Dissertation

Eduardo Oliveira

March 17, 2025

1 Decentralization and Distributed Systems

Decentralization and distributed systems are fundamental paradigms that redefine how data is processed, stored, and verified across various domains. Unlike traditional architectures that rely on a centralized entity to manage operations, decentralized systems distribute control among multiple participants. This approach enhances fault tolerance [25], ensures system resilience, and mitigates risks associated with single points of failure. Distributed systems, in turn, rely on a network of interconnected nodes to collectively maintain and process data, enabling scalability and redundancy [10]. These principles underpin various modern technologies, including blockchain [34] and decentralized file storage networks [41], which eliminate reliance on centralized intermediaries and foster transparency and security.

1.1 Decentralization and Distributed Systems the foundation for Blockchain and IPFS

Blockchain technology embodies decentralization by ensuring that no single entity controls data integrity and transaction validation. Transactions are recorded on a distributed ledger that is maintained collectively by network participants through consensus mechanisms [34]. By employing cryptographic techniques [21] and game-theoretic incentives [39], blockchain achieves trustless verification, preventing unauthorized modifications while ensuring transparency.

Similarly, the InterPlanetary File System (IPFS) leverages distributed system principles to provide decentralized data storage [4]. Unlike conventional file storage, which relies on centralized servers, IPFS distributes files across a peer-to-peer network, addressing them based on their content rather than location. This approach not only ensures data persistence but also enhances accessibility by allowing multiple nodes to host and retrieve the same content. In contrast to blockchain, which primarily records transactions and state changes, IPFS enables efficient and scalable storage of large data objects, complementing blockchain's immutability with a robust storage layer.

Both blockchain and IPFS exemplify the synergy between decentralization and distributed computing. Blockchain secures and verifies data integrity, while

IPFS ensures scalable, redundant storage, collectively forming the foundation for decentralized applications, provenance tracking, and secure data sharing.

1.2 Decentralization and Distributed Systems in the Open Science Platform

The Open Science Platform integrates decentralized and distributed technologies to enhance scientific transparency, reproducibility, and accessibility. Traditional research infrastructures often suffer from data silos, paywalled access, and risks of data loss or manipulation. By leveraging blockchain and IPFS, the platform ensures that research artifacts remain tamper-proof, permanently accessible, and verifiable.

Blockchain serves as a provenance-tracking mechanism by recording immutable hashes of research data, ensuring the integrity and authenticity of published findings. Researchers can timestamp and digitally sign experimental protocols, datasets, and publications, guaranteeing that any modifications are publicly traceable [6]. IPFS complements this functionality by hosting scientific artifacts in a decentralized manner, preventing single points of failure and enabling unrestricted access to research outputs.

Through the integration of these technologies, the Open Science Platform mitigates the risks associated with centralized control in research dissemination. Traditional repositories may impose restrictions on data access, suffer from institutional biases, or become unavailable over time. In contrast, a decentralized infrastructure empowers researchers to share knowledge freely, ensuring that scientific progress remains transparent and universally accessible.

Decentralization and distributed systems redefine how data integrity, accessibility, and transparency are maintained across various domains. Blockchain and IPFS provide complementary solutions that enhance security, immutability, and scalability. In the context of Open Science, these technologies eliminate reliance on centralized institutions, ensuring that research artifacts remain verifiable and permanently accessible. By leveraging decentralization, the Open Science Platform fosters an ecosystem of trustless collaboration, where scientific knowledge can be openly shared and validated by the global research community. To fully grasp the impact of these technologies, it is essential to examine their core components, underlying mechanisms, and real-world applications. The following sections explore blockchain fundamentals, decentralized applications (dApps), and IPFS, detailing how each contributes to building resilient and transparent digital infrastructures.

2 Blockchain

2.1 Foundational Aspects

Blockchain is a decentralized, immutable ledger technology designed to facilitate secure and transparent transactions within distributed networks. Initially con-

ceptualized for the Bitcoin blockchain [34], this technology has since evolved into a multi-purpose infrastructure underpinning various domains, including finance, supply chain management, and digital identity verification.

The development of blockchain, however, did not occur in isolation. The concept of a cryptographically secured chain of blocks predates Bitcoin and draws from earlier research on distributed consensus and cryptographic techniques. A key component in blockchain structures is the Merkle tree, introduced by Ralph Merkle in the 1980s [28]. These trees enable efficient data integrity verification by organizing hashes in a hierarchical structure, which is crucial for maintaining the integrity of blockchain data.

Building on these foundational cryptographic concepts, Stuart Haber and W. Scott Stornetta proposed a method for securely time-stamping digital documents in 1991 [17]. This innovation was significant because it prevented backdating and tampering, laying the groundwork for immutable records. In 1992, Haber, Stornetta, and Bayer further refined this approach by incorporating Merkle trees into their time-stamping system, thereby improving efficiency and strengthening security [3]. These advancements not only contributed to the development of blockchain but also highlighted the potential of decentralized, immutable ledgers for maintaining verifiable records.

The inherent properties of blockchain—decentralization, immutability, transparency, and security—make it particularly well-suited for addressing challenges in scientific reproducibility. By maintaining an auditable and tamper-proof history of research data and workflows, blockchain ensures long-term verifiability and integrity. This application leverages the foundational principles established by early cryptographic and distributed systems research, demonstrating how blockchain can extend beyond financial transactions to improve scientific research reproducibility.

2.2 Public and Private Blockchains

Blockchains can be classified based on their accessibility and governance models, primarily into public and private blockchains.

2.2.1 Public Blockchains

Public blockchains, such as Bitcoin and Ethereum, are open to anyone, allowing unrestricted participation in the network. These blockchains prioritize decentralization and security at the expense of scalability. Consensus mechanisms in public blockchains typically rely on PoW or PoS, where participants must follow established protocols to validate transactions [12]. Public blockchains offer transparency, as all transactions are recorded on a publicly accessible ledger, making them suitable for applications requiring trustless environments.

2.2.2 Private and Permissioned Blockchains

Private blockchains restrict participation to authorized entities, making them suitable for enterprise applications. Hyperledger Fabric and Hyperledger Iroha are examples of permissioned blockchain frameworks designed for regulated environments where identity verification and compliance are critical [8]. Private blockchains provide improved scalability and efficiency since they do not require energy-intensive consensus mechanisms like PoW. However, they trade off decentralization, as a governing authority typically oversees the network.

2.2.3 Hybrid Blockchain Models

Hybrid blockchain models combine aspects of both public and private blockchains. These architectures allow organizations to maintain a private ledger while interacting with public networks for verification and transparency purposes. Such approaches are particularly useful in industries where both privacy and auditability are required, such as supply chain management and financial services [2]. By integrating permissioned chains with public networks, hybrid models balance the need for data confidentiality with the benefits of public verification.

2.3 Consensus Mechanisms in Blockchain

Consensus mechanisms are fundamental to blockchain networks, ensuring agreement among distributed nodes without requiring centralized authority. These mechanisms validate transactions and maintain the integrity of the ledger, preventing issues such as double-spending and malicious attacks.

2.3.1 Proof of Work (PoW)

Proof of Work (PoW) was first implemented in Bitcoin [34] and remains one of the most well-known consensus mechanisms. PoW requires network participants, known as miners, to solve complex cryptographic puzzles using computational resources. The first miner to find a valid solution can append a new block to the blockchain and receive a block reward. This process ensures security but comes at the cost of significant energy consumption [35]. Additionally, the difficulty adjustment mechanism ensures that blocks are produced at a steady rate by modifying the complexity of the puzzle based on the total computational power of the network.

2.3.2 Proof of Stake (PoS)

Proof of Stake (PoS) is a consensus mechanism designed to address the scalability and energy inefficiencies of Proof of Work (PoW). Its development reflects a paradigm shift in blockchain security and governance, with roots in early cryptocurrency discourse and iterative improvements over time. The concept emerged in 2011 through discussions on the Bitcointalk forum, predating its formal implementation. The first practical application appeared in 2012 with

PPCoin (later Peercoin) [23]. While Peercoin pioneered PoS, it adopted a hybrid PoW/PoS model to bootstrap initial security, where PoW mined the first blocks, and PoS secured subsequent ones. The first pure PoS implementation arrived in 2013 with NXT Blackcoin [11], which eliminated mining entirely and relied solely on staking. These projects demonstrated that decentralized consensus could be achieved without energy-intensive computations, setting the stage for modern PoS systems like Ethereum 2.0 [12]and Cardano [22].

2.3.3 Mining and Block Validation

Mining is the process by which transactions are validated and added to a blockchain. In PoW-based systems, miners compete to solve cryptographic puzzles, while in PoS-based systems, validators are selected to propose and confirm blocks based on their stakes. Mining serves two key purposes: securing the network by making attacks computationally expensive and issuing new tokens as rewards. This incentive structure aligns participant behavior with the network's security goals [6]. For example, Bitcoin employs PoW mining, while Ethereum 2.0 uses PoS validation.

2.3.4 Byzantine Fault Tolerance (BFT)

Byzantine Fault Tolerance (BFT) is a property of distributed systems that allows them to function correctly even if some nodes act maliciously or fail [25]. Traditional consensus mechanisms, such as Practical Byzantine Fault Tolerance (PBFT) [9], require $2f + 1$ honest nodes out of $3f + 1$ total nodes to tolerate f Byzantine faults. PBFT-based systems provide high efficiency and finality but require a known set of validators, making them more suitable for permissioned blockchains like Hyperledger Iroha.

Hyperledger Iroha incorporates a specialized BFT consensus mechanism called YAC (Yet Another Consensus), which is optimized for voting-based block validation and low-latency operations. This integration ensures that Iroha can achieve consensus efficiently while maintaining the robustness expected of BFT systems.

2.4 YAC Consensus and Byzantine Fault Tolerance

The YAC (Yet Another Consensus) algorithm ensures Byzantine Fault Tolerance (BFT) [33] by employing a voting-based mechanism to achieve consensus in permissioned blockchain networks. YAC achieves BFT through these steps:

2.4.1 Voting for Block Hash

Validators in the network vote on the hash of the proposed block rather than its entire content. This reduces communication overhead while ensuring consistency among honest nodes.

2.4.2 Fault Tolerance

YAC tolerates Byzantine faults by requiring a supermajority (e.g., 2/3 of validators) to agree on the block hash. This guarantees that even if some nodes act maliciously or fail, the system can still reach consensus.

2.4.3 Finality

Once a supermajority is reached, the block is considered finalized, and all honest nodes accept it as part of the blockchain. This prevents forks and ensures the integrity of the ledger.

2.4.4 Permissioned Design

YAC is specifically designed for permissioned blockchains, where validators are pre-approved entities. This controlled environment enhances security and reduces the likelihood of large-scale malicious attacks.

YAC's lightweight design and focus on efficient communication make it suitable for enterprise-grade applications, such as Hyperledger Iroha, where high performance and reliability are critical.

2.5 Consensus Algorithm for YAC

The YAC (Yet Another Consensus) algorithm ensures Byzantine Fault Tolerance (BFT) through a voting-based process, as follows:

2.5.1 Fault Tolerance

The network must satisfy:

$$n \geq 3f + 1$$

Where:

- n: Total number of nodes in the network.
- f: Maximum number of Byzantine (malicious or faulty) nodes tolerated.

This ensures that the network can tolerate up to f Byzantine nodes while still achieving consensus.

2.5.2 Supermajority Agreement

For a block to be finalized, a supermajority of nodes must agree on its hash:

$$v > \frac{2n}{3}$$

Where:

- v: Number of votes for the block hash.
- n: Total number of nodes.

This condition guarantees both safety and liveness in consensus.

2.5.3 Voting Process

The voting process involves two phases:

1. **Proposal Phase**: A leader node proposes a block hash.
2. **Voting Phase**: Nodes vote on the proposed hash, and votes are collected until the supermajority condition is met.

2.5.4 Finality

Once a block achieves supermajority agreement, it is considered finalized and added to the blockchain. This ensures that all honest nodes accept the same block, preventing forks.

2.6 Conclusion

Consensus mechanisms, mining, and fault tolerance strategies are critical in ensuring blockchain security and functionality. The choice between PoW, PoS, and BFT-based approaches impacts the efficiency, decentralization, and security of blockchain networks. Furthermore, the distinction between public, private, and hybrid blockchains influences their applications, with public blockchains prioritizing trustlessness and private blockchains emphasizing control and efficiency. Understanding these foundational aspects enables the development of blockchain-based solutions tailored to the needs of Open Science and research reproducibility.

2.7 Public Key Cryptography in Blockchain

Public key cryptography plays a fundamental role in securing blockchain networks by enabling secure transactions, identity verification, and data integrity without requiring a centralized authority [35]. It forms the foundation for digital signatures, key management, and encryption mechanisms that ensure trust and security in decentralized environments.

2.7.1 Role of Public Key Cryptography in Blockchain

Blockchain networks rely on asymmetric cryptography, also known as public key cryptography, to authenticate and authorize transactions [38]. Each participant in the network possesses a pair of cryptographic keys: a **public key**, which serves as an address that others can use to send transactions, and a **private key**, which is used to sign transactions and prove ownership. When a user initiates a transaction, they generate a **digital signature** using their private key, allowing other participants to verify the authenticity of the transaction without revealing the private key itself [27].

This mechanism ensures that only the rightful owner of an asset can authorize its transfer, preventing fraud and unauthorized access [44]. Additionally,

cryptographic hashing techniques complement public key cryptography by ensuring data integrity and linking transactions in an immutable ledger [29].

2.7.2 Commonly Used Cryptographic Ciphers and Standards

Several cryptographic ciphers and standards are widely used in blockchain implementations to provide strong security guarantees:

- **RSA (Rivest-Shamir-Adleman):** A traditional public key cryptosystem based on the difficulty of factoring large prime numbers [38]. While RSA is widely used in general cryptographic applications, its key sizes are relatively large compared to modern alternatives, making it less practical for blockchain applications.
- **Elliptic Curve Cryptography (ECC):** A more efficient asymmetric cryptography scheme that provides the same level of security as RSA but with significantly smaller key sizes [31]. This efficiency makes ECC the preferred choice for blockchain applications.
- **ECDSA (Elliptic Curve Digital Signature Algorithm):** A widely adopted digital signature scheme based on ECC, used in Bitcoin and Ethereum to secure transactions [20].
- **EdDSA (Edwards-curve Digital Signature Algorithm):** A modern alternative to ECDSA, known for its faster signature verification and improved security properties [5]. It is used in newer blockchain protocols like Monero.
- **X25519:** A secure key exchange protocol based on Curve25519, commonly used in cryptographic operations for secure communication in blockchain applications [26].
- **Ed25519 with SHA-3:** Hyperledger Iroha natively employs Ed25519, a variant of EdDSA, combined with SHA-3 for enhanced security. This cryptographic combination ensures efficient key pair generation, digital signatures, and verification processes tailored to Iroha's permissioned blockchain environment [19].

2.7.3 Elliptic Curve Cryptography (ECC) in Blockchain

Elliptic Curve Cryptography (ECC) is a type of public key cryptography that leverages the mathematical properties of elliptic curves over finite fields to provide strong security with smaller key sizes [24]. The security of ECC is based on the **Elliptic Curve Discrete Logarithm Problem (ECDLP)**, which is computationally hard to solve [18].

In blockchain systems, ECC is primarily used for:

1. **Digital Signatures:** ECC enables the creation of compact and secure digital signatures, such as those used in Bitcoin (ECDSA) and newer blockchain protocols (EdDSA and Ed25519) [20, 5].

2. **Key Pair Generation:** Blockchain wallets generate private-public key pairs using elliptic curves, ensuring that users can securely sign and verify transactions [45]. For example, Hyperledger Iroha specifically utilizes Ed25519 with SHA-3 for key pair generation, providing robust security and efficiency [19].

3. **Scalability and Efficiency:** Due to its small key size and lower computational requirements, ECC allows blockchain networks to process transactions more efficiently while maintaining security [13].

Bitcoin, for instance, uses the **secp256k1** elliptic curve for key generation and signing, which provides a 256-bit key length offering high security with lower processing overhead compared to traditional cryptographic methods [7]. In contrast, Hyperledger Iroha leverages Ed25519 with SHA-3 to ensure a balance between performance, security, and compatibility with modern cryptographic best practices [19].

Conclusion

Public key cryptography is a cornerstone of blockchain security, enabling authentication, digital signatures, and secure communication. The adoption of ECC and its derivatives, such as ECDSA and EdDSA, has significantly improved the efficiency and scalability of blockchain networks, making them resilient to attacks while minimizing computational and storage costs. Future blockchain advancements may incorporate more advanced cryptographic techniques, including post-quantum cryptography, to further enhance security in decentralized systems.

3 Introduction to Smart Contracts

Smart contracts represent a fundamental technological innovation that automates and enforces agreements between parties without requiring trusted intermediaries. First conceptualized by Nick Szabo in the late 1990s, smart contracts were envisioned as self-executing contractual arrangements, where terms could be translated into code and automatically enforced by the system itself[42]. Szabo illustrated this concept using a vending machine analogy, where the machine's mechanism guarantees the delivery of goods upon receiving the correct payment, requiring minimal trust between parties[43].

4 Technical Framework and Operation

At a technical level, smart contracts are programs that encode business logic and operate on specialized virtual machines embedded within blockchain or distributed ledger infrastructures. Their implementation follows a structured process:

Business requirements are defined collaboratively between stakeholders and developers. Conditions triggering contract execution are established (e.g., payment authorization, delivery confirmation). Development teams code these conditions and responses using specialized platforms. Security testing and validation are performed before deployment. Once deployed, contracts monitor event data from trusted sources ("oracles"). Upon fulfillment of pre-defined conditions, the contract automatically executes[40].

This automation eliminates the need for intermediaries while providing transparency, immutability, and trustless verification of transactions. All executions are recorded on the blockchain, creating an immutable audit trail that enhances accountability.

5 Smart Contracts in Decentralized Applications (dApps)

Smart contracts form the backbone of decentralized applications (dApps) by providing the autonomous business logic that operates independently of centralized control. In the dApp ecosystem, smart contracts enable trustless interactions where participants engage in complex transactions without requiring mutual trust. Smart contracts also provide transparent Governance, all network participants have visibiltiy of the encoded rules. Assets can be transferred automatically when conditions are met enabling automated value exchange and complex multi-party agreements can be executed achieving a decentralized decision making process[1].

6 Hyperledger Implementations

Within the Hyperledger ecosystem, multiple implementations of smart contract frameworks exist, each with distinct approaches and capabilities.

6.1 Hyperledger Fabric

Hyperledger Fabric has evolved to support Ethereum Virtual Machine (EVM) bytecode smart contracts, allowing contracts to be written in languages such as Solidity or Vyper. This enhancement, introduced in version 1.3, enables developers to migrate or create decentralized applications for permissioned platforms, expanding the framework's versatility[14].

6.2 Hyperledger Burrow

Hyperledger Burrow represents a permissioned Ethereum-compatible blockchain that executes smart contract code on a permissioned virtual machine. Built on a Tendermint consensus engine, Burrow provides transaction finality and high

throughput for proof-of-stake systems. Burrow serves three key functions in the Hyperledger ecosystem:

A bridge to the Tendermint/Cosmos ecosystem. An Ethereum side-chain with compatibility for advanced smart contract languages. A lightweight, hackable EVM/Solidity execution library[15].

As a fully-fledged blockchain and smart contract framework, Burrow incorporates a Unix-style permissioning model directly into its EVM implementation, allowing granular control over actions such as sending tokens, creating contracts, or becoming validators[32].

6.3 Hyperledger Iroha

Hyperledger Iroha takes a distinctive approach to smart contracts. In Iroha v1, a limited set of commands was provided without Turing-complete computation capabilities. However, Iroha v2 introduces Iroha Special Instructions (ISI) to enable Turing-complete computation while maintaining ease of use for common tasks.

Unlike Ethereum's model of deploying contracts that users interact with, Iroha employs a data model of domains, accounts, and assets that change through various interactions. ISI functions more like database triggers than deployed smart contracts, with event triggers placed on transactions that match specific parameters[16].

7 Smart Contracts for Scientific Research Reproducibility

The application of smart contracts extends beyond financial transactions to scientific research, where they can significantly enhance reproducibility. By encoding experimental protocols, data collection methodologies, and analysis procedures as smart contracts, researchers can create immutable records of their methods, ensuring that others can reproduce their work precisely[37].

Smart contracts can automate:

Data provenance tracking. Experimental parameter recording. Statistical analysis execution. Publication of results with verification.

8 Conclusion

Smart contracts represent a paradigm shift in how agreements are codified, executed, and verified in decentralized systems. From Szabo's initial conceptualization to current implementations across various blockchain platforms, smart contracts continue to evolve in sophistication and application scope. The various implementations within the Hyperledger ecosystem demonstrate the versatility of this technology, while emerging applications in scientific research highlight

its potential beyond financial use cases. As the technology matures, smart contracts will likely become an increasingly integral component of trusted digital interactions across numerous domains.

9 The InterPlanetary File System (IPFS): A Peer-to-Peer Approach to Distributed Storage

The InterPlanetary File System (IPFS) is a decentralized, peer-to-peer distributed file system designed to interconnect computing devices through a unified storage network [4]. By employing content-addressable storage and cryptographic hashing, IPFS enables the efficient sharing and retrieval of immutable data objects. Conceptually, it operates as a single, large-scale BitTorrent swarm exchanging objects within a Git-like repository. The system leverages a high-throughput, content-addressed block storage model with content-addressed hyperlinks, forming a generalized Merkle Directed Acyclic Graph (DAG). This architecture underpins various functionalities, including versioned file systems, blockchain data structures, and a more persistent and decentralized web. By integrating a Distributed Hash Table (DHT), an incentivized block exchange protocol, and a self-certifying namespace, IPFS eliminates single points of failure and minimizes the necessity for inter-node trust.

9.1 Background and Architectural Principles

IPFS is inspired by and synthesizes multiple established peer-to-peer technologies, including DHTs, BitTorrent, Git, and Self-Certifying Filesystems (SFS) [4]. While traditional web protocols such as HTTP provide an efficient means of requesting and transmitting data, they do not inherently incorporate modern file distribution strategies, resulting in limitations concerning redundancy, security, and scalability. By contrast, IPFS addresses these challenges by integrating and refining established distributed system techniques into a unified framework, ensuring efficient content retrieval and data persistence across geographically distributed nodes.

9.2 Key Architectural Components

IPFS is structured as a modular protocol stack, with distinct subsystems contributing to its overall functionality [4]:

- **Identities:** Manages node identity generation and verification, typically employing static cryptographic puzzles derived from S/Kademlia.
- **Network:** Governs peer-to-peer communication through various transport mechanisms, including WebRTC DataChannels and uTP.
- **Routing:** Facilitates object discovery and peer lookup through a DHT, defaulting to a structure based on S/Kademlia and Coral.

- **Exchange:** Utilizes BitSwap, a block exchange protocol that promotes efficient data replication through an incentive-driven market mechanism.
- **Objects:** Implements a Merkle DAG to store immutable, content-addressed objects, forming the foundation for versioned data structures.
- **Files:** Introduces a versioned file system hierarchy inspired by Git, supporting efficient data storage and retrieval.
- **Naming:** Incorporates a self-certifying mutable name system, enabling dynamic reference updates for evolving datasets.

These components collectively ensure a robust and scalable approach to distributed data management, making IPFS a compelling alternative to centralized storage solutions.

9.3 Content Addressing and Data Integrity

A fundamental principle of IPFS is its reliance on content addressing, wherein data objects are identified by the cryptographic hash of their contents [4]. This methodology not only facilitates efficient deduplication but also guarantees data integrity, as any modification to an object results in a new, distinct hash. The Merkle DAG structure further enhances the efficiency of storage and retrieval processes, allowing for seamless handling of large datasets and enabling robust versioning capabilities. By leveraging cryptographic hashing, IPFS inherently verifies data authenticity and prevents unauthorized modifications.

9.4 Routing and Peer Discovery

IPFS employs a DHT-based routing mechanism to locate peers and retrieve specific data objects [4]. This approach enables highly scalable and decentralized object discovery, as network participants contribute to maintaining a distributed index of content addresses. Small data values (typically under 1KB) can be stored directly within the DHT, whereas larger datasets are distributed across multiple nodes, optimizing both redundancy and retrieval speed.

9.5 Block Exchange and Incentive Mechanisms

The BitSwap protocol governs IPFS's block exchange mechanism, ensuring efficient data distribution and replication [4]. BitSwap operates as a marketplace where nodes trade blocks with one another, prioritizing the replication of rarer blocks to optimize availability. This economic model incentivizes participation and data persistence, ensuring that frequently accessed content remains accessible without relying on a centralized infrastructure.

9.6 Security and Node Authentication

Security within IPFS is underpinned by a cryptographic identity system, where nodes are identified by the hash of their public key [4]. This ensures that nodes can authenticate one another upon connection, exchanging public keys to verify identities. Additionally, IPFS adopts a flexible multihash format for cryptographic digests, allowing for algorithmic adaptability as cryptographic standards evolve.

9.7 Integration with the Open Science Platform

The Open Science Platform relies on principles of transparency, accessibility, and reproducibility, necessitating a decentralized and verifiable data storage framework. IPFS aligns with these objectives by offering a secure and efficient mechanism for storing and sharing research data in an immutable, content-addressed format. The use of Merkle DAGs ensures the integrity and versioning of scientific datasets, preventing data manipulation and enhancing reproducibility. Furthermore, IPFS's decentralized nature mitigates concerns associated with data loss, institutional control, and access restrictions. By integrating IPFS into the Open Science Platform, researchers can leverage a distributed, trustless system for publishing, archiving, and verifying scientific outputs, ultimately fostering an ecosystem that supports open and verifiable research practices.

In the context of the Open Science Platform, IPFS serves not only as a distributed file repository but also as a crucial component in managing metadata associated with users and projects. By leveraging content addressing through cryptographic hashing, IPFS ensures that all stored files and metadata are uniquely identified and verifiable. The Content Identifier (CID) plays a fundamental role in this process, serving as a persistent reference that links stored data to blockchain records. This integration enhances data integrity and traceability, as each CID is immutably recorded on the blockchain, ensuring that files remain accessible, unaltered, and verifiable over time. Through this approach, IPFS contributes to the Open Science Platform's core objectives by supporting transparency, reproducibility, and the secure storage of research artifacts.

9.8 Conclusion

IPFS represents a paradigm shift in distributed storage and data sharing, offering a robust alternative to conventional centralized systems. By employing content addressing, cryptographic verification, and peer-to-peer networking, IPFS enhances data integrity, security, and availability. Its modular architecture, inspired by proven peer-to-peer technologies, positions it as a versatile solution for numerous applications, including blockchain, versioned file systems, and decentralized web infrastructure. Within the context of the Open Science Platform, IPFS provides a foundational layer for ensuring research transparency and reproducibility, reinforcing the principles of Open Science through decentralized, verifiable, and persistent data management.

10 Mathematical Foundations of IPFS

10.1 Cryptographic Hashing Functions

The security and functionality of IPFS rely heavily on cryptographic hashing. Specifically, IPFS typically employs the SHA-256 algorithm [27], which generates a fixed-size (256-bit) output from variable-sized input data. The mathematical properties of cryptographic hash functions that make them suitable for IPFS include:

- Determinism: The same input always produces the same output hash.
- Pre-image resistance: Given a hash value h, it is computationally infeasible to find any input x such that $\text{hash}(x) = h$.
- Second pre-image resistance: Given an input x_1, it is computationally infeasible to find a different input x_2 such that $\text{hash}(x_1) = \text{hash}(x_2)$.
- Collision resistance: It is computationally infeasible to find any two different inputs x_1 and x_2 such that $\text{hash}(x_1) = \text{hash}(x_2)$ [36].

11 IPFS and Blockchain Complementarity

Blockchains excel at providing distributed consensus and immutable transaction records but face significant challenges when storing large volumes of data. Knowledge files vary in size and type, and storing them directly on a blockchain would cause data volume to surge, creating unsustainable storage pressure [30, 46]. This issue arises because blockchains are not designed to handle the large and diverse datasets typically associated with decentralized applications.

11.1 Addressing Blockchain Storage Limitations

IPFS addresses this limitation by providing a complementary distributed storage layer. Only content hashes (CIDs) need to be stored on the blockchain, rather than the full content itself. This approach dramatically reduces blockchain storage requirements while maintaining cryptographic links to the original content [4, 44]. By using content addressing, IPFS ensures data integrity while alleviating the storage burden on the blockchain [47].

References

[1] Maher Alharby and Aad van Moorsel. Blockchain-based smart contracts: A systematic mapping study. *Computer Science & Information Technology*, pages 125–140, 2017.

[2] Alessio Maria Attanasi and Maria Grazia Pazienza. Hybrid blockchains: Survey and classification. *Information*, 11(4):222, 2020.

[3] Dave Bayer, Stuart Haber, and W. Scott Stornetta. Improving the efficiency and reliability of digital time-stamping. *Sequences II: Methods in Communication, Security, and Computer Science*, pages 329–334, 1993.

[4] Juan Benet. Ipfs - content addressed, versioned, p2p file system. *arXiv preprint*, arXiv:1407.3561, 2014.

[5] D.J. Bernstein, T. Lange, and R. Niederhagen. High-speed high-security signatures. *Journal of cryptographic engineering*, 2(2):77–89, 2012.

[6] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. *Proceedings - IEEE Symposium on Security and Privacy*, pages 104–121, 2015.

[7] D.R. Brown. Standards for efficient cryptography (sec) 1: Elliptic curve cryptography. Technical report, Certicom Research, 2010.

[8] Christian Cachin. Architecture of the hyperledger blockchain fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers (DCCL)*, 2016.

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

[10] George Coulouris, Jean Dollimore, Tim Kindberg, and Jonathon Blair. *Distributed Systems: Concepts and Design*. Addison-Wesley, 2011.

[11] Osher Deri. From peercoin and nxt to algorand - a history of proof of stake. misc article, 2020.

[12] Vitalik Buterin et al. Combining ghost and casper. *arXiv preprint arXiv:2003.03052*, 2020.

[13] C.F. Fan, Y. Chen, R. Chen, H. Zhou, and L. Chen. Analysis of elliptic curve cryptography for blockchain. *Future Generation Computer Systems*, 89:735–743, 2018.

[14] Hyperledger Foundation. Hyperledger fabric now supports ethereum, 2018. Accessed: 2025-03-17.

[15] Hyperledger Foundation. Burrow - the boring blockchain, 2019. Accessed: 2025-03-17.

[16] Hyperledger Foundation. Iroha special instructions dsl, 2020. Accessed: 2025-03-17.

[17] Stuart Haber and W. Scott Stornetta. How to time-stamp a digital document. *Journal of Cryptology*, 3(2):99–111, 1991.

[18] D. Hankerson, A.J. Menezes, and S.A. Vanstone. *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.

[19] Hyperledger Foundation. *Hyperledger Iroha Documentation*, 2025. Accessed: 2025-03-17.

[20] D. Johnson, A. Menezes, and S. Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International Journal of Information Security*, 1(1):36–63, 2001.

[21] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC, 3rd edition, 2020.

[22] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: The underlying science behind cardano's proof-of-stake blockchain. *Cryptology ePrint Archive*, 2020:352, 2020.

[23] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. White paper, 2012.

[24] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[25] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[26] A. Langley, M. Hamburg, and M. Naehrig. Curve25519: new diffie-hellman speed records. *International Association for Cryptologic Research*, 2016.

[27] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, 1996.

[28] Ralph C. Merkle. A digital signature based on a conventional encryption function. In *Advances in Cryptology — CRYPTO '87*, volume 293, pages 369–378. Springer Berlin Heidelberg, Berlin, Heidelberg, 1988.

[29] R.C. Merkle. A digital signature based on a conventional encryption function. *Advances in cryptology—CRYPTO'87*, pages 369–378, 1988.

[30] Arthur Miller, David S. Rowe, and Alex Papageorgiou. Scaling blockchain storage and its applications. *Journal of Computer Science and Technology*, 31(4):873–886, 2016.

[31] V.S. Miller. Use of elliptic curves in cryptography. *Advances in cryptology—CRYPTO'85*, pages 417–426, 1986.

[32] Monax. Hyperledger burrow, 2020. Accessed: 2025-03-17.

[33] Fedor Muratov, Andrei Lebedev, Nikolai Iushkevich, Bulat Nasrulin, and Makoto Takemiya. YAC: BFT Consensus Algorithm for Blockchain, September 2018. arXiv:1809.00554 [cs].

[34] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[35] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder. *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press, 2016.

[36] National Institute of Standards and Technology. Fips pub 180-4: Secure hash standard (shs), 2012. Accessed: 2025-03-17.

[37] Sina Pilehchiha. Improving reproducibility in smart contract research. Master's thesis, Concordia University, 2022. Accessed: 2025-03-17.

[38] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[39] Tim Roughgarden. *Twenty-One Lectures on Algorithmic Game Theory*. Cambridge University Press, 2016.

[40] Simplilearn. What is a smart contract in blockchain and how does it work?, 2022. Accessed: 2025-03-17.

[41] M. Sookhak, H. Talebian, and M. W. Storer. Decentralized storage systems: A survey. *ACM Computing Surveys*, 54(6):1–36, 2021.

[42] Nick Szabo. Smart contracts: Building blocks for digital markets. *EXTROPY: The Journal of Transhumanist Thought*, 16, 1996.

[43] Nick Szabo. Formalizing and securing relationships on public networks. *First Monday*, 2(9), 1997.

[44] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum White Paper*, 2014.

[45] Y. Wu, X. Zheng, J. Ren, K. Zhang, and Y. Shen. Blockchain-based secure key management scheme with high availability. *IEEE Access*, 6:27766–27774, 2018.

[46] X. Xu, L. Yang, and L. Zhao. A survey of blockchain data storage issues and solutions. *Journal of Computer Networks and Communications*, 2018:1–17, 2018.

[47] X. Zhang, H. Li, and B. Xu. Decentralized data storage systems: Blockchain and beyond. *Journal of Blockchain Research*, 3(2):45–56, 2020.