for the next shift, at the start of the R_j^{th} last epoch of the current shift, SAS server i needs to do the following:

- Generate a $hash_i$ and a proof π_i by executing

$$\langle hash_i, \pi_i \rangle \leftarrow \text{VRF}_{sk_i}(\text{'sas\_server\_}i\text{'}||sroot) \tag{6.1}$$

  where $sroot$ is the L-Chain j's state root of the last shift;

- Submit $hash_i$ and π_i to G-Chain by calling $\mathcal{C}_R$'s ServerReshuffle function which updates the $Lottery[j][i]$ variable.

When the last epoch starts, each of L-Chain j's anchor witnesses collects the available hash-proof pairs from $Lottery[j][\cdot]$ from $\mathcal{C}_R$ and performs the following steps:

- Sort the hash-proof pairs by hash value in ascending order;
- Along this order, for every hash-proof pair $\langle \pi_i, hash_i \rangle$ (denote SAS server i the generator), verify it by performing

$$decision \leftarrow \text{VerifyVRF}_{pk_i}(\text{`}sas\_server'||sroot, hash_i, \pi_i) \tag{6.2}$$

- If $decision$ = true, add i to $selected[]$; then continue the last step for the next hash-proof pair in the order;
- Terminate when $selected[]$ has S_j entries, and submit $selected[]$ to $\mathcal{C}_R$ by calling ReshuffleConfirm.

Lastly, ReshuffleConfirm updates $\mathcal{SG}_j$ in $\mathcal{C}_R$ only when the majority witnesses of L-Chain j also sign for the same $selected[]$. A newly updated SAS server i for L-Chain j will need to connect to the anchor witnesses through TLS communication before the start of the new shift, under the PKI established by the regulator (as is in CBRS [224]).

6.4.4 L-Chain Operation

L-Chain is a special-purpose permissioned blockchain where the SAS servers and witnesses have different responsibilities in the ledger curating process, with the former providing transaction execution service (thus enforcing state changes from the last block) while the latter providing transaction ordering service (thus finalizing new blocks). Both SAS servers and witnesses maintain the L-Chain ledger. At the bootstrapping phase, the genesis block is

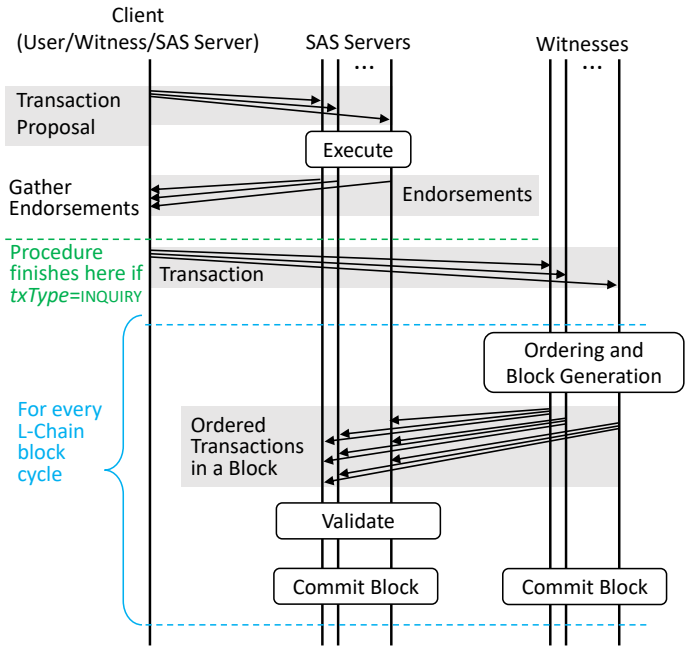


Figure 6.3: L-Chain workflow built on Hyperledger Fabric's execute-order-validate model [7].

established among the initial anchor witnesses and SAS servers, who are regulator-certified entities and published in $\mathcal{C}_R$. The genesis block serves the L-Chain's first beacon block with a hash pointer to the most recent G-Chain block. The genesis block also comes with an initial $\mathcal{C}_{SA}$ setup, specifying the region information and initial anchor witnesses. For the CBRS band and others which have priority licensees, the initial anchor witnesses can be conveniently undertaken by the priority licensees. Since L-Chain is permission-controlled, a new witness must first operate a valid spectrum user (or the proxy for a group of users) and pass the user registration process.

Separation of Execution and Consensus. We adopt a separation paradigm for L-Chain transaction execution and consensus making, where SAS servers and witnesses undertakes transaction execution and transaction serialization respectively. The separation concept was originally proposed in [236] to reduce server redundancy and increase modularity in state

machine replication systems, and later adapted to permissioned blockchain frameworks such as Tendermint and Hyperledger Fabric [7] for efficiency and modularity purposes. It is in contrast to traditional blockchain consensus that relies on an atomic consensus module, such as Ethereum, where transaction ordering and execution are performed in one operation. While both paradigms can realize a blockchain state machine and smart contracts and yield similar throughput performance, the separation paradigm supports client-in-the-loop endorsement mechanism which provides better architectural modularity, flexibility, and low latency of obtaining service. To realize this design in L-Chain, we leverage Hyperledger Fabric [7] and one L-Chain transaction workflow is shown in 6.3. A client, e.g., a spectrum user, sends out a transaction proposal containing input to an L-Chain contract. The SAS servers execute the contracts and return the outputs and state changes as endorsements. The client need to collect more than half endorsements before sending out the real transaction for block generation. After the witnesses generate a block containing serialized transactions, they broadcast the block to SAS servers who will validate the block proof and all other endorsements before committing the block. The witnesses are only responsible for the block generation (i.e., transaction ordering), who also commit the block to their local L-Chain ledgers.

From the deployment perspective, this separation paradigm is key to the system's responsiveness in processing spectrum access requests (R-2) as it capitalizes on the physical differences between SAS servers and local witnesses. First, SAS servers have presumably sufficient computing capability for executing transactions. Meanwhile, local witnesses are not necessarily powerful computers, as they represent diverse types of spectrum users. Second, local witnesses tend to physically reside in the region where their associated users access the spectrum. Their close proximity yields low communication delays, which enables the usage of deterministic consensus schemes such as Raft [162] or efficient BFT protocols [197, 237] for the ordering task. In comparison, we do not assume any locality of SAS servers and they

may reside in a cloud cluster across the country.

Encoding $\mathcal{C}_{SA}$. The spectrum access contract $\mathcal{C}_{SA}$ profiles local spectrum participants (i.e., witnesses and users) and enables SAS servers to provide consistent spectrum management service. $\mathcal{C}_{SA}$ pseudocode is shown in Algorithm 12. At the start of a shift, everyone in the newly selected SAS server group $\mathcal{SG}$ calls the ShiftUpdate function indicating its incoming service, which needs approvals from the majority witnesses. At each epoch onset, $\mathcal{SG}$ fetches latest regulatory information from $\mathcal{C}_R$, including updates of interference model parameters and anchor change confirmation in case an anchor witness was changed by the regulator. Adding new or removing existing witnesses require the majority of existing witnesses invoking the WitnessUpdate function. Next we provide details $\mathcal{C}_{SA}$ fulfil two essential spectrum management tasks.

User Management (T-1): For a new spectrum user to register with the L-Chain, it needs to pass the off-line device registration procedure (e.g., the CBSD registration protocol [226]) with each connected SAS server. After which the servers add this user by calling UserUpdate with a majority vote. User removal also is accomplished through UserUpdate, via either the user's own action or the majority of SG.

Access Assignment (T-2): A spectrum user submits its spectrum access request by invoking the Request function along with operational parameters, including the requested zone, channel range, and effective isotropic radiated power (EIRP). This function encodes a channel assignment algorithm that calculates a decision based on the existing user's device parameter, operational parameters, as well as current channel availability, through a predefined interference model (i.e., $iModel$ in Algorithm 12). For this $iModel$ and later implementation, we adopt a simple first-come-first-serve allocation criterion, where the assignment is approved if the requested channel range at the request zone is available and

the user's EIRP would not cause harmful interference to users of neighboring zones. More complex channel allocation schemes exist in the literature, as we discuss in §6.8. Besides the Access request type, there also exist Inquiry and Heartbeat request types, which enable spectrum user to perform simple inquiries on readily available channels and provide periodic proofs of liveness (per the existing SAS in CBRS) respectively. Our implementation in §6.7 also included all three types of request to demonstrate this backward compatibility.

Performance Features. In terms of L-Chain performance, as long as the user's requests are less frequent than L-Chain's block frequency, the response can be finalized by $\Omega(1)$ scale of the L-Chain block interval. In implementation we set L-Chain's block interval to one second, while in current SAS implementation the tightest response timing requirement (a.k.a. response to Heatbeat) should be done within 60 seconds [97, 226]. we require all transactions be finalized in 5 blocks or otherwise fail. On the other hand, if a spectrum user has a high level of trust on the availability of its local witnesses, they can start using the allocating channels right after receiving the majority SAS server endorsements, just like the receiving an Inquiry response.

6.5 Analyses

Next we analyze the security and performance properties of BD-SAS under the aforementioned threat model (§6.3.4). We assume there are N SAS servers globally, F of which are potentially Byzantine. Every L-Chain's SAS server group size is fixed to S and witness population W.

Proposition 6.1 (Fault tolerance). As long as $N > 2F$ (i.e., honest majority), G-Chain operations are safe against Byzantine SAS servers. The same fault tolerance also applies to

L-Chain except for exponentially diminishing probability as S increases.

Proof. The G-Chain's tolerance against Byzantine SAS servers follows from the underlying blockchain consensus which is secure under the honest majority assumption. L-Chain's tolerance derives from the verifiable randomness of the SAS server reshuffle procedure (§6.4.3) executed for every shift. The chance that the majority of the S SAS servers are Byzantine is $(\frac{F}{N})^{\lceil S/2 \rceil}$, which diminishes exponentially as S increases since $N > 2F$. □

Remark 6.2. The liveness aspect of of L-Chain security, i.e., all valid L-Chain transactions should be eventually finalized in the ledger, depends on the availability of the local witnesses and the consensus protocol they run for the ordering task. We consider it a standalone challenge as the witnesses can internally manage their own trust levels and choice of consensus protocol as long as there are an honest majority. In the implementation we stick to Fabric's Raft consensus for prototyping.

Proposition 6.3 (Adaptive corruption resilience). Both G-Chain and L-Chain operations are resilient to an adaptive adversary who can corrupt any SAS server in the time scale of shifts.

Proof. The G-Chain's resilience against adaptive corruptions on SAS servers follows as long as the honest majority assumption holds at all times. Meanwhile, L-Chain's adaptive corruption resilience is ensured by the timing regime of the SAS server reshuffle procedure (§6.4.3). That is, SAS servers call the ReshufflePropose function only at the start of the R-th last G-Chain blocks of before the current shift ends. The adversary has no additional information (i.e., no different than random guessing) on whether a SAS server will be chosen by an L-Chain. Similarly, the adversary's successful corruption on certain SAS servers for the current shift does not last into the next shift. □

Proposition 6.4 (Responsiveness). Given that each spectrum user submits spectrum requests (i.e., inquiry, access, or heartbeat) less frequently than the L-Chain block's maximum trans-

action throughput, the spectrum user can get an access assignment (or an inquiry/heartbeat response) within the time scale of $\Omega(1)$ seconds.

Proof. This follows from L-Chain's separation paradigm as is shown in Fig. 6.3. A user is ready to propose a formal transaction when its transaction proposal collected endorsements from the majority of the S SAS servers, within $O(S)$ TLS communication sessions. This TLS communication performance is on par with the server-client model of existing SAS, since the endorsements can be collected in parallel. Meanwhile, an assignment is not finalized until passing the ordering and block generation phase which is handled by the witnesses. The finalization can be done by the next L-Chain block cycle as long as users' requests do not overwhelm the L-Chain block's maximum transaction throughput. □

6.6 Implementation

We used the Ethereum Rinkeby testnet (https://rinkeby.etherscan.io/) to emulate the G-Chain (i.e., a public blockchain) and the Hyperledger Fabric platform to implement the L-Chain prototypes. The reason for choosing Rinkeby to emulate G-Chain is two-fold. First, it offers complete smart contract tool set (identically to that in the Ethereum main blockchain). Second, it has a static $B = 15s$ block interval which can act as a stable timing source for BD-SAS's epoch and shift changes. We implemented the G-Chain regulatory contract $\mathcal{C}_R$ in Solidity with 202 lines of code. For the SAS server reshuffle procedure, we used the VRF implementation of [155] which is based on the elliptic curve signature system ed25519 with 32-byte private/public keys, 64-byte hashes, and 80-byte proofs.

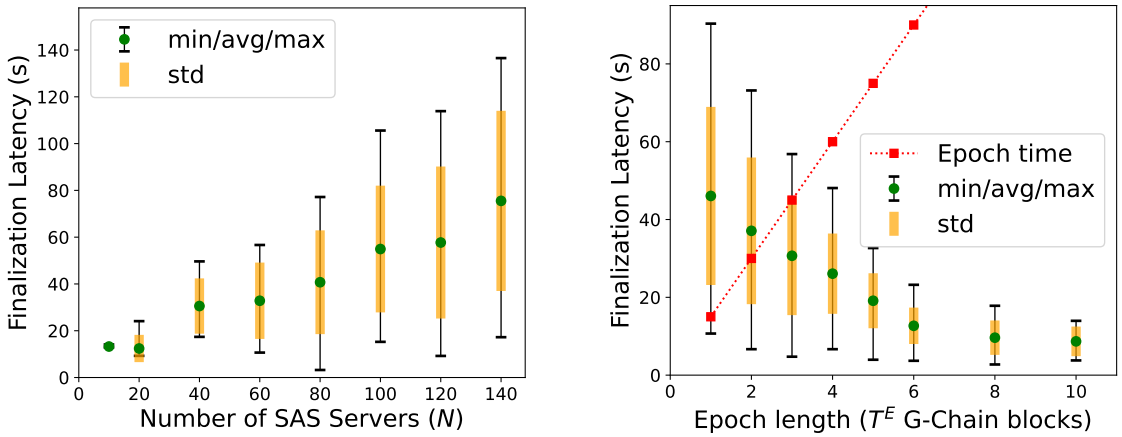
Our L-Chain implementations consisted of $\{3, 5, 7, 9\}$ SAS servers, each with fixed 5 witnesses, all of which were instantiated in docker containers in a Linux testbed on top of a AWS

T2.xlarge machine. The L-Chain block interval was fixed to 1 second and and every block could enclose up 1MB of transactions. Fabric's native Raft consensus (a crash-fault-tolerant protocol) was used for the witnesses' ordering task. The epoch time T^E took value from $\{1, 2, ..., 10\}$ (in B) and the shift time T^S was fixed to 1000 (in B), where L-Chain index is left out for convenience. We implemented the spectrum access contract $\mathcal{C}_{SA}$ in Golang with 234 lines of code. $\mathcal{C}_{SA}$ is designed to enables three types of spectrum access operations: Access—which returns a channel assignment, Inquiry—which returns the information of a readily available channel-location, and Heartbeat—which is called periodically by a spectrum user to demonstrate liveness and returns an authorization for continuing channel usage. It is noted that our $\mathcal{C}_{SA}$ implementation adopts a straightforward first-come-first-first channel allocation model and does not consider multi-user interference coordination when calculating an assignment, which would need an optimization-based allocation technique. We will explore this option in future work.

6.7 Evaluation

6.7.1 G-Chain Performance and Feasibility

We evaluated the feasibility of Rinkeby as G-Chain by testing its response latency for fulfilling two key G-Chain operations: reshuffle and local update. The first operation involves all N SAS servers calling the ReshufflePropose function of $\mathcal{C}_R$ simultaneously, simulating a squeeze situation that stress-tests the G-Chain, which in practice will also provide the highest security against adaptive attacks since the VFR results are exposed with minimal time. It is observed from Figure 6.4a that the finalization latency of all ReshufflePropose calls generally increases linearly with N. This indicates that we would need to increase the epoch length given the



(a) Calling ReshufflePropose (squeeze)

(b) Calling LocalUpdate (regular)

Figure 6.4: Stress testing transaction finalization latency of Ethereum Rinkeby testnet (as G-Chain). (a) A once-a-shift squeeze scenario where all N SAS servers call $\mathcal{C}_R$'s ReshufflePropose simultaneously. (b) The regular operation scenario where N (=100) SAS servers (representing 5 L-Chains) call $\mathcal{C}_R$'s LocalUpdate intermittently across an epoch.

larger N, for having the reshuffle procedure finished in one epoch. For example, if $N = 140$, we would need to set a minimum epoch time of 10 G-Chain block cycles (i.e., 150s). In comparison, the second operation involves $N = 100$ SAS servers, representing 20 L-Chains, calling the LocalUpdate function $\mathcal{C}_R$ periodically but at different instants, simulated an uncongested and orderly situation. It is observed from Figure 6.4b that only when $T^E \geq 4$ (i.e., 60 seconds) can the G-Chain ensure every update be confirmed within one epoch.

For the SAS server reshuffle procedure, we benchmarked the VRF operations on a Linux machine with 16GB memory, 3.0GHz CPU frequency Linux machine. It took 537ms to generate the VRF hash and proof, and 441ms to verify them. We can see that these off-chain delays added to the reshuffle procedure are negligible compared to time cost of their on-chain part.

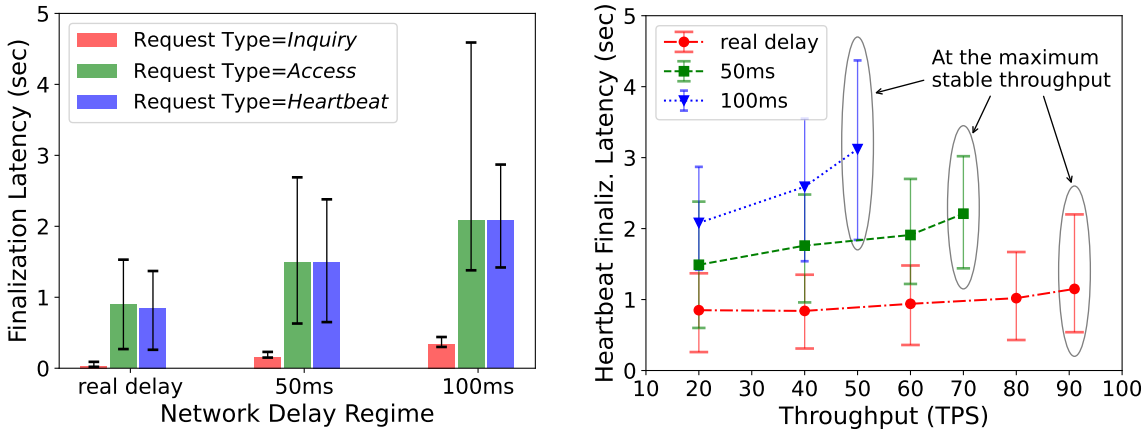
6.7.2 L-Chain Performance

Next we evaluate the performance for our L-Chain prototypes with respect to two metrics: the finalization latency of a spectrum request and the throughput capacity for the L-Chain to handle a certain volume of such requests. In particular, we evaluated all three types of request: Inquiry, Access, and Heartbeat. The tests on processing Inquiry and Access requests were configured with a fixed invocation frequency at 20 transactions per second (TPS). The tests on Heartbeat requests were configured with varying invocation frequencies from 20 TPS to the maximum stable throughput (i.e., all transactions result in success). This emulates the practical case that the Heartbeat requests would be much more frequent, potentially than the other two. We used Hyperledger Caliber [107], a blockchain benchmarking tool, to simulate the abovementioned transaction traffic which invokes the Request function with the corresponding request type. Moreover, to simulate network delays in the Internet, we used three delay regimes to L-Chain participants:

- *real*: the end-to-end packet delay statistics measured among commercial cloud servers across the US. This delay regime was only used on the L-Chain with 5 SAS servers. The measurement and L-Chain node resemblance are shown in Table 6.2.

- $50ms$ or $100ms$: the synthetic end-to-end packet delay added uniformly to every pair of L-Chain participants. These two delay regimes were used on the L-Chain with $S \in \{3, 5, 7, 9\}$ SAS servers.

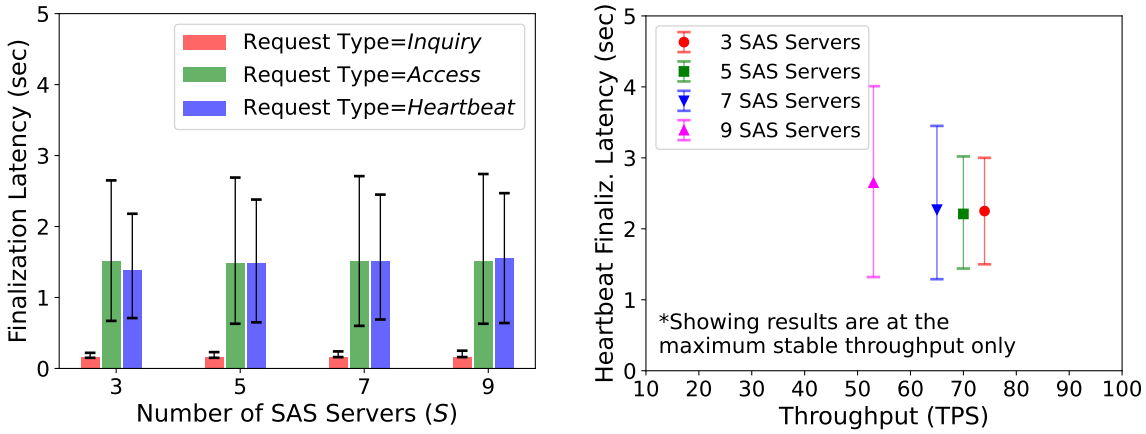
We first evaluated the performance of one L-Chain with 5 SAS servers under all three delay regimes, as is shown in Figure 6.5. Figure 6.5a shows the finalization latency of three types of requests with the request frequency fixed at 20 TPS. We observe that even under the harshest $100ms$ delay regime, the system is able to finalize an Inquiry request, an Access



(a) At constant 20 TPS throughput.

(b) Heartbeat stress test.

Figure 6.5: Performance of the L-Chain with 5 SAS servers under three different delay regimes.



(a) At constant 20 TPS throughput.

(b) Heartbeat stress test.

Figure 6.6: Request finalization latency (min/avg/max) of L-Chains with $S \in \{3, 5, 7, 9\}$ SAS servers under the $50ms$ delay regime.

request, or a Heartbeat request within 0.5s, 5s, or 3s respectively. The higher latency variation in finalizing the Access request was likely caused by the varying size of channel-location availability slots in our $\mathcal{C}_{SA}$ implementation, while an Inquiry or Access request invokes constant look-ups. Figure 6.5b shows the finalization latency of Heartbeat requests only but with varying throughput pressure. We observe that higher delays reduces the maximum state throughput, which is intuitive in that higher delays decrease the system's overall service efficiency. In both Figure 6.5a and Figure 6.5b, it is observed that under the real delay scenario and within the stable throughput region, the three types of requests can be finalized within 2 seconds. This latency performance demonstrates the L-Chain's feasibility in fulfilling the existing CBRS ecosystem's timing requirement, in which the Heartbeat interval is typically set to 30 or 60 seconds [97].

We further investigated the performance implication of SAS population S. The results for L-Chains with four different S under the same $50ms$ delay regime are shown in Figure 6.6. Figure 6.6a shows that under the stable throughput scenario, S has negligible impact on the finalization latency. Meanwhile, Figure 6.6b shows that larger S does become a constraining factor on the maximum stable throughput and also on the finalization latency with the request invocation rate approaches the maximum stable throughput. Considering that a larger S provides a higher level of adaptive attack resilience (see Proposition 6.3), we will need to choose a practical S in the design phase to strike a balance between security and throughput capacity in practical deployment.

6.8 Discussion

Optimality of Spectrum Allocation. The current spectrum assignment function, as is encoded in our spectrum access contract (Algorithm 12) and implemented in our prototype, is a straightforward first-come-first-serve allocation scheme. In practice, spectrum allocation can deploy a much more complex, hopefully optimization-based algorithm to guarantee succinct spectrum assignment with specific service requirements such as on fairness, bounded response time, or channel utilization rate, as were indicated in prior arts [15, 108, 139, 239]. However, instantiating such optimization-based allocation algorithm on blockchain smart contract would incur very high on-chain cost, as every invocation of the algorithm would be executed and stored by every L-Chain curators. To cope with this scalability constraint, we are eyeing on two potential solutions: (1) customizing optimization-based allocation algorithms into minimal yet effective versions so they are suited for blockchain deployment; (2) using dedicated secure environments, such as a hardware-based trusted execution environment (TEE) [57, 181], to execute the algorithms off-chain and return the result on-chain using a secure commit protocol [51, 231].

Privacy Challenge. The current BD-SAS's spectrum access assignment function is executed in an transparent, on-chain manner in that every L-Chain curator knows each user's assignment record. This can be a disincentive for privacy-aware spectrum users, who may not wish to share their access or device information to other witnesses, whose associated users also reside in the same local region. To provide privacy protection to spectrum users, aside from the off-chain TEE solutions mentioned above, we could employ obfuscation mechanisms, such as those with differential privacy [73, 76], to allow users control their privacy leakage. Moreover, using efficient on-chain multiparty computation is also a potential direction to explore.

6.9 Conclusion

We formulated a decentralized SAS model for fault-tolerant and automated dynamic spectrum sharing and introduced the BD-SAS architecture as a concrete solution. BD-SAS consists of two layers of blockchains: a G-Chain for global regulation and SAS server management at and L-Chains for providing spectrum access assignment to spectrum users in the local regions, without having the users place trust individual SAS servers. We implemented a BD-SAS prototype using Ethereum Rinkey testnet to emulate the G-Chain and Hyperledger Fabric platform to implement the L-Chains. The result demonstrated the feasibility of G-Chain in performing the critical security mechanisms including SAS server reshuffling as well as the responsiveness of L-Chain in generating spectrum access assignments amid stringent timing requirements. Lastly, we identified challenges for BD-SAS and directions for future extension.

Algorithm 12: Spectrum Access Contract $\mathcal{C}_{SA}$ Pseudocode (essential functions only)

var $Zones[].\{id, desc, licensees\}$, $ZCH[][].\{zone, status\}$ // Readily available zone-channel
var $iModel.\{schema, para1, para2, ...\}$ //Interference model
var $Witnesses[].\{type\}$ //*type* is anchor or normal
var $Servers[]$, $NS[]$ //Next SAS server group
var $Users[].\{tier, opParam\}$ //Users profile
var $Assignment[].\{user\_info, zone, chs, \text{ERP}, status\}$ //Access assignment, indexed by user id
Function IntfModel(z, c, EIRP) //Internal use
    Checks if EIRP is high enough to cause harmful interference to neighboring zones of z on channel c, using parameters specified in $iModel$. If yes, return fail; otherwise textbfreturn success;
Function Init() // Contract creation (omitted for brevity)
Function EpochUpdate() // Allows servers to update variables by voting (omitted for brevity)
Function ShiftUpdate() // Witnesses accept new SAS servers by voting (omitted for brevity)
Function WitnessUpdate(tp, $users$, $action$, p)// Update by voting (omitted for brevity)
Function UserUpdate(u, $desc$, $opPara$, $action$) // $\mathcal{S}$: function caller
    if $\mathcal{S} \in Servers$ and $action =$ add then
        Add $User[\mathcal{S}].\{desc, par\}$ if the function is called by the majority SG ;
    if $act =$ remove then
        Remove $Users[u]$ if $u = \mathcal{S}$ or the function is called by the majority SG ;
Function Request($desc$, $type$, $zone$, chs, EIRP) // *type*: Inquiry/Access/Heartbeat
    if $\mathcal{S} \in Users$ and $type = Inquiry$ then
        return $ZCH[z][c].status$ for all $c \in chs$;
    if $\mathcal{S} \in Users$ and $type = Access$ then
        if $\exists c \in chs$ that $ZCH[zone][c] =$ occupied or IntfModel($zone$,c,EIRP)=fail then
            return fail;
        else
            $A[\mathcal{S}].\{user\_info, zone, chs, \text{EIRP}, status\} \leftarrow \{desc, zone, chs, \text{EIRP}, \text{assigned}\}$;
            $ZCH[zone][chs] \leftarrow$ occupied;
    if $\mathcal{S} \in Users$ and $type = Heartbeat$ then
        $A[\mathcal{S}].\{user\_info, status\} \leftarrow \{desc, \text{authorized}\}$;

Table 6.2: Round trip times (in millisecond) among real cloud nodes and the node resemblance to L-Chain entities.

Node	Location	L-Chain Resemblance	1	2	3	4	5	6	7
1	AWS (N.Virginia)	Local Witnesses	-	-	-	-	-	-	-
2	Lab (N.Virginia)	Local Users	2.4	-	-	-	-	-	-
3	AWS (Ohio)	SAS Server 1	11.8	12.0	-	-	-	-	-
4	AWS (California)	SAS Server 2	62.0	82.3	50.1	-	-	-	-
5	AWS (Oregon)	SAS Server 3	66.9	71.3	49.5	21.8	-	-	-
6	Google (Nevada)	SAS Server 4	62.2	73.5	62.2	18.6	40.3	-	-
7	Google (Utah)	SAS Server 5	52.8	66.6	52.9	20.8	41.1	20.0	-

Chapter 7

Summary and Future Work

7.1 Summary

The decentralization and secure-by-design property promised by blockchain technology is unprecedented and truly inspiring. Meanwhile, there are still many challenges to address to clear the way for the technology's wider adoption. In this work we have addressed a number of blockchain's fundamental security issues from a cross-layer perspective and developed novel applications that leverage blockchain to solve domain-specific security problems.

In Chapter 2 we presented a comprehensive review on a wide range of popular consensus protocols for blockchain networks. Based on a novel five-component framework, we provided a taxonomy on different protocol classes and security analysis on each class. We hope the five-component framework, taxonomy, protocol abstractions, and succinct tutorial on protocol design can help researchers and developers grasp the key security-performance trade-offs in blockchain consensus and facilitate future protocol design.

In Chapter 3 we studied the security of a key component of blockchain core—the consensus protocol. Through modeling analysis on the impact of a node, we were able to quantify the relationship between the heterogeneity of network connectivity and the several key blockchain consensus security metrics, including fork rate and mining revenue distribution. We hope this model can provide a methodology of analyzing blockchain system security

from the cross-layer perspective.

In Chapter 4 we developed a decentralized truth discovering data feed system called DecenTruth to tackle the external data truthfulness challenge facing blockchain DApps. DecenTruth harmonized truth discovery—a data mining technique—with asynchronous Byzantine fault tolerant consensus to enable decentralized nodes to collectively discover truths about external data objects while being resilient to malicious nodes and sources.

To demonstrate the applicability and extensibility of blockchain to solving wider societal challenges, in Chapter 5 we combined blockchain smart contract and trusted execution environment (TEE) technology to tackle a long-standing data privacy challenge, namely data usage control. With blockchain as a contract platform and TEE for secure offloading on-chain data operations, the proposed PrivacyGuard system can hopefully enable a vibrant data economy in which individual data owners have full control over the access and usage of their private data.

As a novel application of blockchain in wireless networking, in Chapter 6 we developed a blockchain-based decentralized spectrum access system, called BD-SAS, as an alternative to the existing centralized SAS paradigm. With two layers of blockchain distributions, BD-SAS enables automatic regulation-compliant spectrum management services and inter-SAS synchronization while tolerating individual SAS server failures. BD-SAS stands as a new direction towards reliable, policy-compliant spectrum management in a low-trust environment.

7.2 Future Work

7.2.1 Analyzing Blockchain Consensus Security in Practical Network

Just like the Internet, public blockchain systems are constantly evolving. In Chapter 3 we explored the relationship between consensus security and topology of blockchain networks under two adversarial mining cases. The analyses essentially rely on the availability of exact block propagation data, which will be effective for analyzing real-world blockchain systems only if the network structure is static and white-box, down to the link level. However, due to the sheer amount of nodes with different network locations and dynamicity of public blockchain systems, it is extremely difficult to generate a complete network propagation map. In practice, measurement of public blockchain network only yield propagation data on a more coarse level, such as the point-to-point delays (of block transmission) between limited amount of known nodes and the overall network throughput (of block finalization). Moreover, public blockchain networks are subject to constant structural changes as peers join and leave.

To bring our modeling analysis to the practical level, we will extend our model to the scenario where only partial network connectivity information is available. We plan to explore new game-theoretic methods to model the communication capability of nodes at the group of distribution level. The following assumptions can be made:

- A node or a colluding pool may only estimate its centrality in the network from partially available measurements of point-to-point block transmission delays. This copes with the above consideration that knowing the entire propagation map of a public blockchain network down to the link level is extremely difficult.
- A node or a colluding pool may strategically choose a mining strategy that maximizes

its potential mining gain, based upon its centrality measurement and the statistical inference of other nodes' behavior. Due to the sheer size of a public blockchain network, we assume the latter information is gathered from a stationary distribution rather than at an individual level, and then classical game-theoretic formulations such as mean-field games can be used for equilibrium analysis.

This new analytical model will be particularly useful for individual nodes to assess its mining strategy and profitability as well the overall consensus security of the blockchain network. The outcome will be compared to our existing analysis described in Chapter 3 and other concurrent studies [69, 92, 178].

7.2.2 Efficient and Robust Consensus Mechanisms

For both blockchain and classical distributed computing networks, the consensus protocol underpins the network's security and performance. When we apply blockchain paradigms to other networking scenarios, however, the drastically different networking conditions may render the existing Internet-based consensus protocols useless. We plan to develop efficient and robust consensus mechanisms customized to certain networking/synchrony profiles, leveraging efficient cryptographic primitives, fault-tolerant protocol design, and network analysis. There are two research tasks, with respect to theory and practice:

- The first task resides in the theoretical realm. We will develop flexible consensus mechanisms that can self-adapt to dynamic network conditions (of both the P2P overlay and the physical network) for the optimal performance, for instance, to attain the highest possible transaction throughput and fault tolerance while weathering temporary network partitions.

- The second tasks involving implementing consensus protocol for a wide range of practical networks that can potentially use consensus or consistency mechanisms. In particular, the flexible, self-adaptive consensus mechanism mentioned above could be particularly useful for ad hoc communication scenarios such as UAV formation control and automotive platooning.

7.2.3 New Decentralized Architectures and Applications

The blockchain paradigm has great potential in a wide range of applications not limiting to financial ones. With PrivacyGuard and BD-SAS as examples, we will continue exploring novel decentralized architectures and applications that can prevail in a zero-trust environment while emulating the performance of their centralized counterparts. Potential applications that can be realized in a decentralized manner include trustworthy peer-to-peer knowledge aggregation (extension from Chapters 4 and 5), identity and data management for 5G/nextG mobile networks and IoT, and non-cooperative spectrum sharing (extension from Chapter 6, which currently falls into cooperative spectrum sharing). This research direction is potentially interdisciplinary, involving applied cryptography, distributed systems, cyber-physical systems, security and privacy policy, and economics.

Bibliography

[1] Michael Abd-El-Malek, Gregory R Ganger, Garth R Goodson, Michael K Reiter, and Jay J Wylie. Fault-scalable byzantine fault-tolerant services. ACM SIGOPS Operating Systems Review, 39(5):59–74, 2005.

[2] Ittai Abraham, Dahlia Malkhi, et al. The blockchain consensus layer and bft. Bulletin of EATCS, 3(123), 2017.

[3] John Adler, Ryan Berryhill, Andreas Veneris, Zissis Poulos, Neil Veira, and Anastasia Kastania. Astraea: A decentralized blockchain oracle. In 2018 IEEE international conference on internet of things (IThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData), pages 1145–1152. IEEE, 2018.

[4] Bowen Alpern and Fred B Schneider. Defining liveness. Information processing letters, 21(4):181–185, 1985.

[5] Yackolley Amoussou-Guenou, Antonella Del Pozzo, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. Correctness and fairness of tendermint-core blockchains. arXiv preprint arXiv:1805.08429, 2018.

[6] Sébastien Andreina, Jens-Matthias Bohli, Ghassan O Karame, Wenting Li, and Giorgia Azzurra Marson. PoTS-a secure proof of TEE-stake for permissionless blockchains. IACR Cryptology ePrint Archive, 2018:1135, 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. arXiv preprint arXiv:1801.10228, 2018.

[8] Thirasara Ariyarathna, Prabodha Harankahadeniya, Saarrah Isthikar, Nethmi Pathirana, HMN Dilum Bandara, and Arjuna Madanayake. Dynamic spectrum access via smart contracts on blockchain. In 2019 IEEE Wireless Communications and Networking Conference (WCNC), pages 1–6. IEEE, 2019.

[9] ARM Limited. ARM security technology building a secure system using trustzone technology, 2009. [Online]. Available: https://developer.arm.com/documentation/PRD29-GENC-009492/c/TrustZone-Hardware-Architecture. [Accessed: May 8, 2022].

[10] Hagit Attiya and Jennifer Welch. Distributed computing: fundamentals, simulations, and advanced topics, volume 19. John Wiley & Sons, 2004.

[11] Hagit Attiya, Amotz Bar-Noy, and Danny Dolev. Sharing memory robustly in message-passing systems. Journal of the ACM (JACM), 42(1):124–142, 1995.

[12] Enrico Bacis, Sabrina De Capitani di Vimercati, Sara Foresti, Stefano Paraboschi, Marco Rosa, and Pierangela Samarati. Mix&slice: Efficient access revocation in the cloud. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pages 217–228. ACM, 2016.

[13] Joonsang Baek and Yuliang Zheng. Simple and efficient threshold cryptosystem from the gap diffie-hellman group. In IEEE Global Telecommunications Conference (GLOBECOM 03), volume 3, pages 1491–1495. IEEE, 2003.

[14] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. Consensus in the age of blockchains. arXiv preprint arXiv:1711.03936, 2017.

[15] Shubhekshya Basnet, Ying He, Eryk Dutkiewicz, and Beeshanga Abewardana Jayawickrama. Resource allocation in moving and fixed general authorized access users in spectrum access system. IEEE Access, 7:107863–107873, 2019.

[16] Annika Baumann, Benjamin Fabian, and Matthias Lischke. Exploring the bitcoin network. In WEBIST (1), pages 369–374, 2014.

[17] Alireza Beikverdi and JooSeok Song. Trend of centralization in bitcoin's distributed network. In 2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), pages 1–6. IEEE, 2015.

[18] Michael Ben-Or. Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols. In Proceedings of the second annual ACM symposium on Principles of distributed computing, pages 27–30. ACM, 1983.

[19] Michael Ben-Or, Boaz Kelmer, and Tal Rabin. Asynchronous secure computations with optimal resilience. In Proceedings of the thirteenth annual ACM symposium on Principles of distributed computing, pages 183–192. ACM, 1994.

[20] Burak Benligiray, Saša Milic, and Heikki Vänttinen. Decentralized APIs for Web 3.0. API3 Foundation Whitepaper, 2020.

[21] Iddo Bentov, Charles Lee, Alex Mizrahi, and Meni Rosenfeld. proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y. ACM SIGMETRICS Performance Evaluation Review, 42(3):34–37, 2014.

[22] Iddo Bentov, Ariel Gabizon, and Alex Mizrahi. Cryptocurrencies without proof of work. In International Conference on Financial Cryptography and Data Security, pages 142–157. Springer, 2016.

[23] Alysson Bessani, João Sousa, and Eduardo EP Alchieri. State machine replication for the masses with bft-smart. In Proceedings of 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pages 355–362. IEEE, 2014.

[24] Alysson Neves Bessani. (BFT) state machine replication: The hype, the virtue... and even some practice. Tutorial on EuroSys 2012, 2012. [Online]. Available: http://www.di.fc.ul.pt/~bessani/publications/T1-bftsmr.pdf. [Accessed: May 8, 2022].

[25] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In 2007 IEEE symposium on security and privacy (SP'07), pages 321–334. IEEE, 2007.

[26] Bitnodes. Bitnodes estimates the relative size of the bitcoin peer-to-peer network by finding all of its reachable nodes, 2022. [Online]. Available: https://bitnodes.io/. [Accessed: May 8, 2022].

[27] Block.one. EOS.IO technical white paper v2, 2018. [Online]. Available: https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md. [Accessed: May 8, 2022].

[28] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Theory of Cryptography Conference, pages 253–273. Springer, 2011.

[29] Mic Bowman, Andrea Miele, Michael Steiner, and Bruno Vavala. Private data objects: an overview. arXiv preprint arXiv:1807.05686 [cs.CR], 2018.

[30] Gabriel Bracha. An asynchronous [(n-1)/3]-resilient consensus protocol. In Proceedings of the third annual ACM symposium on Principles of distributed computing, pages 154–162. ACM, 1984.

[31] Gabriel Bracha. Asynchronous byzantine agreement protocols. Information and Computation, 75(2):130–143, 1987.

[32] Gabriel Bracha and Sam Toueg. Asynchronous consensus and broadcast protocols. Journal of the ACM (JACM), 32(4):824–840, 1985.

[33] brainbot technologies AG. The Raiden Network, 2022. [Online]. Available: https://raiden.network/. [Accessed: May 8, 2022].

[34] Ulrik Brandes. A faster algorithm for betweenness centrality. Journal of mathematical sociology, 25(2):163–177, 2001.

[35] Lorenz Breidenbach, Christian Cachin, Benedict Chan, Alex Coventry, Steve Ellis, Ari Juels, Farinaz Koushanfar, Andrew Miller, Brendan Magauran, Daniel Moroz, et al. Chainlink 2.0: Next steps in the evolution of decentralized oracle networks, 2021.

[36] Ethan Buchman. Tendermint: Byzantine fault tolerance in the age of blockchains. PhD thesis, University of Guelph, 2016.

[37] Vitalik Buterin. Long-range attacks: The serious problem with adaptive proof of work. ethereum foundation blog, 2014. [Online]. Available: https://blog.ethereum.org/2014/05/15/long-range-attacks-the-serious-problem-with-adaptive-proof-of-work/. [Accessed: May 8, 2022].

[38] Vitalik Buterin. Privacy on blockchain. ethereum foundation blog, 2016. [Online]. Available: https://blog.ethereum.org/2016/01/15/privacy-on-the-blockchain/. [Accessed: May 8, 2022].

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

[40] Christian Cachin and Stefano Tessaro. Asynchronous verifiable information dispersal. In 24th IEEE Symposium on Reliable Distributed Systems (SRDS'05), pages 191–201. IEEE, 2005.

[41] Christian Cachin and Marko Vukolić. Blockchain consensus protocols in the wild. In 31 International Symposium on Distributed Computing (DISC). Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[42] Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup. Secure and efficient asynchronous broadcast protocols. In Annual International Cryptology Conference, pages 524–541. Springer, 2001.

[43] Christian Cachin, Klaus Kursawe, and Victor Shoup. Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography. Journal of Cryptology, 18(3):219–246, 2005.

[44] Yuxi Cai, Nafis Irtija, Eirini Eleni Tsiropoulou, and Andreas Veneris. Truthful decentralized blockchain oracles. International Journal of Network Management, page e2179, 2021.

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

[46] Ran Canetti and Tal Rabin. Fast asynchronous byzantine agreement with optimal resilience. In Proceedings of the twenty-fifth annual ACM symposium on Theory of computing, pages 42–51. ACM, 1993.

[47] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. Journal of the ACM (JACM), 51(4):557–594, 2004.

[48] Ning Cao, Cong Wang, Ming Li, Kui Ren, and Wenjing Lou. Privacy-preserving multi-keyword ranked search over encrypted cloud data. IEEE Transactions on parallel and distributed systems, 25(1):222–233, 2014.

[49] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI 99), volume 99, pages 173–186. USENIX, 1999.

[50] Tushar Deepak Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. Journal of the ACM (JACM), 43(2):225–267, 1996.

[51] Raymond Cheng, Fan Zhang, Jernej Kos, Warren He, Nicholas Hynes, Noah Johnson, Ari Juels, Andrew Miller, and Dawn Song. Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts. In 2019 IEEE European Symposium on Security and Privacy (EuroS&P), pages 185–200. IEEE, 2019.

[52] Raymond Cheng, Fan Zhang, Jernej Kos, Warren He, Nicholas Hynes, Noah Johnson, Ari Juels, Andrew Miller, and Dawn Song. Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts. In 2019 IEEE European Symposium on Security and Privacy (EuroS&P), pages 185–200. IEEE, 2019.

[53] Arka Rai Choudhuri, Matthew Green, Abhishek Jain, Gabriel Kaptchuk, and Ian Miers. Fairness in an unfair world: Fair multiparty computation from public bulletin boards. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pages 719–728. ACM, 2017.

[54] Anton Churyumov. Byteball: A decentralized system for storage and transfer of value, 2016. [Online]. Available: https://obyte.org/Byteball.pdf. [Accessed: May 8, 2022].

[55] Allen Clement, Edmund L Wong, Lorenzo Alvisi, Michael Dahlin, and Mirco Marchetti.

Making byzantine fault tolerant systems tolerate byzantine faults. In Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, volume 9, pages 153–168. USENIX, 2009.

[56] Nxt community. Nxt whitepaper revision 4, 2014. [Online]. Available: https://www.jelurida.com/sites/default/files/NxtWhitepaper.pdf. [Accessed: May 8, 2022].

[57] Victor Costan and Srinivas Devadas. Intel sgx explained. IACR Cryptology ePrint Archive, 2016:86, 2016.

[58] Victor Costan, Ilia Lebedev, and Srinivas Devadas. Sanctum: Minimal hardware extensions for strong software isolation. In 25th USENIX Security Symposium (USENIX Security 16), pages 857–874. USENIX, 2016.

[59] James Cowling, Daniel Myers, Barbara Liskov, Rodrigo Rodrigues, and Liuba Shrira. Hq replication: A hybrid quorum protocol for byzantine fault tolerance. In Proceedings of the 7th symposium on Operating systems design and implementation, pages 177–190. USENIX, 2006.

[60] Flaviu Cristian, Houtan Aghili, Raymond Strong, and Danny Dolev. Atomic broadcast: From simple message diffusion to Byzantine agreement. Citeseer, 1986.

[61] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, et al. On scaling decentralized blockchains. In International Conference on Financial Cryptography and Data Security, pages 106–125. Springer, 2016.

[62] Bart Custers and Helena Uršič. Big data and data reuse: a taxonomy of data reuse for balancing big data benefits and personal data protection. International Data Privacy Law, 6(1):4–15, 2016.

[63] Phil Daian, Rafael Pass, and Elaine Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake. In International Conference on Financial Cryptography and Data Security, pages 23–41. Springer, 2019.

[64] Anupam Datta, Matthew Fredrikson, Gihyuk Ko, Piotr Mardziel, and Shayak Sen. Use privacy in data-driven systems: Theory and experiments with machine learnt programs. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pages 1193–1210. ACM, 2017.

[65] 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.

[66] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. In Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on, pages 1–10. IEEE, 2013.

[67] Christian Decker, Jochen Seidel, and Roger Wattenhofer. Bitcoin meets strong consistency. In Proceedings of the 17th International Conference on Distributed Computing and Networking, page 13. ACM, 2016.

[68] Xavier Défago, André Schiper, and Péter Urbán. Total order broadcast and multicast algorithms: Taxonomy and survey. ACM Computing Surveys (CSUR), 36(4):372–421, 2004.

[69] Amir Dembo, Sreeram Kannan, Ertem Nusret Tas, David Tse, Pramod Viswanath, Xuechao Wang, and Ofer Zeitouni. Everything is a race and nakamoto always wins. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, pages 859–878. ACM, 2020.

[70] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017. [Online]. Available: http://archive.ics.uci.edu/ml. [Accessed: May 8, 2022].

[71] Digiconomist. Bitcoin energy consumption index, 2022. [Online]. Available: https://digiconomist.net/bitcoin-energy-consumption. [Accessed: May 8, 2022].

[72] Sisi Duan, Michael K Reiter, and Haibin Zhang. Beat: Asynchronous bft made practical. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pages 2028–2041. ACM, 2018.

[73] Cynthia Dwork. Differential privacy: A survey of results. In International conference on theory and applications of models of computation, pages 1–19. Springer, 2008.

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

[75] Eslam Elnikety, Aastha Mehta, Anjo Vahldiek-Oberwagner, Deepak Garg, and Peter Druschel. Thoth: Comprehensive policy compliance in data retrieval systems. In 25th USENIX Security Symposium (USENIX Security 16), pages 637–654. USENIX, 2016.

[76] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In Proceedings of the 2014 ACM SIGSAC conference on computer and communications security, pages 1054–1067. ACM, 2014.

[77] Ethereum Foundation. Ethereum wire protocol (ETH), 2021. [Online]. Available: https://github.com/ethereum/devp2p/blob/master/caps/eth.md. [Accessed: May 8, 2022].

[78] Ethereum Wiki. Casper proof of stake compendium, 2020. [Online]. Available: https://eth.wiki/en/concepts/casper-proof-of-stake-compendium. [Accessed: May 8, 2022].

[79] Ethereum Wiki. Sharding roadmap, 2020. [Online]. Available: https://eth.wiki/sharding/sharding-roadmap. [Accessed: May 8, 2022].

[80] Ethereum Wiki. Grinding attack on proof of stake, 2022. [Online]. Available: https://eth.wiki/en/concepts/proof-of-stake-faqs. [Accessed: May 8, 2022].

[81] European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). Official Journal of the European Union, 2016. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj. [Accessed: May 8, 2022].

[82] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In International conference on financial cryptography and data security, pages 436–454. Springer, 2014.

[83] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. {Bitcoin-NG}: A scalable blockchain protocol. In 13th USENIX symposium on networked systems design and implementation (NSDI 16), pages 45–59. USENIX, 2016.

[84] Ben Fisch, Dhinakaran Vinayagamurthy, Dan Boneh, and Sergey Gorbunov. Iron: functional encryption using intel sgx. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pages 765–782. ACM, 2017.

[85] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. Journal of the ACM (JACM), 32(2): 374–382, 1985.

[86] BitShares Blockchain Foundation. Bitshares documentation, 2019. [Online]. Available: https://how.bitshares.works/en/master/. [Accessed: May 8, 2022].

[87] Hyperledger Foundation. Hyperledger Sawtooth, 2021. [Online]. Available: https://wiki.hyperledger.org/display/sawtooth. [Accessed: May 8, 2022].

[88] Matt Franklin. A survey of key evolving cryptosystems. International Journal of Security and Networks, 1(1-2):46–53, 2006.

[89] Luoyi Fu, Jiasheng Xu, Shan Qu, Zhiying Xu, Xinbing Wang, and Guihai Chen. Seeking the truth in a decentralized manner. IEEE/ACM Transactions on Networking, 2021.

[90] 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.

[91] Peter Gaži, Aggelos Kiayias, and Alexander Russell. Stake-bleeding attacks on proof-of-stake blockchains. In 2018 Crypto Valley Conference on Blockchain Technology (CVCBT), pages 85–92. IEEE, 2018.

[92] Peter Gaži, Aggelos Kiayias, and Alexander Russell. Tight consistency bounds for bitcoin. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, pages 819–838, 2020.

[93] Adem Efe Gencer, Soumya Basu, Ittay Eyal, Robbert van Renesse, and Emin Gün Sirer. Decentralization in bitcoin and ethereum networks. In International Conference on Financial Cryptography and Data Security, pages 439–457. Springer, 2018.

[94] Arthur Gervais, Ghassan O Karame, Vedran Capkun, and Srdjan Capkun. Is bitcoin a decentralized currency? IEEE security & privacy, 12(3):54–60, 2014.

[95] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pages 3–16. ACM, 2016.

[96] 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. ACM, 2017.

[97] Google. Changes in Google SAS response, 2020. [Online]. Available: https://support.google.com/sas/answer/9981557. [Accessed: May 8, 2022].

[98] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Proceedings of the 13th ACM Conference on Computer and Communications Security, pages 89–98. ACM, 2006.

[99] Mohamed Grissa, Attila A Yavuz, and Bechir Hamdaoui. TrustSAS: A trustworthy spectrum access system for the 3.5 GHz CBRS band. In IEEE INFOCOM 2019-IEEE Conference on Computer Communications, pages 1495–1503. IEEE, 2019.

[100] Juan Guarnizo and Pawel Szalachowski. Pdfs: practical data feed service for smart contracts. In European Symposium on Research in Computer Security, pages 767–789. Springer, 2019.

[101] Rachid Guerraoui, Nikola Knežević, Vivien Quéma, and Marko Vukolić. The next 700 bft protocols. In Proceedings of the 5th European conference on Computer systems, pages 363–376. ACM, 2010.

[102] Dave Gutteridge. Japanese cryptocurrency monacoin hit by selfish mining attack, 2018. [Online]. Available: https://www.ccn.com/

japanese-cryptocurrency-monacoin-hit-by-selfish-mining-attack/. [Accessed: May 8, 2022].

[103] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[104] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. In Proceedings of the 24th USENIX Security Symposium, pages 129–144. USENIX, 2015.

[105] Tyler Hunt, Congzheng Song, Reza Shokri, Vitaly Shmatikov, and Emmett Witchel. Chiron: Privacy-preserving machine learning as a service. arXiv preprint arXiv:1803.05961 [cs.CR], 2018.

[106] Tyler Hunt, Zhiting Zhu, Yuanzhong Xu, Simon Peter, and Emmett Witchel. Ryoan: A distributed sandbox for untrusted computation on secret data. ACM Transactions on Computer Systems (TOCS), 35(4):1–32, 2018.

[107] Hyperledger Foundation. Hyperledger caliper, 2022. [Online]. Available: https://hyperledger.github.io/caliper/. [Accessed: May 8, 2022].

[108] Naru Jai, Shaoran Li, Chengzhang Li, Y Thomas Hou, Wenjing Lou, Jeffrey H Reed, and Sastry Kompella. Optimal channel allocation in the cbrs band with shipborne radar incumbents. In 2021 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN), pages 80–88. IEEE, 2021.

[109] Harry Kalodner, Steven Goldfeder, Xiaoqi Chen, S Matthew Weinberg, and Edward W Felten. Arbitrum: Scalable, private smart contracts. In Proceedings of the 27th USENIX Conference on Security Symposium, pages 1353–1370. USENIX, 2018.

[110] 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.

[111] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. self-published paper, August, 19, 2012.

[112] Eleftherios Kokoris Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. In 25th USENIX Security Symposium (USENIX Security 16), pages 279–296. USENIX, 2016.

[113] Dániel Kondor, Márton Pósfai, István Csabai, and Gábor Vattay. Do the rich get richer? an empirical analysis of the bitcoin transaction network. PloS one, 9(2): e86197, 2014.

[114] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In Security and Privacy (SP), 2016 IEEE Symposium on, pages 839–858. IEEE, 2016.

[115] Ramakrishna Kotla, Lorenzo Alvisi, Mike Dahlin, Allen Clement, and Edmund Wong. Zyzzyva: speculative byzantine fault tolerance. ACM SIGOPS Operating Systems Review, 41(6):45–58, 2007.

[116] Michael Kratsios. Emerging technologies and their expected impact on non-federal spectrum demand. Executive Office of the President of the United States, 2019.

[117] Jae Kwon. Tendermint: Consensus without mining, 2014. [Online]. Available: https://tendermint.com/static/docs/tendermint.pdf. [Accessed: May 8, 2022].

[118] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. Communications of the ACM, 21(7):558–565, 1978.

[119] Leslie Lamport. Using time instead of timeout for fault-tolerant distributed systems. ACM Transactions on Programming Languages and Systems (TOPLAS), 6(2):254–280, 1984.

[120] Leslie Lamport. The part-time parliament. ACM Transactions on Computer Systems (TOCS), 16(2):133–169, 1998.

[121] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. ACM Transactions on Programming Languages and Systems (TOPLAS), 4(3):382–401, 1982.

[122] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. In Concurrency: the Works of Leslie Lamport, pages 203–226. ACM, 2019.

[123] Leslie Lamport et al. Paxos made simple. ACM Sigact News, 32(4):18–25, 2001.

[124] Daniel Larimer. DPOS BFT—pipelined byzantine fault tolerance. Medium, 2018. [Online]. Available: https://medium.com/eosio/dpos-bft-pipelined-byzantine-fault-tolerance-8a0634a270ba. [Accessed: May 8, 2022].

[125] Colin LeMahieu. Nano: A feeless distributed cryptocurrency network, 2018. [Online]. Available: https://media.abnnewswire.net/media/en/docs/91948-whitepaper.pdf. [Accessed: May 8, 2022].

[126] Chenxing Li, Peilun Li, Wei Xu, Fan Long, and Andrew Chi-chih Yao. Scaling nakamoto consensus to thousands of transactions per second. arXiv preprint arXiv:1805.03870, 2018.

[127] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In 2007 IEEE 23rd International Conference on Data Engineering, pages 106–115. IEEE, 2007.

[128] Xian Li, Xin Luna Dong, Kenneth Lyons, Weiyi Meng, and Divesh Srivastava. Truth finding on the deep web: Is the problem solved? arXiv preprint arXiv:1503.00303, 2015.

[129] Yaliang Li, Qi Li, Jing Gao, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. On the discovery of evolving truth. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 675–684. ACM, 2015.

[130] Yaliang Li, Jing Gao, Chuishi Meng, Qi Li, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. A survey on truth discovery. ACM Sigkdd Explorations Newsletter, 17(2):1–16, 2016.

[131] Yaliang Li, Chenglin Miao, Lu Su, Jing Gao, Qi Li, Bolin Ding, Zhan Qin, and Kui Ren. An efficient two-layer mechanism for privacy-preserving truth discovery. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 1705–1714. ACM, 2018.

[132] Lisk Foundation. Lisk SDK - consensus algorithm, 2022. [Online]. Available: https://lisk.com/documentation/lisk-sdk/protocol/consensus-algorithm.html. [Accessed: May 8, 2022].

[133] Barbara Liskov and James Cowling. Viewstamped replication revisited. CSAIL Technical Reports (July 1, 2003 - present), 2012.

[134] Litecoin Foundation. Litecoin - open source P2P digital currency, 2022. [Online]. Available: https://litecoin.org/. [Accessed: May 8, 2022].

[135] Jieyi Long and Ribao Wei. Scalable bft consensus mechanism through aggregated signature gossip. In 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), pages 360–367. IEEE, 2019.

[136] Loi Luu, Viswesh Narayanan, Kunal Baweja, Chaodong Zheng, Seth Gilbert, and Prateek Saxena. Scp: A computationally-scalable byzantine consensus protocol for blockchains. IACR Cryptology ePrint Archive, 2015:1168, 2015.

[137] Fenglong Ma, Yaliang Li, Qi Li, Minghui Qiu, Jing Gao, Shi Zhi, Lu Su, Bo Zhao, Heng Ji, and Jiawei Han. Faitcrowd: Fine grained truth discovery for crowdsourced data aggregation. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 745–754. ACM, 2015.

[138] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. ACM Transactions on Knowledge Discovery from Data (TKDD), 1(1):3–es, 2007.

[139] KB Shashika Manosha, S Joshi, Tuomo Hänninen, Markku Jokinen, Pekka Pirinen, Harri Posti, Kari Horneman, Seppo Yrjölä, and Matti Latva-aho. A channel allocation algorithm for citizens broadband radio service/spectrum access system. In 2017 European Conference on Networks and Communications (EuCNC), pages 1–6. IEEE, 2017.

[140] J-P Martin and Lorenzo Alvisi. Fast byzantine consensus. IEEE Transactions on Dependable and Secure Computing, 3(3):202–215, 2006.

[141] Julian Martinez. Understanding proof of stake: The nothing at stake theory. Medium, 2018. [Online]. Available: https://medium.com/coinmonks/understanding-proof-of-stake-the-nothing-at-stake-theory-1f0d71bc027. [Accessed: May 8, 2022].

[142] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R Savagaonkar. Innovative instructions and software model for isolated execution. In HASP@ ISCA, page 10, 2013.

[143] Lucas Mearian. FCC eyes blockchain to track, monitor growing wireless spectrums. https://www.computerworld.com/article/3393179/fcc-eyes-blockchain-to-track-monitor-growing-wireless-spectrums.html, 2019.

[144] Sam Meredith. Facebook-cambridge analytica: A timeline of the data hijacking scandal. CNBC, 2018. [Online]. Available: https://www.cnbc.com/2018/04/10/facebook-cambridge-analytica-a-timeline-of-the-data-hijacking-scandal.html. [Accessed: May 8, 2022].

[145] 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.

[146] Silvio Micali, Michael Rabin, and Salil Vadhan. Verifiable random functions. In 40th annual symposium on foundations of computer science (cat. No. 99CB37039), pages 120–130. IEEE, 1999.

[147] Andrew Miller, James Litton, Andrew Pachulski, Neal Gupta, Dave Levin, Neil Spring, and Bobby Bhattacharjee. Discovering bitcoin's public topology and influential nodes, 2015. [Online]. Available: https://allquantor.at/blockchainbib/pdf/miller2015topology.pdf. [Accessed: May 8, 2022].

[148] 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. ACM, 2016.

[149] Achour Mostefaoui, Hamouma Moumen, and Michel Raynal. Signature-free asynchronous byzantine consensus with t< n/3 and o (n2) messages. In Proceedings of the 2014 ACM symposium on Principles of distributed computing, pages 2–9. ACM, 2014.

[150] Finn Lützow-Holm Myrstad. TED Talk: How tech companies deceive you into giving up your data and privacy, 2018. [Online]. Available: https://goo.gl/hSfaUX. [Accessed: May 8, 2022].

[151] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf. [Accessed: May 8, 2022].

[152] National Science and Technology Council. National privacy research strategy, 2016. [Online]. Available: https://www.nitrd.gov/PUBS/NationalPrivacyResearchStrategy.pdf. [Accessed: May 8, 2022].

[153] Christopher Natoli, Jiangshan Yu, Vincent Gramoli, and Paulo Esteves-Verissimo. Deconstructing blockchains: A comprehensive survey on consensus, membership and structure. arXiv preprint arXiv:1908.08316, 2019.

[154] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In 2016 IEEE European Symposium on Security and Privacy (EuroS&P), pages 305–320. IEEE, 2016.

[155] NCC Group. Reference implementation of a verifiable random function (VRF) from IETF draft-irtf-cfrg-vrf-06 specification, 2020. [Online]. Available: https://github.com/nccgroup/draft-irtf-cfrg-vrf-06. [Accessed: May 8, 2022].

[156] Bitcoin Developer Network. Bitcoin wire protocol 101, 2019. [Online]. Available: https://bitcoindev.network/bitcoin-wire-protocol/. [Accessed: May 8, 2022].

[157] Till Neudecker, Philipp Andelfinger, and Hannes Hartenstein. Timing analysis for inferring the topology of the bitcoin peer-to-peer network. In 2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), pages 358–367. IEEE, 2016.

[158] Helen Nissenbaum. Privacy as contextual integrity. Washington Law Review, 79:119, 2004.

[159] Shen Noether. Ring signature confidential transactions for monero. IACR Cryptology ePrint Archive, 2015:1098, 2015.

[160] Olga Ohrimenko, Felix Schuster, Cédric Fournet, Aastha Mehta, Sebastian Nowozin, Kapil Vaswani, and Manuel Costa. Oblivious {Multi-Party} machine learning on trusted processors. In 25th USENIX Security Symposium (USENIX Security 16), pages 619–636. USENIX, 2016.

[161] Brian M Oki and Barbara H Liskov. Viewstamped replication: A new primary copy method to support highly-available distributed systems. In Proceedings of the seventh annual ACM Symposium on Principles of distributed computing, pages 8–17. ACM, 1988.

[162] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In 2014 USENIX Annual Technical Conference (Usenix ATC 14), pages 305–319. USENIX, 2014.

[163] Robin Wentao Ouyang, Lance M Kaplan, Alice Toniolo, Mani Srivastava, and Timothy J Norman. Parallel and streaming truth discovery in large-scale quantitative

crowdsourcing. IEEE Transactions on Parallel and Distributed Systems, 27(10):2984–2997, 2016.

[164] Rafael Pass and Elaine Shi. Hybrid consensus: Efficient consensus in the permissionless model. IACR Cryptology ePrint Archive, 2016:917, 2016.

[165] Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In Proceedings of the ACM Symposium on Principles of Distributed Computing, pages 315–324. ACM, 2017.

[166] Rafael Pass and Elaine Shi. The sleepy model of consensus. In International Conference on the Theory and Application of Cryptology and Information Security, pages 380–409. Springer, 2017.

[167] Rafael Pass, Elaine Shi, and Florian Tramer. Formal abstractions for attested execution secure processors. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 260–289. Springer, 2017.

[168] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. Journal of the ACM (JACM), 27(2):228–234, 1980.

[169] POA Network. POA Network whitepaper, 2018. [Online]. Available: https://github.com/poanetwork/wiki/wiki/POA-Network-Whitepaper. [Accessed: May 8, 2022].

[170] Andrew Poelstra. Distributed consensus from proof of stake is impossible, 2014. [Online]. Available: https://download.wpsoftware.net/bitcoin/old-pos.pdf. [Accessed: May 8, 2022].

[171] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments, 2016. [Online]. Available: https://www.bitcoinlightning.com/wp-content/uploads/2018/03/lightning-network-paper.pdf. [Accessed: May 8, 2022].

[172] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments, 2016.

[173] Serguei Popov. The tangle, 2018. [Online]. Available: http://www.descryptions.com/Iota.pdf. [Accessed: May 8, 2022].

[174] Band Protocol. Bandchain whitepaper, 2022. [Online]. Available: https://docs.bandchain.org/whitepaper/. [Accessed: May 8, 2022].

[175] Michael O Rabin. Randomized byzantine generals. In 24th Annual Symposium on Foundations of Computer Science (sfcs 1983), pages 403–409. IEEE, 1983.

[176] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In Proceedings of the twenty-first annual ACM symposium on Theory of computing, pages 73–85. ACM, 1989.

[177] Michael K Reiter. A secure group membership protocol. IEEE Transactions on Software Engineering, 22(1):31–42, 1996.

[178] Ling Ren. Analysis of nakamoto consensus. IACR Cryptology ePrint Archive, 2019: 943, 2019.

[179] Team Rocket. Snowflake to avalanche: A novel metastable consensus protocol family for cryptocurrencies, 2018. [Online]. Available: https://ipfs.io/ipfs/QmUy4jh5mGNZvLkjies1RWM4YuvJh5o2FYopNPVYwrRVGV. [Accessed: May 8, 2022].

[180] Danny Ryan and Chih-Cheng Liang. EIP 1011: Hybrid Casper FFG. Ethereum Improvement Proposals, 2018. [Online]. Available: https://eips.ethereum.org/EIPS/eip-1011. [Accessed: May 8, 2022].

[181] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. Trusted execution environment: what it is, and what it is not. In 2015 IEEE Trustcom/BigDataSE/ISPA, volume 1, pages 57–64. IEEE, 2015.

[182] Sara Salinas and Sam Meredith. Tim cook: Personal data collection is being 'weaponized against us with military efficiency'. CNBC, 2018. [Online]. Available: https://www.cnbc.com/2018/10/24/apples-tim-cook-warns-silicon-valley-it-would-be-destructive-to-block-strong-privacy-laws.html. [Accessed: May 8, 2022].

[183] 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.

[184] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In 2014 IEEE Symposium on Security and Privacy, pages 459–474. IEEE, 2014.

[185] Fred B Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. ACM Computing Surveys (CSUR), 22(4):299–319, 1990.

[186] Felix Schuster, Manuel Costa, Cédric Fournet, Christos Gkantsidis, Marcus Peinado, Gloria Mainar-Ruiz, and Mark Russinovich. Vc3: Trustworthy data analytics in the cloud using sgx. In Security and Privacy (SP), 2015 IEEE Symposium on, pages 38–54. IEEE, 2015.

[187] David Schwartz, Noah Youngs, and Arthur Britto. The ripple protocol consensus algorithm, 2014. [Online]. Available: https://www.allcryptowhitepapers.com/ripple-whitepaper/. [Accessed: May 8, 2022].

[188] Shayak Sen, Saikat Guha, Anupam Datta, Sriram K Rajamani, Janice Tsai, and Jeannette M Wing. Bootstrapping privacy compliance in big data systems. In Security and Privacy (SP), 2014 IEEE Symposium on, pages 327–342. IEEE, 2014.

[189] Shanghao Shi, Yang Xiao, Wenjing Lou, Chonggang Wang, Xu Li, Y. Thomas Hou, and Jeffrey H. Reed. Challenges and new directions in securing spectrum access systems. IEEE Internet of Things Journal, 8(8):6498–6518, 2021.

[190] Victor Shoup and Rosario Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In International Conference on the Theory and Applications of Cryptographic Techniques, pages 1–16. Springer, 1998.

[191] Rohit Sinha, Sivanarayana Gaddam, and Ranjit Kumaresan. Luciditee: Policy-compliant fair computing at scale. IACR Cryptology ePrint Archive, 2019:178, 2019.

[192] Yonatan Sompolinsky and Aviv Zohar. Accelerating bitcoin's transaction processing. fast money grows on trees, not chains. IACR Cryptology ePrint Archive, 2013:881, 2013.

[193] 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.

[194] Yonatan Sompolinsky and Aviv Zohar. Phantom: A scalable BlockDAG protocol. IACR Cryptology ePrint Archive, 2018:104, 2018.

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

[196] Dokyung Song, Julian Lettner, Prabhu Rajasekaran, Yeoul Na, Stijn Volckaert, Per

Larsen, and Michael Franz. Sok: sanitizing for security. In 2019 IEEE Symposium on Security and Privacy (SP), pages 1275–1295. IEEE, 2019.

[197] Joao Sousa, Alysson Bessani, and Marko Vukolic. A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform. In 2018 48th annual IEEE/IFIP international conference on dependable systems and networks (DSN), pages 51–58. IEEE, 2018.

[198] Balaji S. Srinivasan and Leland Lee. Quantifying decentralization, 2017. [Online]. Available: https://news.earn.com/quantifying-decentralization-e39db233c28e. [Accessed: May 8, 2022].

[199] Steemit Inc. Steem—an incentivized, blockchain-based, public content platform., 2018. [Online]. Available: https://steem.com/steem-whitepaper.pdf. [Accessed: May 8, 2022].

[200] Peng Sun, Zhibo Wang, Yunhe Feng, Liantao Wu, Yanjun Li, Hairong Qi, and Zhi Wang. Towards personalized privacy-preserving incentive for truth discovery in crowdsourced binary-choice question answering. In Proceedings of the 2020 IEEE International Conference on Computer Communications. IEEE, 2020.

[201] Latanya Sweeney. k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 10(05):557–570, 2002.

[202] Ewa Syta, Iulia Tamas, Dylan Visher, David Isaac Wolinsky, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ismail Khoffi, and Bryan Ford. Keeping authorities" honest or bust" with decentralized witness cosigning. In 2016 IEEE Symposium on Security and Privacy (SP), pages 526–545. IEEE, 2016.

[203] Nick Szabo. Formalizing and securing relationships on public networks. First Monday, 2(9), 1997.

[204] Tendermint Inc. Cosmos hub - validators overview, 2022. [Online]. Available: https://hub.cosmos.network/main/validators/overview.html. [Accessed: May 8, 2022].

[205] Tendermint Inc. Tendermint core - peer discovery, 2022. [Online]. Available: https://docs.tendermint.com/master/spec/p2p/node.html. [Accessed: May 8, 2022].

[206] The Office of the Federal Register (OFR) and the Government Publishing Office. OFR: Electronic Code of Federal Regulations, Title 47: Telecommunication, Part 96 - Citizens Broadband Radio Service, 2015. [Online]. Available: https://www.ecfr.gov/current/title-47/chapter-I/subchapter-D/part-96. [Accessed: May 8, 2022].

[207] Yifan Tian, Jiawei Yuan, and Houbing Song. Secure and reliable decentralized truth discovery using blockchain. In 2019 IEEE Conference on Communications and Network Security (CNS), pages 1–8. IEEE, 2019.

[208] Jo Van Bulck, Nico Weichbrodt, Rüdiger Kapitza, Frank Piessens, and Raoul Strackx. Telling your secrets without page faults: Stealthy page table-based attacks on enclaved execution. In 26th USENIX Security Symposium (USENIX Security 17), pages 1041–1056. USENIX, 2017.

[209] Jo Van Bulck, Frank Piessens, and Raoul Strackx. Foreshadow: Extracting the keys to the intel {SGX} kingdom with transient out-of-order execution. In 27th USENIX Security Symposium (USENIX Security 18). USENIX, 2018.

[210] Giuliana Santos Veronese, Miguel Correia, Alysson Neves Bessani, and Lau Cheuk Lung. Spin one's wheels? byzantine fault tolerance with a spinning primary. In Pro-

ceedings of the 28th IEEE International Symposium on Reliable Distributed Systems, pages 135–144. IEEE, 2009.

[211] Vassilios S Verykios, Elisa Bertino, Igor Nai Fovino, Loredana Parasiliti Provenza, Yucel Saygin, and Yannis Theodoridis. State-of-the-art in privacy preserving data mining. ACM Sigmod Record, 33(1):50–57, 2004.

[212] Sabrina De Capitani Di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati. Encryption policies for regulating access to outsourced data. ACM Transactions on Database Systems (TODS), 35(2):12, 2010.

[213] Visa Inc. VisaNet: The technology behind Visa, 2013. [Online]. Available: https://usa.visa.com/dam/VCOM/download/corporate/media/visanet-technology/visa-net-booklet.pdf. [Accessed: May 8, 2022].

[214] Marko Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. In International Workshop on Open Problems in Network Security, pages 112–125. Springer, 2015.

[215] Dong Wang, Tarek Abdelzaher, Lance Kaplan, and Charu C Aggarwal. Recursive fact-finding: A streaming approach to truth estimation in crowdsourcing applications. In 2013 IEEE 33rd International Conference on Distributed Computing Systems, pages 530–539. IEEE, 2013.

[216] Guojun Wang, Qin Liu, and Jie Wu. Hierarchical attribute-based encryption for fine-grained access control in cloud storage services. In Proceedings of the 17th ACM conference on Computer and communications security, pages 735–737. ACM, 2010.

[217] 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.

[218] Wenhao Wang, Guoxing Chen, Xiaorui Pan, Yinqian Zhang, XiaoFeng Wang, Vincent Bindschaedler, Haixu Tang, and Carl A Gunter. Leaky cauldron on the dark land: Understanding memory side-channel hazards in sgx. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pages 2421–2434. ACM, 2017.

[219] Yaqing Wang, Fenglong Ma, Lu Su, and Jing Gao. Discovering truths from distributed data. In 2017 IEEE International Conference on Data Mining (ICDM), pages 505–514. IEEE, 2017.

[220] Wave Team. Wave-NG protocol, 2022. [Online]. Available: https://docs.waves.tech/en/blockchain/waves-protocol/waves-ng-protocol. [Accessed: May 8, 2022].

[221] Martin BH Weiss and William Lehr. Market based approaches for dynamic spectrum assignment, 2009. [Online]. Available at SSRN: https://ssrn.com/abstract=2027059. [Accessed: May 8, 2022].

[222] Martin BH Weiss, Kevin Werbach, Douglas C Sicker, and Carlos E Caicedo Bastidas. On the application of blockchains to spectrum management. IEEE Transactions on Cognitive Communications and Networking, 5(2):193–205, 2019.

[223] Wireless Innovation Forum. Spectrum Sharing Committee Policy and Procedure Coordinated Periodic Activities Policy Version, 2018. [Online]. Available: https://winnf.memberclicks.net/assets/CBRS/WINNF-SSC-0008.pdf. [Accessed: May 9, 2022].

[224] Wireless Innovation Forum. CBRS Communications Security Technical Specifi-

cation, 2020. [Online]. Available: https://winnf.memberclicks.net/assets/CBRS/WINNF-TS-0065.pdf. [Accessed: May 9, 2022].

[225] Wireless Innovation Forum. Signaling Protocols and Procedures for Citizens Broadband Radio Service (CBRS): Extensions to Spectrum Access System (SAS) - SAS Interface Technical Specification (Release 2), 2021. [Online]. Available: https://winnf.memberclicks.net/assets/CBRS/WINNF-TS-3003.pdf. [Accessed: May 9, 2022].

[226] Wireless Innovation Forum. Signaling Protocols and Procedures for Citizens Broadband Radio Service (CBRS): Spectrum Access System (SAS) - Citizens Broadband Radio Service Device (CBSD) Interface Technical Specification, 2022. [Online]. Available: https://winnf.memberclicks.net/assets/CBRS/WINNF-TS-0016.pdf. [Accessed: May 9, 2022].

[227] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. Ethereum Yellow Paper, 2022. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf. [Accessed: May 9, 2022].

[228] Karl Wüst and Arthur Gervais. Do you need a blockchain? In 2018 Crypto Valley Conference on Blockchain Technology (CVCBT), pages 45–54. IEEE, 2018.

[229] Karl Wüst, Sinisa Matetic, Silvan Egli, Kari Kostiainen, and Srdjan Capkun. ACE: asynchronous and concurrent execution of complex smart contracts. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, pages 587–600. ACM, 2020.

[230] Yang Xiao, Ning Zhang, Jin Li, Wenjing Lou, and Y. Thomas Hou. Distributed consensus protocols and algorithms. book chapter in Blockchain for Distributed Systems Security, Wiley & Sons, 2019.

[231] Yang Xiao, Ning Zhang, Jin Li, Wenjing Lou, and Y Thomas Hou. Privacyguard: Enforcing private data usage control with blockchain and attested off-chain contract execution. In European Symposium on Research in Computer Security (ESORICS), pages 610–629. Springer, Cham, 2020.

[232] Yang Xiao, Ning Zhang, Wenjing Lou, and Y Thomas Hou. Modeling the impact of network connectivity on consensus security of proof-of-work blockchain. In IEEE INFOCOM 2020-IEEE Conference on Computer Communications, pages 1648–1657. IEEE, 2020.

[233] Yang Xiao, Ning Zhang, Wenjing Lou, and Y Thomas Hou. A survey of distributed consensus protocols for blockchain networks. IEEE Communications Surveys & Tutorials, 22(2):1432–1465, 2020.

[234] Yang Xiao, Shanghao Shi, Wenjing Lou, Chonggang Wang, Xu Li, Ning Zhang, Y. Thomas Hou, and Jeffrey H. Reed. Decentralized spectrum access system: Vision, challenges, and a blockchain solution. IEEE Wireless Communications, 29(1): 220–228, 2022. doi: 10.1109/MWC.101.2100354.

[235] Yuanzhong Xu, Weidong Cui, and Marcus Peinado. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In 2015 IEEE Symposium on Security and Privacy, pages 640–656. IEEE, 2015.

[236] Jian Yin, Jean-Philippe Martin, Arun Venkataramani, Lorenzo Alvisi, and Mike Dahlin. Separating agreement from execution for byzantine fault tolerant services. In Proceedings of the nineteenth ACM symposium on Operating systems principles, pages 253–267. ACM, 2003.

[237] 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. ACM, 2019.

[238] Xiaoxin Yin, Jiawei Han, and S Yu Philip. Truth discovery with multiple conflicting information providers on the web. IEEE Transactions on Knowledge and Data Engineering, 20(6):796–808, 2008.

[239] Xuhang Ying, Milind M Buddhikot, and Sumit Roy. Sas-assisted coexistence-aware dynamic channel assignment in cbrs band. IEEE Transactions on Wireless Communications, 17(9):6307–6320, 2018.

[240] Seppo Yrjölä. Analysis of blockchain use cases in the citizens broadband radio service spectrum sharing concept. In International Conference on Cognitive Radio Oriented Wireless Networks, pages 128–139. Springer, 2017.

[241] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In INFOCOM, 2010 proceedings IEEE, pages 1–9. IEEE, 2010.

[242] Vlad Zamfir. Introducing Casper "the Friendly Ghost". ethereum foundation blog, 2015. [Online]. Available: https://blog.ethereum.org/2015/08/01/introducing-casper-friendly-ghost/. [Accessed: May 9, 2022].

[243] Daniel Zhang, Dong Wang, Nathan Vance, Yang Zhang, and Steven Mike. On scalable and robust truth discovery in big data social media sensing applications. IEEE Transactions on Big Data, 5(2):195–208, 2018.

[244] Fan Zhang, Ethan Cecchetti, Kyle Croman, Ari Juels, and Elaine Shi. Town crier: An authenticated data feed for smart contracts. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pages 270–282. ACM, 2016.

[245] Fan Zhang, Deepak Maram, Harjasleen Malvai, Steven Goldfeder, and Ari Juels. Deco: Liberating web data using decentralized oracles for tls. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, pages 1919–1938. ACM, 2020.

[246] Hanwen Zhang, Supeng Leng, and Haoye Chai. A blockchain enhanced dynamic spectrum sharing model based on proof-of-strategy. In ICC 2020-2020 IEEE International Conference on Communications (ICC), pages 1–6. IEEE, 2020.

[247] Ning Zhang, He Sun, Kun Sun, Wenjing Lou, and Y Thomas Hou. Cachekit: Evading memory introspection using cache incoherence. In 2016 IEEE European Symposium on Security and Privacy (EuroS&P), pages 337–352. IEEE, 2016.

[248] Ning Zhang, Kun Sun, Wenjing Lou, and Y Thomas Hou. Case: Cache-assisted secure execution on arm processors. In 2016 IEEE Symposium on Security and Privacy (SP), pages 72–90. IEEE, 2016.

[249] Ning Zhang, Jin Li, Wenjing Lou, and Y Thomas Hou. Privacyguard: Enforcing private data usage with blockchain and attested execution. In Data Privacy Management, Cryptocurrencies and Blockchain Technology, pages 345–353. Springer, 2018.

[250] Ning Zhang, Kun Sun, Deborah Shands, Wenjing Lou, and Y Thomas Hou. Trusense: Information leakage from trustzone. In IEEE INFOCOM 2018-IEEE Conference on Computer Communications, pages 1097–1105. IEEE, 2018.

[251] Zhengquan Zhang, Yue Xiao, Zheng Ma, Ming Xiao, Zhiguo Ding, Xianfu Lei, George K Karagiannidis, and Pingzhi Fan. 6g wireless networks: Vision, requirements, architecture, and key technologies. IEEE Vehicular Technology Magazine, 14 (3):28–41, 2019.

[252] Zhou Zhao, James Cheng, and Wilfred Ng. Truth discovery in data streams: A single-pass probabilistic approach. In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, pages 1589–1598. ACM, 2014.

[253] Wenting Zheng, Ankur Dave, Jethro G. Beekman, Raluca Ada Popa, Joseph E. Gonzalez, and Ion Stoica. Opaque: An oblivious and encrypted distributed analytics platform. In 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17), pages 283–298, Boston, MA, 2017. USENIX Association. ISBN 978-1-931971-37-9.

[254] Guy Zyskind, Oz Nathan, and Alex Pentland. Enigma: Decentralized computation platform with guaranteed privacy. arXiv preprint arXiv:1506.03471, 2015.

[255] Guy Zyskind, Oz Nathan, and Alex 'Sandy' Pentland. Decentralizing privacy: Using blockchain to protect personal data. In Security and Privacy Workshops (SPW), 2015 IEEE, pages 180–184. IEEE, 2015.