Trade-off betweew security and scalability in blockchain systems

Kahina Khacef

Thèse de doctorat de Sorbonne Université

École Doctorale Informatique, Télécommunications et Électronique
ED130

Trade-off Between Security and Scalability in Blockchain Systems

Présentée par

Kahina KHACEF

Pour obtenir le grade de

DOCTEUR de SORBONNE UNIVERSITE

Date de soutenance : 09 Décembre 2022

Composition du jury :

Mme Hanene Azzag	Université Sorbonne Paris Nord	Rapporteur
Mme Valentina Dragos	ONERA	Rapporteur
Mme Daniela Grigori	Université Paris Dauphine	Examinatrice
Mr Walid Gaaloul	Politechnique de Paris	Examinateur
Mr Mourad Ouziri	Université Paris Cité	Co-Encadrant
Mme Salima Benbernou	Université Paris Cite	Directrice

Remerciements

Je tiens à remercier tous ceux qui m'ont soutenue et aidée durant ma thèse. Je souhaite exprimer ma plus grande gratitude aux directeurs de l'école doctorale pour leur soutien, leur professionnalisme et leur mérite. Je remercie également la Sorbonne Université pour m'avoir soutenue. Je remercie également mes collègues à la CGT, mes amis de l'équipe Syel et de l'équipe Phare qui m'ont aidée et encouragée durant cette période difficile. Enfin, je remercie ma banque qui m'a accordé des prêts étudiants pour pouvoir financer mes déplacements et assister à des congrès. Je suis très reconnaissante à tous ceux qui m'ont soutenue et aidée durant cette période.

Le plus grand remerciement va à mon père, pour sa confiance et son amour inconditionnel. Ainsi qu'à ma meilleure amie Narimane, qui a partagé tous les moments, bons et difficiles, jour après jour.

Résumé : Cette thèse propose et évalue des méthodes pour décentraliser et faire évoluer la blockchain. La première contribution de cette thèse propose un protocole d'identification décentralisé et sécurisé qui profite de la puissance et de la résilience des blockchains. La clé publique et la signature sont enregistrées dans la blockchain après validation de l'identité du signataire par des smart contracts. La deuxième contribution propose SecuSca, une approche qui fait un compromis entre sécurité et évolutivité en créant un sharding dans lequel les blocs sont stockés sur différents nœuds. La troisième contribution optimise l'approche en choisissant des nœuds selon leurs capacités. Les méthodes proposées ont été évaluées expérimentalement et ont montré leurs avantages pour la décentralisation et l'évolutivité de la blockchain.
Mots clés. Blockchain publique, Preuve de travail, Décentralisation, Sharding, Sécurité, Disponibilité.

Abstract:

The development of Blockchain has enabled the emergence of high technology in the sensitive and active sectors by allowing the reliability of information via consensus, the immutability of records, and transaction transparency. This thesis presents the design, implementation, and evaluation of techniques to scale the blockchain. The first part of this thesis consists of building a decentralized, secure peer-to-peer messaging protocol using a PKI-based blockchain, which can be an email, a website, or some other form of message. Managing users' identities by the Blockchain eliminates the single point of failure of traditional PKIs. By using smart contracts to validate, store and revoke the certificate on a public blockchain. Security and scalability are considered two significant challenges in blockchains' rapid and smooth deployment in businesses, enterprises, and organizations. The ability to scale up a blockchain lies mainly in improving the underlying technology rather than deploying new hardware. The second contribution of the thesis proposes SecuSca, an approach that makes a trade-off between security and scalability when designing blockchain-based systems. It designs an efficient replication model, which creates dynamic sharding wherein blocks are stored in various nodes. To maintain the required level of security, the proposed approach shows that blockchain replication over the Peer_to_Peer network is minimized as the blockchain's length evolves. Furthermore, a sharding protocol over the network is proposed to get access to the blockchain data based on historical transactions. The protocol reduces old blocks' replication; these blocks can be discarded from specific nodes and stored by others. The nodes willing to store the coming blocks and their data are chosen randomly. The block header of each block is kept to achieve consensus. Next, we optimize the latest approach by choosing the entering nodes following the nodes' capacities instead of randomly.
Key words. Permissionless Blockchain, Proof_of_Work, Decentralization, Sharding, Security, Availability.

Contents

List of Figures

Part I.

Introduction

Chapter 1

Motivations

Blockchain-based cryptocurrencies like Bitcoin [108] and Ethereum [133] can function without a trusted central authority to verify transactions, as nodes in the network independently verify that all transactions are valid and reject any blocks containing invalid transactions. This means that the block producer (i.e. a miner) does not need to be trusted to produce blocks with valid transactions. Permissionless blockchains aim to be a persistent, distributed, consistent, and ever-growing transaction log, which is publicly auditable by anyone who can join or leave the network without permission. They achieve this by utilizing two components of consensus to ensure security and consistency: full blockchain replication and the Proof-of-Work (PoW) cryptography puzzle. The PoW is demonstrably secure against a large number of participants who wish to disrupt the system and allows a configurable and independent rate of block creation, regardless of the size of the system. Additionally, the broadcast primitive relies on Peer-to-Peer (P2P) network properties without the presence of a trusted third party, despite the malicious behavior of certain nodes.

Blockchains' capacity to be publicly verified has opened the door to a new generation of decentralized systems and computing platforms that are not reliant on trusted centralized parties. In the first contribution of this thesis 3, we propose a model for PKI based on the blockchain, to benefit from the advantages of the blockchain.

1.1 The benefits of using blockchain for storage

1. Security: Blockchain technology is secure by design, making it very difficult to tamper with or modify the data stored on the blockchain. Because of its decentralized structure, the data stored on the blockchain is replicated across the network and constantly updated, eliminating the possibility of a single point of failure and making it harder for hackers to penetrate.

2. Transparency: Blockchain transactions are visible to all participants in the network and transactions are time-stamped and immutable. This makes it easier to audit and trace back transactions to their source.

3. Efficiency: Storing data on a blockchain can reduce the time and cost of data processing. Transactions are processed almost immediately and don't require a third party to verify or approve the transaction.

4. Data Integrity: Blockchain technology is resistant to data tampering, as the data is stored in a distributed ledger and is cryptographically secured. This ensures that the data stored on the blockchain is accurate and reliable.

This blessing, however, comes with a drawback; if every node in the network processes and validates the transactions, the node with the fewest resources will limit the network's transaction throughput. Blockchains maintain a continuously-growing history of ordered information to achieve consensus and provide security to the blockchain. Additionally, storage costs increase along with the number of transactions, which could be a barrier limiting blockchain scalability and a challenge to gaining widespread adoption. Bitcoin [108] and Ethereum [133] are based on a full replication system, which means that every network node (the node responsible for validation) stores the whole historical Blockchain. Nodes validate a new transaction by checking the recorded state and then holding each transaction. This replication tolerates up to 49% of malicious nodes, allowing for a high level of security; however, this severely limits storage efficiency and scalability.

This design is crippling for the scalability of the blockchain since these nodes store the blockchain from the first block and will continue to receive new information to store.

1. **Storage capacity per full node**: Blockchain such as Bitcoin or Ethereum maintains a continuously-growing history of ordered information to achieve consensus and provide security to the blockchain. Validator nodes that participate in consensus replicate all previous validation states to prevent double-spending, used as proof of correct status to keep the system functioning. Validator nodes validate new transactions by checking their states (recorded transactions) and storing them, which requires a lot of storage space. Every validator node in those blockchains stores and processes all states and transactions. A validator node is a node in a blockchain network that can independently verify all transactions and the current state of the network. This process enhances security and maintains traceability but limits scalability because the volume of data each node must store will continue to grow. The nodes must

maintain all states and supply extra hardware and memory capacity to store the massive volumes of data for the blockchain to continue functioning.

2. **Cost of using blockchain network**: As the number of users and transactions on the blockchain network increases, nodes need to process and store more data with higher fees to maximize their earnings. In addition, since All transaction requests require fees, Miners prioritize transactions that pay higher fees. Therefore, if transactions need to be verified quickly, the user must pay higher fees to get priority. The more a user agrees to pay high transaction fees, the faster they will be processed. The Bitcoin Cash protocol [20] helps reduce transaction fees and improves transaction speed. It rejects the block size limitation introduced by Bitcoin (limited to 1 MB). Blocks in the Bitcoin cash chain correspond to a limit of 32 MB, or approximately 250 transactions per second (compared to $24,000$ transactions per second for Visa). As the blocks of the Bitcoin cash are large enough to allow miners to mine all transactions, there is no need to pay transaction fees for the transaction to be mined on the next block. Although, Bitcoin cash raises concerns regarding the difficulty of hosting a full node.

Blockchain networks often impose a strict limit on block size to allow nodes with limited resources to participate in the network. However, due to the limited on-chain capacity during peak periods, users are forced to pay higher fees as they compete to have their transactions included on the blockchain.

Many projects use cloud computing to store their Blockchain to increase scalability, e.g., Solana and Ethereum, which are stored in cloud computing due to massive block sizes, thereby sacrificing decentralization. In fact, a well-known blockchain trilemma has been raised claiming that no decentralized ledger system can simultaneously achieve i) *security*, ii) *decentralization*, and iii) *scalability*.

The solutions to scale blockchain must not sacrifice the properties of blockchain, such as decentralization and public verification. Therefore, the ability to scale a blockchain lies primarily in improving the foundation of blockchain technology, preventing the hardware shortages triggered by classical cryptocurrencies.

In this thesis, we will be interested in blockchain protocols using Proof_of_Work. In the background 2, we present the Nakamoto protocol, primarily used in Bitcoin, as well as the security and cryptography protocols it establishes to achieve a distributed and secure consensus simultaneously. In the first Chapter 3, we propose an architecture for a blockchain-based authentication system. This proposal removes trusted third parties responsible for validating and issuing digital certificates, for example, certification authorities. All authentication information is encrypted, hashed, and stored on the blockchain. This first part of the thesis illustrates the use

of blockchain in fields other than finance.

Blockchain information is stored in blocks, where each block is composed of the block header, which includes consensus data (Nonce to find the correct hash, the hash of the last previous block and the transactions hashes, the root of Merkle tree), and the application data where the block transactions are stored. Miners can create new blocks by downloading the block headers of the old blocks and the last six blocks of the longest chain (longest chain rule, see 2.5.2). Based on this consensus, we propose the SECUSCA protocol. The first version is presented in Chapter 4, and an improved version is in Chapter 5. This protocol aims to improve scalability by reducing block replication without compromising security. We have established a protocol that follows a replication function that gives the block replications at different positions. First, the algorithms choose nodes to store the blocks. The number of nodes selected is small than the total number of nodes. Then, old blocks are deleted from some nodes. Nodes store the block according to their storage capacities.

Summary :

1.2 Weaknesses of Centralized Systems

Security is a fundamental element of societies, businesses and organisations. Consumers of information technology (Internet, computing, storage, etc.) require a high level of security to ensure proper management of digital data, such as health information, digital identities and access to the Internet. Unfortunately, many attacks, such as identity theft, money theft and data theft, are frequently committed. To face this problem, governments, banks and technology companies set up systems and security standards to protect users' sensitive information. ANSSI [5] has the main mission of securing sensitive state data, providing its expertise and technical assistance to local authorities and companies. Banks ensure that customers spend money with the right people and appropriate amounts, and their lawyers check that products are not copied or distributed without authorization. It is necessary to introduce new data security frameworks, which can be based either on centralizing data to a trusted third party, or on decentralizing data to multiple parties. The most recent information systems are complex, as they combine functions, third-party libraries and continuous development. To face this growing complexity, it is essential to optimize security and analysis solutions. The most minor errors can have serious consequences, and automation of these functions can enable a thorough analysis and concentrate engineers' knowledge on the vulnerable parts of the code. Users trust centralized systems, identify themselves on digital platforms where they send and receive data, and store information in the cloud. Centralizing data could certainly unify data security efforts in a robust system, but this centralized solution presents many challenges.

- A trusted third party represents a vulnerable point for hacking through which sensitive data is transmitted. The concentration of data in one system provides an opportunity for hackers, as it allows them to access a large amount of data at once.
- Risk of corruption: Centralized systems are often managed by one entity, meaning there is no mechanism to prevent a person from corrupting the system.
- Data concentration: Centralizing data means it is stored on servers that can be vulnerable to cyber attacks and unauthorized access.
- Low reliability: Centralized systems are often prone to outages and longer downtime than distributed systems, which can lead to delays and errors.

- Low scalability: Centralized systems are often limited in terms of capacity and features, which can lead to scalability and performance issues.
- High costs: Centralized systems are often more expensive than distributed systems, as they require investments for hardware, software, and personnel.

The massive collection of personal data poses a new threat to privacy. Indeed, we are increasingly entrusting personal data to companies in exchange for free access to social networks, messaging services or other services. Technology giants such as Google, Amazon, Apple, Facebook or Microsoft (GAFAM) control the personal data of millions of users, posing serious challenges in terms of data security. They have accumulated a large amount of information in a short time, which they use with artificial intelligence algorithms to improve their products, but also to monitor and influence their users. Recent scandals such as Cambridge Analytica or Edward Snowden's revelations have highlighted the dangers of this accumulation of personal data between actors over which we have little or no control. Thus, the protection of privacy has come back to the centre of the political debate. In response to these expectations, the European Union opened the way to more stringent data trading regulations with the enactment of the GDPR in 2018, and Brazil, Japan and California have followed suit. We are at a turning point: consumer citizens are demanding better regulation of their privacy and security guarantees for their data.

1.2.1 Limitation of Public Key Infrastructure (PKI)

The security of digital exchanges is important for users who engage in digital transactions. A public key infrastructure (or PKI) issues digital certificates to perform cryptographic functions. These are used to verify and authenticate the qualifications of the various parties involved in the electronic exchange of information. A PKI contains several components, the certificate authority (CA) which validates and signs certificate requests. The CA (or Certificate Authority (CA)) is an example of a centralized authority that guarantees user identity. It is one of the main components of the PKI and the decisive and trusted element of the certification process, being responsible for the revocation lists. The registration authority (or RA) is the interface between the user and the certificate authority. It is responsible for identifying the owners of the certificate and ensuring compliance with restrictions related to the use of the certificate. The deposit authority stores digital certificates and centralizes and organizes their deposit.

Public-key encryption allows users to authenticate in order to establish trust between users and computer systems. Public-key encryption relies on the ability of entities to verify the public keys of other entities. For example, suppose a user wants

to log into their bank account using a web browser. The online session is protected by the bank's public key. If the user's browser accepts the wrong public key for the bank, an attacker can intercept the connection and steal the user's credentials.

Certificate Authorities (CAs) issue certificates that describe digital identities and provide a means to verify the authenticity of those certificates (i.e. the services of the Certification Authority are most often used to secure digital communications).

Several CAs have been compromised, such as Comodo in 2011 [43], and many fraudulent certificates have been issued and used for man-in-the-middle attacks [105, 90]. In addition to cryptographic problems [93, 12, 11], if the CA is dishonest or compromised, it can issue certificates proving the authenticity of false public keys, these keys can be generated by an attacker or by the CA itself.

Log-based PKIs: Several interesting solutions have been proposed to address these issues. Enhancements to Log-based PKI (or Log-based PKIs) allow for the creation of public logs to track the activities of the CA, ensuring transparency. In this approach, which has mainly been developed to solve the problem of poor-performing CAs, certificates are not considered valid until they are added to the public logs in an append-only fashion. Certificate Transparency (CT) [92] was the first log-based solution provided by Google to improve the accountability of CA functions and the detection of misused certificates. Everyone can verify the activity of the CA by monitoring CT logs and identifying suspicious certificates that are not included in the public logs. CT uses the Gossip protocol to ensure that the CAs leave persistent proofs of all the certificates they issue. Thus, the activities of a CA are visible ("transparent") to its users and to observers. Users will only accept a certificate if it is accompanied with a proof that it is included in the log. However, CT does not define a mechanism for revoking the registered certificates. Subsequently, extensions such as [18] and [126] are proposed to solve the revocation problem by periodically sending revocation lists to browsers and preventing certification authorities from issuing public key certificates to a domain without being visible to the domain owner. [103] and [23] ensure the transparency of the messaging system.

Such extensions usually require other entities (trusted third parties) in addition to the public protocol requirement [145, 84, 91]. Log-based PKIs have several drawbacks:

- Require a centralized and consistent source of information to operate securely (trusted third parties).
- Do not encourage enough auditing of the CA's behavior.
- Require time and manual effort to report bad CA behavior.

The central role of the certification authority in the traditional public key infrastructure makes it fragile and prone to compromise and operational failures. Main-

tenance of the certification authorities and revocation lists is important, especially in large, loosely connected systems. Log-based PKIs do not effectively solve the problem and require the use of CAs.

1.3 The solution provided by blockchain technology

Blockchain provides a decentralized solution that can eliminate the main points of failure by removing CAs and digital certificates; instead, everything is recorded on the blockchain. This technology marks a new era in the way trust is established between two individuals, it is a fully decentralized system, where trust is created by sharing and validating data between the network nodes, using old known methods such as cryptographic algorithms and peer-to-peer networks. Aliases, public keys, or digital addresses identify the nodes.,

1.3.1 The blockchain

A blockchain is a distributed ledger composed of participants who communicate with each other by exchanging messages and applying consensus to reach a common agreement without a third party of trust. Participants must record the same information in chronological order in their local system and share the information with other participants. Figure 1.4 shows the data structure of the blockchains which forms a sequence of blocks of transactions linked together by hashing to the previous block, thus creating a blockchain.

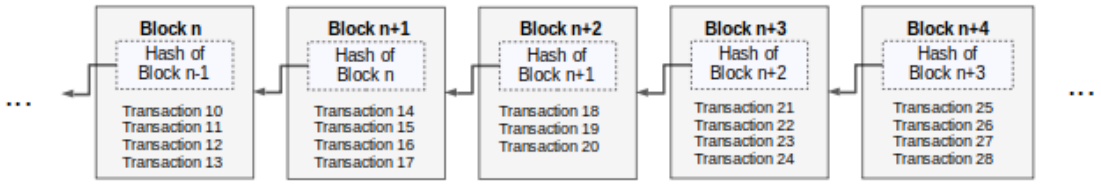


Figure 1.1: Blockchain structure.

The concept of distributed consensus allows for events to be recorded digitally in an incorruptible way so that no one can dispute their occurrence since these events are controlled by multiple independent actors. In 2008, the first cryptocurrency known as Bitcoin was introduced in the Nakamoto paper [108]. The term cryptocurrency refers to a digital currency that works properly with cryptographic methods, hence the name of cryptocurrency (crypto: for cryptography, money: for monetary transactions). Bitcoin allows users to send funds without the supervision

of a central authority by setting up a Proof_of_Work consensus to validate these exchanges, solving the double spending problem.

The Bitcoin network consists of validation nodes that attempt to validate a block of transactions by solving a complex computer problem (or puzzle). These nodes are called miners. This process is competitive and the first miner to successfully solve the puzzle will be rewarded with a certain amount of cryptocurrency for their efforts. Bitcoin has adopted blockchain technology in its protocol architecture: First, the verification of financial transactions without a third party of trust, based on cryptographic methods, in a total distribution managed by network participants interconnected by a peer-to-peer network. The validation nodes verify the entire network to ensure that: (i) the participant actually owns the amount they wish to send, (ii) they have not yet sent it to anyone else. The transactions are then added to blocks, by the miners. They compete and calculate the hash of the entire block to find the correct value that takes into account the difficulty set by consensus. Finally, the first miner to calculate the correct hash of a block below the threshold set by consensus will share his block with the rest of the network. The nodes verify the received block and add it to their chain. The block is included in the chain of all peers.

A consensus is a set of rules that allow nodes connected to a peer-to-peer network to agree on an outcome and ensure the consistency of a single state.

Most of the information on public blockchains is visible to all participants hosting the blockchain. Due to the immutability of records, the reliability of consensus-based information, and the transparency of transactions provided by the blockchain, this technology has been integrated into sensitive and active industries such as smart cities. The review [123] provides systematic literature on specific blockchain use cases and analyzes the design and prototyping of blockchain systems for a sustainable and intelligent urban future, in authentication in the Internet of Things [124], it is of interest to the medical sector [72]for therapeutic patient monitoring, as well as for the secure management and analysis of large volumes of health data [46].

We mention the main elements that blockchains use to be resilient and reliable:

- Blockchain provides a fully distributed ledger management using a peer-to-peer network, a model built in a decentralized manner so that the communication or exchange taking place therein is between nodes of the system that share the same responsibility.

- The transaction logs are included in a blockchain, where each block contains a secure one-way hashing of the previous block. A new block can only be added to the chain if a consensus decision has been taken among the peers of the network.

- The digital fingerprints of the candidate blocks play an important role in the blockchain network. With their help, it is possible to verify the data and ensure their integrity; they can demonstrate the existence of a particular data file at a given moment. Adding a digital fingerprint to the blockchain ensures that manipulation attempts are exposed, as each modification of the data file results in a completely different hash value.

1.4 Proof of work protocol rules

In blockchain networks, transactions and blocks are propagated over the P2P network using the Gossip protocol, as described in [39]. For each new block received and validated, a node announces it to peers, who will request that block if they wish to extend their local blockchain. The gossip propagation process continues until every node in the network has that block. Due to the public and permissionless nature of blockchain and pseudonymity, multiple Sybil attackers can get new identities or accounts with little effort. However, the hash power of the Proof_of_Work mechanism comes from a real hardware investment designed to mitigate Sybil attacks and cannot be easily faked. The longest chain rule means that the longest established chain can serve as a common reference to the network history. The longest chain is considered the most secure chain. The Proof_of_Work protocol improves the termination requirement with the specification of **probabilistic finality**: For every honest node, each new block is either rejected or accepted into its blockchain. A received block can still be rejected but with an exponentially decreasing probability as the blockchain continues to grow.

The Proof_of_Work consensus can be summarized by the following rules:

- Proof_of_Work: the generation of blocks requires finding a preimage to a hash function so that the hash result satisfies a difficulty that is dynamically adjusted to maintain an average block generation interval.

- Gossip: The blockchain uses the peer-to-peer network, and any transaction or block is propagated through the network using the gossip protocol. A relay node that receives a block first sends the block header to its neighbors, and if they have not yet received the block, they request the block with a get block message, see Figure 1.6. A miner who has just mined a new block knows that no other node has the block, so he sends the entire block to his neighbors in the first message.

- Block Validation: Whenever a block is sent and received in the network, the validator nodes verify that the block has followed the Proof_of_Work rules,

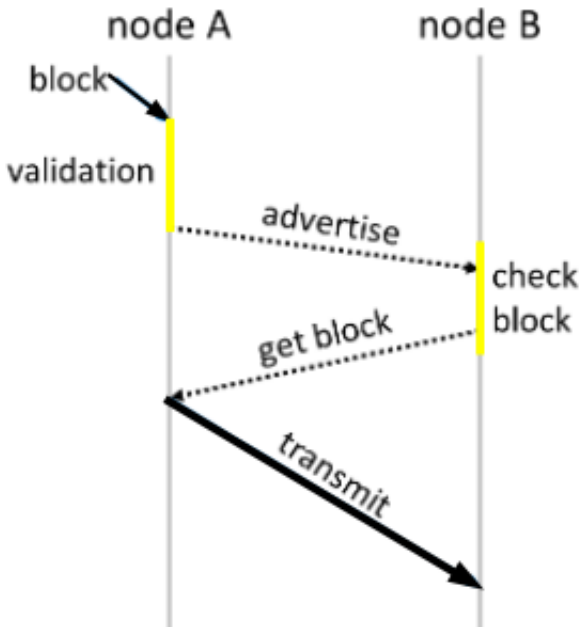


Figure 1.2: The Block Propagation Protocol in the Blockchain.

that the included transactions have not been spent in the past, and that each transaction issuer has the bitcoins they are using.

- The Longest Chain: Miners are encouraged to mine on the last block of the longest chain, as it is considered the most secure chain. There are more mined blocks, and therefore more computing power is provided.

- Mining Fees: Miners earn bitcoins, this incentive encourages miners to follow the protocol rules and compete to validate new blocks, and participate in the maintenance of the blockchain.

1.5 Blockchain based-PKI

The creation of decentralized PKI based on blockchain eliminates potential points of failure of the use of certification authorities that can compromise entire certificates [49]. Alternatives to decentralized PKI proposals include the Web of Trust (WoT). WoT offers a decentralized trust model where the authenticity of public keys and their owners is based on the degree of trust that other WoT entities have in that entity [13, 118, 54]. and ensures the transparency of certificates, revocation, and reliable transaction records. It eliminates the need for a certification authority but has deficiencies in terms of non-repudiation. Other blockchain-based PKI approaches have been proposed that use the blockchain to record data that would be signed and verified by a certification authority, thus guaranteeing the non-repudiation and

immutability of that data [135, 90, 143, 101]. They offer very limited features and the question of third-party trust has not been resolved.

1.5.1 Using blockchain for secure messaging

Using blockchain for secure messaging offers unprecedented levels of security and integrity. Blockchain technology enables a network of users to share information securely and encrypted. The data can be stored in a decentralized manner, meaning it is not controlled by a single entity. This means that the information is protected against theft, corruption, and alteration. Furthermore, blockchain enables users to verify and validate the information, making it a highly secure tool for businesses and individuals who wish to transmit confidential data.

In the first part of this thesis, we propose a blockchain-based identity verification for secure communication without a central authority, so that new keys can be securely registered or revoked based on a consensus mechanism among trust nodes of the system. A PKI that uses the blockchain as a registry by only adding to it and retaining the certification authority as proposed in [135] eliminates the main advantages of using a blockchain, as registration and revocation must still be carried out by a traditional CA. Therefore, we propose a fully decentralized method where decisions on registration and revocation of keys are taken by the nodes of the blockchain, and not by the certification authority. We use a smart contract to verify the user's identity and public keys. The smart contract can confirm the user's identity and provide trust between users for exchanging messages. Peer-to-peer communication is based on the blockchain, and all other communications are considered malicious. A smart contract is code stored and executed on the blockchain. The user can generate: 1) a public key using the ECDSA algorithm, 2) a second public key and 3) two private keys and deduce the identity as a hash of the public key. The first pair of keys are used for authentication, and the second pair of keys is used to revoke or update the first public key. The user can securely store the private key in a secure system in secret, and request to register his identity on the blockchain with the corresponding public key. This contribution is published in [80].

1.6 Limitations of storage and decentralization of Blockchains

Decentralization is the principle of blockchains. This decentralization of exchanges and data storage and network makes this technology one of the most secure. In fact, blockchains are considered to be invulnerable; they rely on each member that makes up the network: the "nodes". These elements that make up the network ensure

that the exchanges are recorded and that each host is partially or totally the owner of the blockchain data. The use of blockchain for decentralized certification and communication systems has great advantages in protecting user information without centralized control point. On the other hand, the major problem our proposal faces is the storage of identities and their associated public keys on the blockchain. As Satoshi describes in [108], the disk space required to store blocks in Bitcoin for a 10-minute block interval is 6 * 24 * 365 * block size per year. As of September 29, 2022, Bitcoin has a size exceeding 400GB. Figure 1.7 shows the linear growth of its blockchain size over the last two years. To reduce the time and capacity required for the storage and verification of keys, only the hash of our certificate is stored on the blockchain.

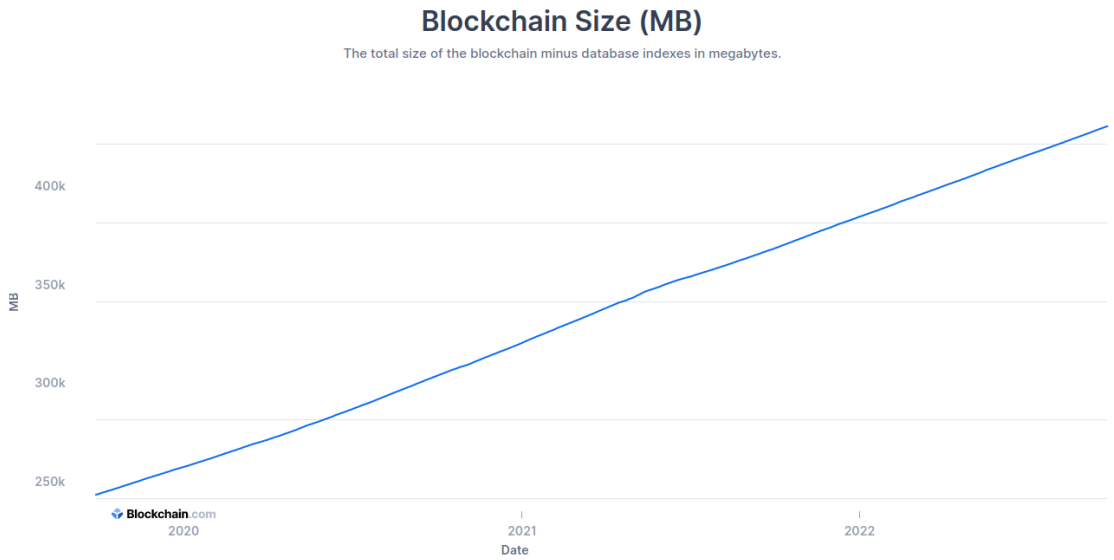


Figure 1.3: The total size of the blockchain in megabytes (MB) for Bitcoin from 2020 to 2022 [4].

The two main blockchains, Bitcoin and Ethereum (first and second generations of blockchain) continue to face issues of growing data stored on their blockchains, Ethereum's blockchain size exceeds 900GB. 65.3% of its network nodes are hosted in centralized cloud services with 51% of these nodes using Amazon AWS Cloud service, which is not geographically distributed (mainly located in the US), making the blockchain dangerously centralized. The blocks of our protocol are similar to Ethereum blocks. The third generation of blockchain attempts to correct the shortcomings of the previous ones: they solve the scalability problem with the emergence of faster blockchains to manage. Among these protocols, we find Mina [24], Solana [142], BitcoinCash, etc.

Mina, the blockchain developed by O(1) labs [24], is considered the smallest blockchain with a size of no more than 22KB. It incorporates zero-knowledge proofs

("succinct non-interactive argument of knowledge" or "Zk-SNARKs") which contain no information about the operations themselves, but guarantee the confidentiality of data and the anonymity of users, to validate transactions. These proofs were first used by the cryptocurrency Zcash [20]. At every step, Zk-SNARKS generates a proof to validate a new transaction without consulting the ledger of all previous transactions with a recursive function and builds a chain of proofs from the first block to the last, allowing a complete history to be built from simple files of a few bytes (new certificates are created from old ones). This greatly reduces the size of the data that each user and validator must download. Despite each Zk-SNARK compression reaching 1Ko, the size of the ledger should still increase over time. Similarly, Solana [142] and Polygon are designed to execute transactions in a few milliseconds with very low performance and costs. Solana is a high-speed blockchain that uses the Proof_of_History timestamping technique to achieve block production times of less than one second.

1.6.1 The size of transaction history

The decentralization of blockchains is limited by the size of the transaction history. It is impossible for a decentralized blockchain to have a database containing a history of transactions that spans years, as it would be too large to be stored and synced over a decentralized network. The size of the transaction history is a key factor limiting the feasibility of decentralizing blockchains. Furthermore, the length of the transaction history requires a significant amount of computing power to ensure that all transactions are valid and compliant with consensus rules. This can become very costly as the transaction history grows longer, which limits the use of decentralized blockchains for applications that require a large number of transactions and a long period of time. Scalability difficulty arises from the data that needs to be stored and sent over the network when the blockchain nodes sync for the finality of consensus or to join the network for the first time. Several new blockchains have emerged that enable scaling.

In the standard blockchain paradigm, transactions are completely replicated across every validator node in the network. Each validator node must store the entirety of the blockchain starting from the first block. The major challenge of designing this system with blockchain is:

- Data storage limits: Blockchain transaction records are usually of the order of kilobytes and blocks do not contain much data. Additionally, the blockchain data structure implies that all state changes are recorded in the blockchain. All the participating nodes in the network must maintain a complete copy of the blockchain. As of September 2022, Bitcoin nodes [4] require more than

400GB [3]of storage space to stay synced with the network.

- Transactions per second: The total number of transactions per block is limited by the block size. Bitcoin block is limited to the size of 1Mb, and given the average size of transactions of around 540 bytes, Bitcoin currently processes around 1950 transactions per block, in an interval of 10 minutes [68].

Although Ethereum allows the implementation of smart contracts, large amounts of data are stored in secure systems outside of the blockchain, and the cost of reading and verifying this data is extremely high [6]. Whereas the validators in Mina only need a small amount of data, the block producers responsible for creating new blocks store the entire blockchain to reach consensus. Mina therefore still has a storage growth problem. Adding to this is the decentralization problem since the transaction history is stored only in the block producers. This blockchain may have thousands of validators, but since there is only a handful of them that can access the transaction history, it is difficult to say that this blockchain is decentralized, because if something happens to this handful of nodes that store the transaction history, it could become possible to seriously manipulate the transactions on this blockchain. Solana has more than 1700 validators, but all transactions are processed by a much smaller group of 150 validators, in addition, this blockchain uses the Hetzner and OVHCloud cloud computing services to store its blockchain and this centralizes the process. Unlike the developers of Polygon who are more concerned with the decentralization of the blockchain, it uses Arweave, a decentralized storage system based on the blockchain.

Algorand [58], whose blockchain counts 1600 "participant nodes" in consensus, these transactions are managed by a smaller group of only 120 "relay nodes", most of which are managed by entities behind Algorand and its subsidiaries. In conclusion, blockchains that promise scalability, whether it be the number of transactions per second, or the use of minimal memory to validate transactions, can undermine the decentralization of the blockchain, creating small committees that control the history of these transactions, as mentioned above. Third-generation blockchains offer scalability at the expense of decentralization. Vitalik Buterin (the founder of Ethereum) developed the concept of the blockchain Trilemma triangle. It is based on the premise that it is not possible to optimize these three principles at the same time: security, scalability, and decentralization.

1.7 Solution to Scale and decentralize blockchain

1.7.1 Related work

Decentralization has been a key topic of blockchain technology since the first Bitcoin block was mined in 2009. When a network tends to be centralized, it is exposed to an increased risk. The goal is to pay particular attention to the distribution of nodes and external influences that may affect it. The fact that blockchain is based on the simultaneous work of multiple nodes precisely characterizes its security. When a blockchain is decentralized through all these layers, it is almost always very secure as there is no single point of failure. This also creates resistance to censorship as no central authority can control the digital asset.

Minor nodes in Bitcoin, and generally in all blockchains working under Proof-of-Work consensus, gather together to maintain their profits; thus, it is possible to increase the computing power by sharing costs. These groups, called mining pools, work according to their own protocol and communicate with the blockchain network via a single node. Their popularity is so great that the three largest groups represent more than 51% of the computing power. Thus, the number of units needed to perform a 51% attack is three, it is the Nakamoto coefficient, by assembling Foundry USA Pool, Antpool and F2Pool which together hold 55% of the total computing power. The higher the number is, the greater the effort to break the network is and vice versa; the lower the number is, the more vulnerable the network is.

A proposal to combat mining pools would be to introduce Sybil fees: the cost of someone using multiple nodes would be higher than the cost of using one node. This would encourage network agents to maintain only one node, thus allocating computing power more efficiently. So far, no means of implementing this solution has been found. Through some blockchains that have been transparent about the exact way their transaction histories are stored, one of the few cryptocurrencies that make this distinction is Bitcoin, whose validating nodes or full nodes store its transaction history; there are currently around 15000 Bitcoin nodes around the world, making it the most decentralized at the blockchain level. Arweave and Filecoin are two data storage solutions and have similarities such as being decentralized and based on blockchain technology. Arweave focuses on the problem of long-term data storage. Filecoin provides storage for web applications. Instead of storing data in a centralized network, Filecoin creates a decentralized network where data is replicated in multiple locations and accessible from anywhere. The main goal of Filecoin is to create a vast network of nodes to store data all over the world, which are incentivized to store other people's data. This allows data to be closer to the sources that need it and also offers no central attack point for attackers. The goal of Filecoin is that

it will become a reliable and cost-effective storage network compared to the cloud storage industry, dominated by massive companies such as Amazon Web Services (AWS) and Google.

Arweave is attempting to solve the issue of long-term data storage by providing reliable and permanent data storage. They have introduced an innovation to the current blockchain technology and have developed an incentive system so that nodes store data permanently. This makes the data both immutable and impossible to delete. If files are replicated countless times on servers all around the world, it is practically impossible to modify a file that has been uploaded onto Arweave.

Arweave has undertaken to solve the problem of long-term data storage on the Internet. It has used blockchain technology to store the data and has set up an incentive system to ensure that peers store the data permanently. Arweave has built a blockweave instead of a blockchain. The blockweave is maintained by many links within the entire data storage. Therefore, the only way to add new data to the blockweave is for a peer to add a randomly selected block already on the blockweave. Only the peers that can add a random block from the history are allowed to store new data [138]. The only way to add new data with an Arweave node is if that node has previously added a file (or group of files) to its memory and randomly selects a block already in its chain to add a new block, miners must refer to the previously stored data proving they still own it. Arweave uses Proof_of_Work and also follows the longest chain rule. In addition to including a block from the history in the following new block, it also includes a link to the last block of the chain. Miners are incentivized to store large amounts of data to increase their chances of finding the right block because the selection of blocks is random. This improvement in data storage and the economic rewards for miners who keep the data create conditions for the data to be stored for long periods of time. Furthermore, Arweave introduces a synchronization block generated once per block of 12, containing a complete list of the balance of each wallet in the system and a hash of each previous block without the transaction. The synchronization blocks help new network participants get started, and it is not necessary to download all the blocks from the genesis block.

The main similarity between Arweave and Filecoin is that they are both storage devices that use blockchain technology on a decentralized system. They are also decentralized, meaning there is no single entity controlling their servers. Instead, the servers are controlled by miners from all over the world who store the data in exchange for a fee. The miners are bound by a set of pre-established rules. The difference with Bitcoin, which uses CPU power to mine new blocks, Arweave uses computing power and storage capacity. The probability that an Arweave node mines a new block b is given by:

P(b) = P(having the block history) * P(finding the correct block hash).

This means that storage on Arweave is done on large nodes; in other words, it requires large storage memories to be able to mine in Arweave. This can create centralization around validators who must be very large nodes. Arweave incentivizes large nodes to continue storing data, and incentivizes all participants to store all block history, which is very inefficient. In Section 6, we propose a more improved solution and the storage will not depend on the memories of the nodes, our protocol will also allow small nodes to store the blocks. Since September 2022, the storage capacity of the Filecoin network is greater than 18 EiB (exbibytes), and that of Arweave is greater than 77.5 TB (terabytes). Therefore, in Chapters 4 and 5, we present improvements to highly secured and decentralized blockchain protocols that allow storing information on the blockchain, optimizing the storage memory required by the nodes.

1.7.2 SECUSCA_* protocols: Sharding transaction history

We are interested in the scalability of blockchain, which is limited by its decentralized and public nature; we formulate a method that can solve the scalability barrier of the blockchain network when the number of nodes and transactions increase, without changing the main consensus to verify and validate new records (financial or other transactions) on the blockchain and without compromising security. We have proposed SecuSca, a mechanism to reduce application data replication in order to maintain blockchain security using the rules of the Proof_of_Work protocol. Our protocol reduces the replication of blocks that nodes must store and transform linearly into polylogarithmically. Exponential improvement without loss of data, as a minimum number of data copies must be maintained. It eliminates transactions from blocks of great depth and preserves consensus information (block headers). This contribution is presented in Chapter 4 and published in [79].

There are two kinds of data in these data: firstly, the application data, which includes transactions, account balances, and the evolution of the state of smart contracts [26, 140], as well as anything included in the data of the blocks themselves. Secondly, the consensus data consists of information that makes up the block header, including the proof of work [47] (or proof of stake) and the nonces needed to discover it. The consensus data allows us to determine the longest chain among numerous forks and make consensus possible. The number of block headers that must be stored and sent to new startup nodes in Bitcoin increases at a constant rate of one block header every ten minutes [41], with size compared to that, for example, while items can be added or removed from the UTXO [25]. Similarly, in Ethereum, the addition or removal of smart contracts [25] always causes block headers to increase at a constant rate of one block header per 12.5 seconds [73].

Résumé :

1.8 Failles des systèmes centralisés

La sécurité est un élément fondamental des sociétés, entreprises et organisations. Les consommateurs de technologies de l'information (Internet, informatique, stockage, etc...) exigent un niveau élevé de sécurité pour veiller à la bonne gestion des données numériques, telles que les informations de santé, les identités digitales et l'accès à Internet. Malheureusement, de nombreuses attaques, telles que l'usurpation d'identité, le vol d'argent et le vol de données sont fréquemment commises. Pour faire face à ce problème, les gouvernements, les banques et les entreprises technologiques mettent en place des systèmes et des normes de sécurité visant à protéger les informations sensibles des utilisateurs. L'ANSSI [5] a pour mission principale la sécurisation des données sensibles de l'État, offrant son expertise et son assistance technique aux collectivités et entreprises. Les banques s'assurent que les clients dépensent auprès des bonnes personnes et des montants appropriés, et leurs avocats vérifient que les produits ne sont pas copiés ou distribués sans autorisation.

Il est nécessaire d'introduire de nouveaux cadres de sécurité des données, qui peuvent être basés soit sur la centralisation des données vers un tiers de confiance, soit sur la décentralisation des données vers plusieurs parties. Les systèmes d'information les plus récents sont complexes, car ils combinent des fonctions, des bibliothèques tierces et un développement continu. Pour faire face à cette complexité grandissante, il est essentiel d'optimiser les solutions de sécurité et d'analyse. Les erreurs les plus mineures peuvent avoir de graves conséquences, et l'automatisation de ces fonctions peut permettre une analyse approfondie et concentrer les connaissances des ingénieurs sur les parties vulnérables du code. Les utilisateurs font confiance aux systèmes centralisés, s'identifient sur les plateformes numériques où ils envoient et reçoivent des données, et stockent des informations dans le cloud.

La centralisation des données pourrait certainement unifier les efforts de sécurité des données dans un système robuste, mais cette solution centralisée présente de nombreux défis.

- Un tiers de confiance représente un point vulnérable aux piratages par lequel transitent des données sensibles. La concentration des données dans un seul système constitue une opportunité pour les pirates, car elle leur permet d'accéder à un grand nombre de données en une seule fois.

- Risque de corruption : Les systèmes centralisés sont souvent gérés par une seule entité, ce qui signifie qu'il n'y a aucun mécanisme de contrôle pour empêcher une personne de corrompre le système.

- Concentration des données : La centralisation des données signifie qu'elles sont stockées sur des serveurs qui peuvent être vulnérables à des attaques informatiques et à des accès non autorisés.
- Faible fiabilité : Les systèmes centralisés sont souvent sujets à des pannes et à des temps d'arrêt plus longs que ceux des systèmes distribués, ce qui peut entraîner des retards et des erreurs.
- Faible scalabilité : Les systèmes centralisés sont souvent limités en termes de capacité et de fonctionnalités, ce qui peut entraîner des problèmes de scalabilité et de performance.
- Coûts élevés : Les systèmes centralisés sont souvent plus chers que les systèmes distribués, car ils nécessitent des investissements pour le matériel, les logiciels et le personnel.

La collecte massive de données personnelles constitue une nouvelle menace pour la vie privée. En effet, nous confions de plus en plus de données personnelles à des entreprises en échange d'un accès gratuit aux réseaux sociaux, aux messageries ou à d'autres services. Les géants de la technologie tels que Google, Amazon, Apple, Facebook ou Microsoft (GAFAM) contrôlent les données personnelles de millions d'utilisateurs, ce qui pose de sérieux défis en matière de sécurité des données. Ils ont accumulé une grande quantité d'informations en peu de temps, qu'ils utilisent avec des algorithmes d'intelligence artificielle pour améliorer leurs produits, mais aussi pour surveiller et influencer leurs utilisateurs. Les scandales récents tels que Cambridge Analytica ou les révélations d'Edward Snowden ont mis en lumière les dangers de cette accumulation de données personnelles entre acteurs sur lesquels nous n'avons que peu ou pas de contrôle. Ainsi, la protection de la vie privée est revenue au centre du débat politique. En réponse à ces attentes, l'Union européenne a ouvert la voie à une plus grande réglementation du commerce des données avec la promulgation du RGPD en 2018, et le Brésil, le Japon et la Californie ont emboîté le pas. Nous sommes à un tournant : les consommateurs-citoyens exigent une meilleure réglementation de leur vie privée et des garanties de sécurité pour leurs données.

1.8.1 Limite des infrastructures à clé publique

La sécurité des échanges numériques est importante pour les utilisateurs qui effectuent des transactions numériques. Une infrastructure à clé publique (ou Public key infrastructure (PKI)) émet des certificats numériques pour exécuter des fonctions cryptographiques. Ces derniers sont utilisés pour vérifier et authentifier les qualifications des différentes parties impliquées dans l'échange électronique d'informations.

Une PKI contient plusieurs composantes, l'autorité de certification (CA) qui valide et signe les demandes de certificats. La CA (ou certificate authority (CA)) [121] est un exemple d'autorité centralisée qui garantit l'identité des utilisateurs. Elle est l'une des principales composantes de la PKI et l'élément décisif et de confiance du processus de certification, étant responsable des listes de révocation. L'autorité d'enregistrement (ou RA) est l'interface entre l'utilisateur et l'autorité de certification. Il est responsable de l'identification des propriétaires du certificat et s'assure du respect des restrictions liées à l'utilisation du certificat. L'autorité de dépôt conserve les certificats numériques et centralise et organise le dépôt de ceux-ci.

La cryptographie à clé publique: Le chiffrement à clé publique permet aux utilisateurs de s'authentifier pour établir la confiance entre les utilisateurs et les systèmes informatiques. Le chiffrement à clé publique repose sur la capacité des entités à authentifier les clés publiques d'autres entités. Par exemple, supposons qu'un utilisateur souhaite se connecter à son compte bancaire à l'aide d'un navigateur Web. La session en ligne est protégée par la clé publique de la banque. Si le navigateur de l'utilisateur accepte la mauvaise clé publique pour la banque, un attaquant peut intercepter la connexion et voler les identifiant de l'utilisateur.

—*Les CAs singent des certificats qui décrivent les identités numériques et fournissent un moyen de vérifier l'authenticité de ces certificats (c'est-à-dire que les services de l'autorité de certification sont le plus souvent utilisés pour protéger les communications numériques).*

Plusieurs CAs ont été compromis comme Comodo en 2011 [43], et de nombreux certificats frauduleux ont été délivrés et utilisé pour les attaques de l'homme du milieu [105, 90]. Outre les problèmes de cryptographie [93, 12, 11], si la CA est malhonnête ou compromise, elle peut délivrer des certificats prouvant l'authenticité des fausses clés publiques, ces clés peuvent être générées par un attaquant ou par l'CA elle-même.

— **Les PKI basées sur les journaux:** Plusieurs solutions intéressantes ont été proposées pour répondre ces problème. Les améliorations de Log-based PKI (ou PKI basées sur les journaux) permettent la création de journaux publics pour suivre les activités de la AC, garantissant ainsi la transparence. Dans cette approche, qui a été principalement développée pour résoudre le problème des autorités de certification peu performantes, les certificats ne sont pas considérés comme valides tant qu'ils ne sont pas ajoutés aux journaux publics en ajout seulement (append-only public logs). Certificate Transparency (CT) [92] a été la première solution basée sur les journaux fournie par Google pour améliorer la responsabilité des fonctions CA et la détection des certificats mal émis. Tout le monde peut vérifier l'activité de l'autorité de certification en surveillant les journaux CT et en identifiant les certificats suspects qui ne figurent pas dans les journaux publics. CT utilise le

protocole Gossip pour s'assurer que les autorités de certification laissent des preuves persistantes de tous les certificats qu'ils délivrent. Ainsi, les activités d'une CA sont visibles ("transparents") pour ses utilisateurs et pour les observateurs. Les utilisateurs n'accepter un certificat que s'il est accompagné d'une preuve qu'il est inclus dans le journal. Cependant, CT ne définit pas de mécanisme de révocation des certificats enregistrés. Par la suite, des extensions comme [18] et [126] sont proposées pour résoudre le problème de révocation en envoyant périodiquement des listes de révocation aux navigateurs et empêchent les autorités de certification de délivrer des certificats de clé publique à un domaine sans être visible par le propriétaire du domaine. [103] et [23] assurent la transparence du système de messagerie.

De telles extensions nécessitent généralement d'autres entités (tiers de confiance) en plus de l'exigence de protocole public [145, 84, 91]. Les PKI basées sur les journaux présentent plusieurs inconvénients :

- nécessitent une source d'informations centralisée et cohérente pour fonctionner en toute sécurité (tiers de confiance).
- n'incitent pas suffisamment à auditer le comportement de l'CA.
- nécessitent du temps et des efforts manuels pour signaler un mauvais comportement de l'CA.

Le rôle central de l'autorité de certification dans l'infrastructure à clé publique traditionnelle la rend fragile et sujette aux compromis et aux défaillances opérationnelles. La maintenance des autorités de certification et des listes de révocation est importante, en particulier dans les grands systèmes faiblement connectés. Les modèles PKI basés sur les journaux ne résolvent pas le problème de manière efficace et nécessitent l'utilisation de CA.

1.9 La solution apportée par la technologie blockchain

La blockchain fournit une solution décentralisée qui peut être utilisée pour éliminer les principaux points de défaillance en supprimant les CAs et les certificats numériques, et à la place tout est enregistré sur la blockchain. Cette technologie marque une nouvelle ère dans la façon dont la confiance est établie entre deux individus, c'est un système entièrement décentralisé, où la confiance est créée en partageant et en validant des données entre les nœuds du réseau, en utilisant d'anciennes méthodes connues, telles que les algorithmes cryptographiques et les réseaux pair à pair. Les nœuds sont identifiés par des alias, des clés publiques ou des adresses numériques.

1.9.1 La blockchain

Une blockchain est un registre distribué composé de participants qui communiquent entre eux en échangeant des messages et en appliquant un consensus pour parvenir à un accord commun sans tiers de confiance. Les participants doivent enregistrer les mêmes informations dans l'ordre chronologique dans leur système local. La Figure 1.4 montre la structure de données des blockchains qui forme une séquence de blocs de transactions liés entre eux par le hachage au bloc précédent, formant ainsi une chaine de blocs.

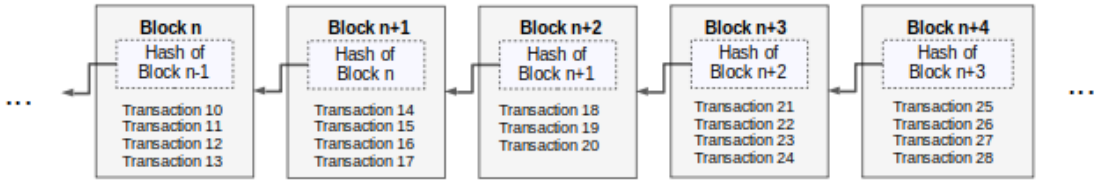


Figure 1.4: La structure de Blockchain.

Le concept de consensus distribué permet d'enregistrer des événements dans le monde numérique de manière incorruptible, afin que personne ne puisse contester leurs occurrences, puisque ces événements sont contrôlés par de multiples acteurs indépendants.

En 2008, la première crypto-monnaie appelée Bitcoin a été introduite dans l'article de Nakamoto [108]. Le terme crypto-monnaie fait référence à une monnaie numérique qui fonctionne correctement avec les méthodes de cryptographie, d'où le nom de crypto-monnaie (crypto : pour la cryptographie, monnaie : pour les transactions monétaires). Bitcoin permet aux utilisateurs de s'envoyer de fond sans la supervision d'une autorité centrale en mettant en place un consensus de Proof_of_Work pour valider ces échanges, résolvant le problème de la double dépense.

Le réseau de Bitcoin consiste en des nœuds de validation qui tentent de valider un bloc de transactions en résolvant un problème informatique complexe (ou une énigme). Ces nœuds sont appelés mineurs. Ce processus est compétitif et le premier mineur à réussir à résoudre l'énigme sera récompensé par une certaine quantité de crypto-monnaie pour ses efforts. Bitcoin a adopté la technologie blockchain dans son architecture de protocole : Premièrement, la vérification des transactions financières sans tiers de confiance, basée sur des méthodes cryptographiques, dans une distribution totale gérée par des participants du réseau interconnectés par un réseau pair à pair. Les nœuds de validation vérifient l'ensemble du réseau pour s'assurer que : (i) le participant posséde bien la somme qu'il souhaite envoyer, (ii) il ne l'a encore envoyée à personne d'autre. Les transactions sont ensuite ajoutées dans des

blocs par les mineurs. Ils s'affrontent et calculent le hachage de tout le bloc pour trouver la valeur correcte qui tient compte de la difficulté fixée par consensus. Enfin, le premier mineur à calculer le hachage correct d'un bloc en dessous du seuil fixé par le consensus partagera son bloc avec le reste du réseau. Les nœuds vérifient le bloc reçu et l'ajoutent à leur chaîne. Le bloc est inclus dans la chaîne de tous les pairs.

Un consensus est un ensemble de règles qui permettent aux nœuds connectés sur un réseau pair à pair de s'entendre sur une finalité et d'assurer la cohérence d'un état unique.

Les informations sur la plupart des blockchains publiques sont visibles par tous les participants hébergeant la blockchain. En raison de l'immuabilité des enregistrements, de la fiabilité des informations basées sur le consensus et de la transparence des transactions fournies par la blockchain, cette technologie a été intégrée dans des industries sensibles et actives comme les smart cities, la revue [123] fournit une littérature systématique sur des cas d'utilisation spécifiques de la blockchain et analyse la conception et le prototypage de systèmes de blockchain pour un avenir urbain durable et intelligent, dans l'authentification dans l'Internet des objets [124], elle intéresse le secteur medical [72] pour le suivi thérapeutique des patients, ainsi que pour la gestion et l'analyse en toute sécurité de données de santé volumineuses [46].

Nous mentionnons les principaux éléments que les blockchains utilisent pour être robustes et fiables:

- Blockchain offre une gestion des journaux entièrement distribuée à l'aide d'un réseau pair à pair, un modèle construit de manière décentralisée, de sorte que la communication ou l'échange qui s'y déroule se fait entre des nœuds du système qui partagent la même responsabilité.

- Les journaux de transactions sont inclus dans une suite de blocs, où chaque bloc contient un hachage unidirectionnel sécurisé du bloc précédent. Un nouveau bloc ne peut être ajouté à la chaîne que si une décision par consensus a été prise entre les pairs du réseau.

- Les empreintes digitales des blocs candidats jouent un rôle important dans le réseau blockchain. avec leur aide, il est possible de vérifier les données et d'assurer leur intégrité; ils peuvent démontrer l'existence d'un fichier de données particulier à un moment donné. L'ajout d'une empreinte digitale à la blockchain garantit que les tentatives de manipulation sont exposées, car chaque modification du fichier de données entraîne une valeur de hachage complètement différente

1.9.2 Les règles du protocole Proof_of_Work

Dans les réseaux blockchain, les transactions et le blocs sont propagés sur le réseau P2P en utilisant le protocole Gossip, comme décrit dans [39]. Pour chaque nouveau bloc reçu et validé, un noeud l'annonce aux pairs, qui demanderont ce bloc s'il souhaitent étendre leur blockchain locale. Le processus de propagation via gossip continue jusqu'à ce que chaque nœud du réseau ait ce bloc. En raison de la nature publique et sans permission de la blockchain et de pseudonymat, plusieurs attaquants Sybil peuvent obtenir de nouvelles identités ou comptes avec peu d'effort. Cependant, la puissance de hachage de mécanisme de Proof_of_Work provient d'un investissement matériel réel conçu pour atténuer les attaques Sybil et ne peut pas être facilement falsifiée.

La règle de la chaîne la plus longue signifie que la chaîne la plus longue établie peut servir de référence commune à l'historique du réseau. La chaine la plus longue est considérée comme la chaine la plus sécurisée.

Le protocol Proof_of_Work améliore l'exigence de la terminaison avec la spécification de **finalité probabiliste** : Pour tout nœud honnête, chaque nouveau bloc est soit rejeté soit accepté dans sa blockchain. Un bloc reçu peut toujours être rejeté mais avec une probabilité décroissante exponentielle à mesure que la blockchain continue de croître.

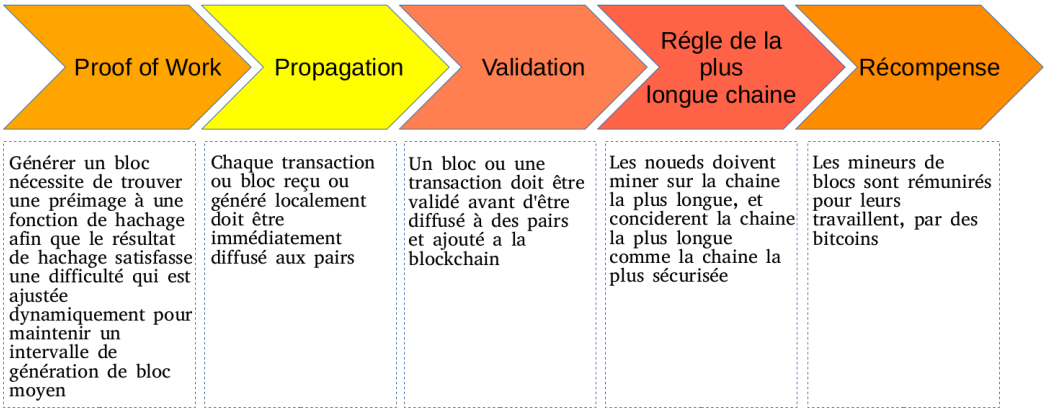


Figure 1.5: Le consensus de Proof_of_Work.

LE consensus Proof_of_Work peut être résumé par les règles suivantes:

- Proof_of_Work: la génération de bloc nécessite de trouver une préimage à une fonction de hachage afin que le résultat de hachage satisfasse une difficulté qui est ajustée dynamiquement pour maintenir un intervalle de génération de bloc moyen, regadrer la Subsection 2.5.1 pour plus de détails.

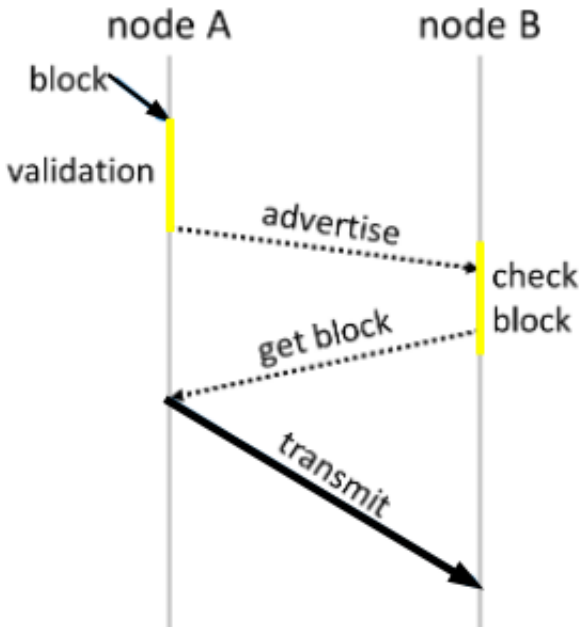


Figure 1.6: Le protocole de propagation de bloc dans la blockchain.

- Gossip: La blockchain utilise le réseau pair à pair, et toute transaction ou bloc est propagé à travers le réseau en utilisant le protocole gossip. Un nœud relais qui reçoit un bloc, envoie d'abord l'en-tête du bloc à ses voisins, ces derniers, s'ils n'ont pas encore reçu le bloc, réclament le bloc avec un message *get block*, voir la Figure 1.6. Un mineur qui vient de miner un nouveau bloc sait qu'aucun autre nœud ne possède le bloc, il envoit dans un premier message le bloc entier a ces voisins.

- Validation de bloc: Chaque fois qu'un bloc est envoyé et reçu dans le réseau, les nœuds validateurs vérifient que le bloc a respecté les règles du Proof_of_Work, que les transactions incluses n'ont pas été dépensées dans le passé (*Double dépenses*, voir la supsection 2.8), et que chaque émetteur de transaction possède les bitcoins qu'ils utilisent.

- La chaine la plus longue: Les mineurs sont incités à miner sur le dernier bloc de la chaîne la plus longue, car elle est considérée comme la chaîne la plus sécurisée. Il y a plus de blocs minés, et donc plus de puissance de calcul fournie.

- Frais de minage: Les mineurs gagnent des bitcoins, cette incitation encourage les mineurs à suivre les règles du protocole et à concourir pour valider de nouveaux blocs, et participer à la maintenance de la blockchain.

1.9.3 Certification

la création de PKI décentralisée basée sur la blockchain supprime les points de défaillance potentiels de l'utilisation d'autorités de certification qui peuvent compromettre des certificats entières [49].

Les alternatives aux propositions de PKI décentralisée incluent le Web of Trust (WoT). WoT offre un modèle de confiance décentralisé où l'authenticité des clés publiques et de leurs propriétaires est basée sur le degré de confiance que d'autres entités WoT ont dans cette entité. [13, 118, 54]. et garantit la transparence des certificats, la révocation et des enregistrements de transaction fiables. Il supprime le besoin d'une autorité de certification, mais présente des lacunes en matière de non-répudiation.

D'autres approches de PKI basée sur blockchain ont été proposées qui utilisent la blockchain pour enregistrer des données qui seraient signées et vérifiées par une autorité de certification, garantissant ainsi la non répudiation et l'immuabilité de ces données [135, 90, 143, 101]. Elles offrent des fonctionnalités très limitées et la question des tiers de confiance n'a pas été résolue. Ces propositions sont présentées dans la Section 3.2.

1.9.4 Utilisation de la blockchain pour une messagerie sécurisée

Dans la première partie de cette thèse, nous proposons une vérification d'identité par la blockchain pour une communication sécurisée sans autorité centrale afin que de nouvelles clés puissent être enregistrées ou révoquées en toute sécurité, sur la base d'un mécanisme de consensus entre des nœuds de confiance faisant partie du système. Une PKI qui utilise la blockchain comme registre en ajout seulement et garde l'autorité de certification telle que proposée dans [135], supprime les principaux avantages de l'utilisation d'une blockchain, puisque l'enregistrement et la révocation doivent toujours être effectués par une CA traditionnelle. Par conséquent, nous proposons une méthode entièrement décentralisée où les décisions d'enregistrement et de révocation des clés sont prises par les nœuds de la blockchain et non par l'autorité de certification. Nous utilisons un smart contract pour vérifier l'identité de l'utilisateur et les clés publiques. Le smart contract peut confirmer l'identité de l'utilisateur et fournir la confiance entre les utilisateurs pour échanger des messages. La communication pair à pair est basée sur la blockchain, et toutes les autres communications sont considérées comme malveillantes. Un smart contract est un code stocké et exécuté sur la blockchain. L'utilisateur peut générer : 1) une clé publique à l'aide de l'algorithme ECDSA, 2) une deuxième clé publique et 3) deux clés privées et déduire l'identité sous forme de hachage de la clé publique. La première paire de clés est utilisée pour les authentifications et la seconde paire de clés est utilisée

pour révoquer ou mettre à jour la première clé publique. Un utilisateur peut stocker en toute sécurité la clé privée dans un système sécurisé en secret, et demander à enregistrer son identité sur la blockchain avec la clé publique correspondante. Cette contribution est publiée dans [80].

1.10 Limites de la décentralisation des blockchains

La décentralisation est le principe des blockchains. Cette décentralisation des échanges et du stockage des données et du réseau fait de cette technologie l'une des plus sécurisées. En fait, les blockchains sont considérées comme invulnérables ; ils s'appuient sur chaque membre qui compose le réseau : les "nœuds". Ces éléments qui composent le réseau garantissent que les échanges sont enregistrés et que chaque hébergeur est partiellement ou totalement propriétaire des données de la blockchain.

L'utilisation de la blockchain pour les systèmes de certification et de communication décentralisés présente de grands avantages dans la protection des informations des utilisateurs sans point de contrôle centralisé. D'autre part, le problème majeur auquel notre proposition est confrontée est le stockage des identités et de leurs clés publiques associées sur la blockchain. Comme Satoshi le décrit dans [108], l'espace disque requis pour stocker des blocs en Bitcoin pendant un intervalle de temps de 10 minutes pour un bloc est $6 * 24 * 365 * tailledebloc$ par an. Au 29 septembre 2022, Bitcoin présente une taille dépassant les $400Go$. La Figure 1.7 représente la croissance linéaire de la taille de sa blockchain au cours des deux dernières années. Pour réduire le temps et la capacité nécessaire au stockage et à la vérification des clés, seul le hachage de notre certificat est stocké sur la blockchain.

Les deux principales blockchains, Bitcoin et Ethereum (première et deuxième générations de blockchain) continuent de faire face à des problèmes de croissance constante des données stockées sur leurs blockchains, la taille de la blockchain d'Ethereum dépasse les $900Go$. $65,3\%$ des nœuds de son réseau sont hébergés dans des services Cloud centralisés avec 51% de ces nœuds utilisent le service Cloud d'Amazon AWS, ces derniers ne sont pas répartis géographiquement (localisés principalement aux États-Unis), et rend la blockchain dangereusement centralisée. Les blocs de notre protocole sont similaires aux blocs Ethereum. La troisième génération de blockchain tente de corriger les lacunes des précédentes: elles résolvent le problème de scalabilité avec l'apparition de blockchains plus rapides à gérer. Parmi ces protocoles, on retrouve Mina [24], Solana [142], BitcoinCash, etc...

La blockchain Mina [24] développée par O(1) labs est considérée comme la plus petite blockchain avec une taille qui ne dépasse pas plus de $22Ko$. Il intègre des certificats zero-knowledge proofs ("succinct non-interactive argument of knowledge" ou "Zk-SNARKs") qui ne contiennent pas d'informations sur les opérations elles-

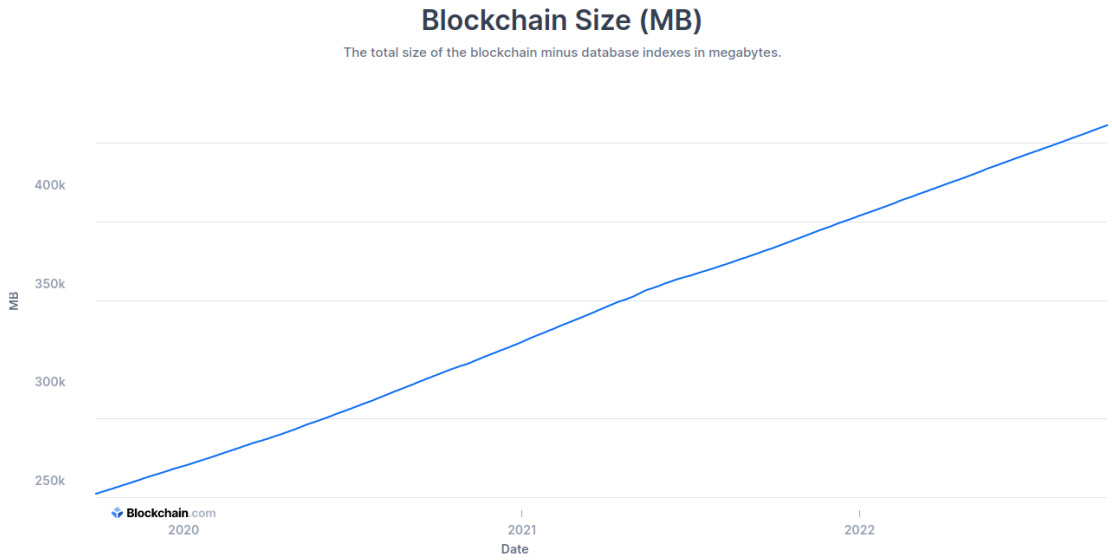


Figure 1.7: La taille totale de la blockchain en mégaoctets (Mo) pour Bitcoin de 2020 à 2022 [4].

mêmes, mais garantissent la confidentialité des données et l'anonymat des utilisateurs, pour valider les transactions. Ces preuves ont d'abord été utilisées par la crypto-monnaie Zcash [20].

Á chaque étape, Zk-SNARKS génère une preuve pour valider une nouvelle transaction sans consulter le registre de toutes les transactions précédentes avec une fonction récursive et construit une chaîne de preuves du premier bloc au dernier, permettant de construire un historique complet à partir de simples fichiers de quelques octets (de nouveaux certificats sont créés à partir d'anciens certificats.) ce qui réduit considérablement la taille des données que chaque utilisateur et validateur doit télécharger. Malgré la compression de chaque Zk-SNARK atteignant $1Ko$, la taille du registre devrait de toute façon augmenter avec le temps.

De même, Solana [142] et Polygon sont conçues pour executer les transactions en quelques millisecondes avec des performances et des coûts très faibles. Solana est une blockchain à haut débit qui utilise la technique d'horodatage Proof_of_History pour atteindre des temps de production de bloc inférieurs à la seconde.

1.10.1 Limites de la décentralisation des Blockchains: historique des transations

La décentralisation des blockchains est limitée par la taille de l'historique des transactions. Il est impossible pour une blockchain décentralisée d'avoir une base de données contenant un historique des transactions qui s'étende sur des années, car elle serait trop volumineuse pour être stockée et synchronisée sur un réseau décentralisé.

La taille de l'historique des transactions est un facteur clé qui limite la faisabilité de la décentralisation des blockchains. De plus, la longueur de l'historique des transactions nécessite une puissance de calcul importante pour s'assurer que toutes les transactions sont valides et conformes aux règles de consensus. Cela peut devenir très coûteux à mesure que l'historique des transactions s'allonge, ce qui limite l'utilisation de blockchains décentralisées pour des applications qui nécessitent une grande quantité de transactions et une longue période de temps.

La difficulté d'évolutivité provient des données qui doivent être stockées et envoyées sur le réseau lorsque les nœuds de la blockchain se synchronisent pour la finalité de consensus ou pour rejoindre le réseau pour la première fois. Plusieurs nouvelles blockchains sont apparues qui permettent de scaler.

Dans le paradigme standard de la blockchain, les transactions sont entièrement répliquées sur chaque noeud validateur du réseau. Chaque nœud validateur doit stocker l'intégralité de la blockchain à partir du premier bloc.

Le défi majeur de la conception de ce systeme avec la blockchain est:

- Limites sur le stockage des données : Les enregistrements de transaction de la blockchain sont généralement de l'ordre du kilo-octet et les blocs ne contiennent pas beaucoup de données. En outre, la structure de donnée de la blockchain implique que tous les changements d'état sont enregistrés dans la blockchain. Tous les nœuds participant au réseau doivent conserver une copie complète de la blockchain. En septembre 2022, les nœuds Bitcoin [4] nécessitent plus de $400Go$ [3] d'espace de stockage pour rester synchronisés avec le réseau.

- Nombre de transactions par seconde: Le nombre total de transactions par bloc est limité par la taille du bloc. Le bloc Bitcoin est limité a la taille de $1Mo$, et compte tenu de la taille moyenne des transactions de près de 540 octets, Bitcoin traite actuellement environ 1950 transactions par bloc, dans un intervalle de 10 minutes [68].

Bien qu'Ethereum permette la mise en œuvre des smarts contracts, les grandes quantités de données sont stockées dans des systèmes sécurisés en dehors de la blockchain, et le coût de lecture et de vérification de ces données est extrêmement élevé [6].

Alors que les validateurs dans Mina n'ont besoin que d'une petite quantité de données, les *bloc producers* chargés de créer de nouveaux blocs stockent toute la blockchain pour parvenir à un consensus. Mina a donc toujours un problème de croissance de stockage. Ajoutant a ça le probleme de décentralisation puisque l'historique des transactions est stocké uniquement dans les *bloc producers*. Cette blockchain peut avoir des milliers de validateurs, mais comme y'a seulement une

poignée d'entre eux qui peuvent accèder à l'historique des transactions, il est difficile de dire que cette blockchain est décentralisée, car si quelque chose arrive à cette poignée de noeuds qui stockent l'historique des transactions, il pourrait devenir possible de manipuler sérieusement les transactions sur cette blockchain.

Solana compte plus de 1700 validateurs, mais toutes les transactions sont traitées par un groupe beaucoup plus petit de 150 validateurs, de plus, cette blockchain utilise les services de Cloud computing Hetzner et OVHCloud pour stocker sa blockchain et cela centralise le processus. Contrairement au développeurs de Polygon qui sont plus soucieux da la décentralisation de la blockchain, elle utilise Arweave, un systeme de stockage décentralisé basé sur la blockchain.

Algorand [58], dont la blockchain compte 1600 "nœuds participants" en consensus, ces transactions sont gérées par un groupe plus restreint de seulement 120 "relay nodes", dont la plupart sont gérés par des entités derrière Algorand et ses filiales.

En conlusion, les blockchains qui promettent la scalabilité, que ce soit le nombre de transactions par seconde, ou l'utilisation de memoire minimale pour valider des transactions, pouvant porter atteinte la décentralisation de la blockchain, en créant des petits committés qui controlent l'historique de ces transactions, comme mentionné ci-dessus. Les blockchains de troisième génération offrent une évolutivité au détriment de la décentralisation.

Vitalik Buterin (le fondateur d'Ethereum) a développé le concept du triangle du Trilemme des blockchains. Ce dernier part du principe qu'il n'est pas possible d'optimiser ces trois principes en même temps : sécurité, évolutivité (scalabilité) et décentralisation.

1.11 Solutions pour une décentralisation de l'historique d'une blockchain

1.11.1 Solutions existantes

La décentralisation a été un sujet clé de la technologie blockchain depuis que le premier bloc Bitcoin a été miné en 2009. Lorsqu'un réseau tend à être centralisé, il est exposé à un risque accru. L'objectif est de porter une attention particulière à la distribution des nœuds et aux influences extérieures qui peuvent l'affecter. Le fait que la blockchain soit basée sur le travail simultané de plusieurs nœuds caractérise précisément sa sécurité. Lorsqu'une blockchain est décentralisée à travers toutes ces couches, elle est presque toujours très sécurisée car il n'y a pas de point de défaillance unique. Cela crée également une résistance à la censure car aucune autorité centrale ne peut contrôler l'actif numérique.

Les nœuds mineurs dans Bitcoin, et globalement dans toutes les blockchains

fonctionnant sous le consensus de Proof_of_Work, se regroupent pour maintenir leurs profits; ainsi, il est possible d'augmenter la puissance de calcul en partageant les coûts. Ces groupes, appelés pools de minage, fonctionnent selon leur propre protocole et communiquent avec le réseau de blockchain via un seul nœud. Leur popularité est si grande que les trois plus grands groupes représentent plus de 51 % de la puissance de calcul. Ainsi, le nombre d'unités nécessaires pour effectuer une attaque à 51 % est de trois, c'est le coefficient de Nakamoto, en assemblant Foundry USA Pool, Antpool et F2Pool qui détiennent ensemble 55% de la puissance de calcul totale. Plus le nombre est élevé, plus l'effort pour casser le réseau est important et vice versa ; plus le nombre est faible, plus le réseau est vulnérable.

Une proposition pour lutter contre lee pools de minage serait d'introduire des frais Sybil : le coût d'une personne utilisant plusieurs nœuds serait plus élevé que le coût d'utilisation d'un seul nœud. Cela encouragerait les agents du réseau à ne maintenir qu'un seul nœud, allouant ainsi la puissance de calcul plus efficacement. Jusqu'à présent, aucun moyen de mettre en œuvre cette solution n'a été trouvé.

À travers quelques blockchains qui ont été transparentes sur la manière exacte dont leurs historiques de transactions sont stockés, l'une des rares crypto-monnaies qui font cette distinction est le Bitcoin, dont les nœuds validateurs ou full nodes stockent son historique de transactions, il y a actuellement environ 15000 nodes Bitcoin autour du monde, ce qui en fait le plus décentralisé au niveau de la blockchain.

Arweave et Filecoin sont deux solutions de stockage de données et présentent des similitudes telles qu'être décentralisées et basées sur la technologie blockchain. Arweave se concentre sur le problème du stockage de données à long terme. Filecoin offre du stockage pour des applications Web. Au lieu de stocker des données dans un réseau centralisé , Filecoin crée un réseau décentralisé où les données sont répliquées sur plusieurs emplacements et accessibles de n'importe où. Le but principal de Filecoin est de créer un vaste réseau de noeuds pour stocker des données partout dans le monde, ces derniers sont incitées à stocker les données d'autres personnes. Cela permet aux données d'être plus proches des sources qui en ont besoin et n'offre également aucun point d'attaque central pour les attackants. Le but de Filecoin est qu'il deviendra un réseau de stockage fiable et peu coûteux par rapport a l'industrie du stockage en Cloud, dominée par des sociétés massives telles qu'Amazon Web Services (AWS) et Google.

Arweave propose de stocker les données de manière fiable et permanente. Alors que ce projet entreprenait de résoudre le problème de stockage de données à long terme, ils ont apporté une innovation à la technologie actuelle de la blockchain et ont développé un système incitatif unique pour que les noeuds stockent les données de manière permanente. Cela rend les données à la fois immuables et impossibles à supprimer. Si des copies de fichiers sont répliquées d'innombrables fois sur des

serveurs partout dans le monde, il est pratiquement impossible de modifier un fichier qui a été téléchargé sur Arweave.

Arweave a entrepris de résoudre le problème du stockage de données à long terme sur Internet. Il a utilisé la technologie blockchain pour stocker les données et a mis en place un système incitatif pour que les pairs stockent les données de manière permanente. Arweave a construit un blockweave au lieu d'une blockchain. Le blockweave est maintenu par de nombreux liens à l'intérieur de l'ensemble du stockage de données. Par conséquent, la seule façon d'ajouter de nouvelles données au blockweave est qu'un pair ajoute un bloc sélectionné au hasard déjà sur le blockweave. Seuls les pairs qui peuvent ajouter un bloc aléatoire de l'historique sont autorisés à stocker de nouvelles données [138]. La seule façon d'ajouter de nouvelles données avec un nœud Arweave est si ce nœud a précédemment ajouté un fichier (ou un groupe de fichiers) à sa mémoire et sélectionne aléatoirement un bloc déjà dans sa chaîne pour ajouter un nouveau bloc, les mineurs doivent se référer aux données précédemment stockées prouvant qu'ils les possèdent toujours. Arweave utilise Proof_of_Work et respecte également la règle de la chaîne la plus longue. En plus d'inclure un bloc de l'historique dans le nouveau bloc suivant, il inclut également un lien au dernier bloc de la chaîne. Les mineurs sont incités à stocker de grandes quantités de données pour augmenter leurs chances de trouver le bon bloc car la sélection des blocs est aléatoire. Cette amélioration du stockage des données et les récompenses économiques pour les mineurs qui conservent les données créent des conditions pour que les données soient stockées pendant de longues périodes. De plus, Arweave introduit un bloc de synchronisation généré une fois par bloc de 12, contenant une liste complète du solde de chaque portefeuille du système et un hachage de chaque bloc précédent sans la transaction. Les blocs de synchronisation aident les nouveaux participants du réseau à s'amorcer, et il n'est pas nécessaire de télécharger tous les blocs à partir du bloc de genesis.

La principale similitude entre Arweave et Filecoin est qu'ils sont tous deux des périphériques de stockage qui utilisent la technologie blockchain sur un système décentralisé. Ils sont également décentralisés, ce qui signifie qu'il n'y a pas d'une entité centrale contrôlant leurs serveurs. Au lieu de cela, les serveurs sont contrôlés par des mineurs du monde entier qui stockent les données moyennant des frais. Les mineurs sont liés par un ensemble de règles préétablies.

la difference avec Bitcoin, qui utilise la puissance de calcule de CPU pour miner des nouveaux blocs, Arweave utilise la puissance de calcul et la capacité de stockage. La probabilité qu'un noeud Arweave mine un nouveau bloc b est donnée par:

$$P(b) = P(\text{posséde le bloc de l'historique}) * P(\text{trouve le bon hachage de bloc})$$

Cela signifie que le stockage sur Arweave se fait sur de gros nœuds ; en d'autres

termes, il faut disposer de grandes mémoires de stockage pour pouvoir miner dans Arweave. Cela peut créer une centralisation autour des validateurs qui doivent être de très grands nœuds. Arweave incite les grands nœuds à continuer de stocker des données, et incite tous les participants a stocker tous les blocs historiques, ce qui est très inefficace. Dans 6, nous proposons une solution plus améliorée et le stockage ne dépendra pas des mémoires des nœuds, notre protocole permettra également aux petits nœuds de sauvegarder les blocs.

Depuis septembre 2022, la capacité de stockage du réseau Filecoin est supérieure à 18 EiB (exbibytes), et celle de Arweave est supérieure à 77,5 TB (téraoctets). Par conséquent, dans les Chapitres 4 et 5, nous présentons des améliorations aux protocoles de blockchain hautement sécurisés et décentralisés qui permettent de stocker des informations sur la blockchain, en optimisant la mémoire de stockage requise.

1.11.2 Les protocoles SECUSCA_*: méthodes de sharding de l'historique des transactions

Nous nous intéressons à la scalabilité de blockchain, qui est limitée par sa nature décentralisée et publique ; nous formulons une méthode qui peut résoudre la barrière d'évolutivité du réseau blockchain lorsque le nombre de nœuds et de transactions augmente, sans changer le consensus principal pour vérifier et valider de nouveaux enregistrements (financiers ou autres transactions) sur la blockchain et sans compromettre la sécurité. Nous avons proposé SecuSca, un mécanisme pour réduire la réplication des données d'application afin de maintenir la sécurité de la blockchain en utilisant les règles du protocole Proof_of_Work. Notre protocole réduit la réplication des blocs que les nœuds doivent stocker et transformer linéairement en polylogarithmique. Amélioration exponentielle sans perte de données, car un nombre minimum de copies de données doit être maintenu. Il supprime les transactions des blocs de grande profondeur et préserve les informations de consensus (en-têtes de bloc). Cette contribution est présentée dans le Chapitre 4 et publiée dans [79].

Il y a deux informations dans ces données : premièrement, les données d'application, qui comprennent les transactions, les soldes de compte et l'évolution de l'état des contrats intelligents [26, 140], ainsi que tout ce qui est inclus dans les données de bloc elles-mêmes. Deuxièmement, les données de consensus consistent en des informations qui constituent l'en-tête du bloc, y compris la preuve de travail [47] (ou preuve de participation) et les nonces nécessaires pour le découvrir. Les données de consensus nous permettent de déterminer la chaîne la plus longue parmi de nombreuses fourches et de rendre le consensus possible. Le nombre d'en-têtes de blocs qui doivent être stockés et envoyés aux nouveaux nœuds de démarrage dans Bit-

coin augmente à un taux constant d'un en-tête de bloc toutes les dix minutes [41], avec une taille par rapport à celle, par exemple, tandis que des éléments peuvent être ajoutés ou supprimés de l'UTXO [25]. De même, dans Ethereum, l'ajout ou la suppression de contrats intelligents [25] fait toujours que les en-têtes de blocs augmentent à un taux constant d'un en-tête de bloc par 12,5 secondes [73].

1.12 Description des travaux

Ce manuscrit est organisé en trois parties. **Introduction et contexte**, **protocole de communication sécurisée avec l'utilisation de la blockchain**, et **protocole de sharding des transactions de la blockchain**, et 6 chapitres.

La thèse décrite dans ce manuscrit porte sur l'utilisation de la blockchain pour décentraliser les méthodes de communication sécurisées et l'historique des événements sur la blockchain.

- Partie I.., cette partie contient les chapitres suivants:
  - dans la Chapitre 1 nous exposons et introduisons notre problèmatique pour réaliser des systèmes décentralisés sur la blockchain, premièrement, pour créer un système de messagerie sécurisé utilisant une infrastructure à clé publique sans faire confiance à l'autorité de certification, et deuxièmement, nous concevons un protocole de sharding pour l'historique des transactions et des blocs dans la blockchain sur les nœuds du réseau, tout en respectant les trois principes de la blockchain que sont la décentralisation, la scalabilité et la sécurité, en utilisant le consensus Proof_of_Work.
  - dans le Chapitre 2 nous présentons un Background de la blockchain et les principes de consensus.
- Partie II. ., le Chapitre 3 propose l'utilisation de la blockchain pour l'identification numérique des utilisateur et permettre une communication sécurisée, une solution qui permet à chaque utilisateur de garder le contrôle sur ses données, en évitant leur centralisation par toute grande entreprise. La validation se fait par des smarts contracts. Traditionnellement, l'association de la clé publique et de l'identité est vérifiée par l'autorité de certification qui signe les certificats. la blockchain élimine de nombreuses failles de sécurité en raison de sa nature décentralisée, une fois que deux utilisateurs sont authentifiés par la blockchain, les utilisateurs peuvent communiquer de manière équitable et sécurisée. Ceci vise à donner aux utilisateurs le contrôle et la propriété de leurs données et de leur identité. Ces avantages nous ont motivés à utiliser les blockchains pour construire un nouveau système PKI décentralisé.

- Partie III.., présente notre protocole de sharding qui encourage les nœuds à stocker des parties de la blockchain, contrairement aux blockchains qui utilisent le modèle UTXO, et qui stockent toutes les transactions sur des nœuds complets, exemple Bitcoin, notre protocole réduit la réplication de blocs sur tout le réseau de la blockchain, tout en gardant un minimum de répliques d'une donnée pour assurer sa disponibilité. Cete partie est constituée des chapitres suivants:

  - dans le Chapitre 4 nous présentons un état de l'art des solutions de scalabilité pour les blockchains, et nous décrivons notre premier protocole de sharding, *SECUSCA_1*. La blockchain sera répartie sur un sous ensemble de noeuds.
  - dans le chapitre 5, nous proposons une amélioration de la proposition précédente dans le choix des nœuds qui stockent les blocs de la blockchain. Les nœuds sont sélectionnés en fonction de leur capacité de stockage restante, l'objectif étant d'éviter la saturation des nœuds de petite capacité dans *SECUSCA_1*, et donc d'améliorer la décentralisation de la blockchain, tout en permettant la sélection de petits nœuds. La nouvelle proposition s'appelle *SECUSCA_2*.

- Partie IV.., le Chapitre 6 conclut ces travaux de thèse et présente quelques travaux en cours de réalisation et perspectives d'utilisation de nos différents protocoles.

Chapter 2

Background

In this chapter, we give a background on blockchain, how it works, and consensus operates in the distributed system to achieve consistency.

Introduction

Trust is the cornerstone of society, business, and the government. People are becoming more discerning and demanding privacy regarding managing their data, such as health data, identities, or managing financial accounts. Still, they continue to trust insurers, banks, and other centralized systems that promise data confidentiality. Banks ensure that we transact with the appropriate parties and for the right amounts. Lawyers control that products are not copied or illegally distributed. Trust in computer science is a concept borrowed from human society. For example, Golbeck [59] presented trust as a commitment to believe in the proper conduct of the future actions of another entity. Likewise, [62] defined trust as the quantified belief by a trustor (The entity that sets up a trust) concerning a trustee's competence, honesty, security, and dependability within a specific context. Trust-based systems [97, 62] involve the feeling of being safe to use, which develops a security policy that allows verified users to access the computer system through cryptography and provides security at different levels. Most rely on public key certificates, which are digital certificates issued by a trusted third party, a certificate authority that certifies the identity of the owner of a public key. PGP and X.509 specify the format for the most used certificates. The certificate authority was one of many concepts of trusted computing. Any cryptography service must support the relevant concepts of users' security policies, credentials, and trust relationships.

A reputation-based trust system uses the other's experiences and establishes a reputation measure to trust another entity. The outcome of a trust decision depends on various factors, including the trustor's inclination to believe and its impressions of and prior interactions with the trustee. Applications requiring a trust specification include content selection for medical systems [132, 127, 17], mobile computing [111, 137, 67], electronic commerce [125, 78], as well as internet of things (IoT) [120], to track patients for therapeutic monitoring in the health. However, using intermediaries is generally time-consuming, costly, and security-risky in the age of hackers. It is also incredibly complicated.

A new era is needed because people's information is still controlled, and there is a lack of privacy. Blockchain brings change by carrying the torch of a decentralized system. Blockchain basically operates as a public database where everyone can keep

tabs on who owns how many cryptocurrencies or other digital assets. The world economy has seen the emergence of Bitcoin [108], a virtual and stateless currency, as an unheard-of transnational monetary system that operates independently of any central banking institution. Bitcoin was developed in 2009 by the enigmatic figure known only by the pseudonym *Satoshi Nakamoto*. It has aroused the interest of commentators, regulators, and economic theory specialists. It is a non-fiat; a user establishes the value and sends and receives transactions from other users using a computer and internet network. The revolution of Bitcoin is that the operations are distributed and not issued by any government or central bank. Its existence depends on the workings of a distributed IT system, which forms a vast Peer_to_Peer network connecting programs. i.e., Bitcoin clients implementing the Bitcoin protocol on individual computers worldwide. Once the program runs on the computer, the related information is saved on a digital file that sits in a distributed ledger using electronic signatures, enabling Bitcoin to perform the traditional functions of money. The history of all previous Bitcoin transactions, including the creation of new bitcoin units, is stored on the Blockchain. A series of blocks make up the Blockchain. Each block relates to its previous block and is maintained by a decentralized network after all the records have been approved by consensus. The average time between two Bitcoin blocks is 10 minutes. The first block #0 was created in 2009, and at the time of this writing, block #740362 has been added as the newest block to the chain.

The Blockchain is a public distributed ledger maintained by nodes worldwide which removes the requirement for a trusted third party and the associated costs for such institutions (banks, payment companies, credit card companies). The Blockchain holds a sequential record of all transactions occurring within a network. Bitcoin is the world's first cryptocurrency and first public Blockchain network.

Blockchain technology has numerous advantages in comparison to traditional centralized systems. It has earned its stripes regarding data integrity, security, and immutability.

- Decentralized control (no single party has an a priori privileged role).
- Consensus on a single state (single source of truth).
- Tamper-proof recording (validated transactions cannot be deleted or updated).
- Privacy preservation (publicly shared data but secured and privatized by cryptographic techniques such as cryptographic hashing and private-public key cryptography).

The blockchain system ensures data availability and transparency for all participating members by maintaining all historical and current transactions at each

node for blockchain verification. The main abstraction of reading blockchain requires the user to maintain a full node to run the consensus protocol and maintain a local replica of a blockchain. An increase in the number of participants makes the blockchain system complex and leads to the saturation of the network. This leads to substantial transaction costs to process data with increased storage space, degrading network performance.
Data stored in the blockchain cannot be tampered with during and after block generation. An adversary will fail to modify historical data stored on the blockchain because of cryptographic techniques used in distributed blockchain storage: (1) Asymmetric Key - that each node uses to sign and verify the integrity of the transaction, and (2) Hash function - a mathematical algorithm that maps arbitrary size data to a unique fixed-length binary output. A hash function (e.g., $SHA-256$) is computationally infeasible to recover input from output hash. An attacker fails to tamper with a block after a size t of the blockchain sufficiently secure according to *reference*, with t as the number of blocs in the blockchain. Even if the adversary tries to cover up this tampering by breaking the previous block's hash and so on, this attempt will ultimately fail when the genesis block is reached. It is complicated to modify data blocks across the distributed network. A small change in the original data makes the hash unrecognizable different, which secures the blockchain.

2.1 Blockchain

Blockchain is a decentralized data structure consisting of a sequence of blocks that record transactions maintained by nodes without the need for a central authority. The block header and the block data are the two main parts of each block. As illustrated in Figure 2.1, each block contains the previous hash block, approved and replicated among different nodes based on consensus. When nodes mine a new block and broadcast it in the network, all nodes verify it, and its hash block is added to the last block in the chain. Nodes communicate by sending messages through a Peer_to_Peer network topology on top of the internet. Transactions are entered into the blockchain chronologically and stored as a series of blocks replicated over multiple nodes. Each node records and saves an identical copy of the ledger. The transaction is a record of an exchange between two or more parties, e.g., one party provides a service while a second party pays for it. A party (sender) must have some funds and sign a statement transferring them to another party (receiver). The transaction is signed using asymmetric cryptography with a private key from the participant to anonymous participant(s) who are only known by their public keys. Merkle trees are a form of a data structure for storing transactions.

2.1.1 Data Structure

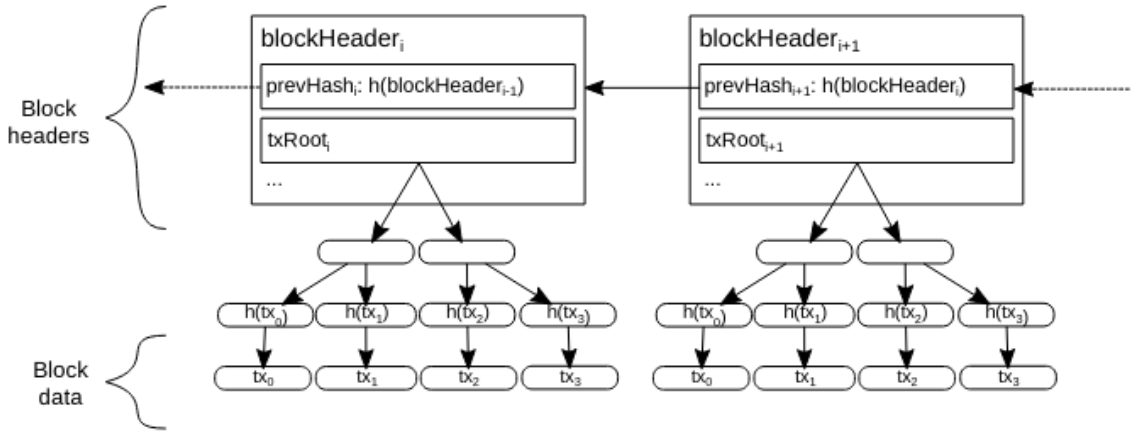


Figure 2.1: Blockchain structure.

2.1.2 Genesis Block

The first block in the chain is known as the genesis block. The distance between a block B and the genesis block is called the height. The genesis block has a lower height of 0, and the last block in the blockchain has a higher height of H. Nodes are incited to mine in the longest chain to prevent the creation of forks.

2.1.3 Fork

The open environment of blockchain makes it difficult to reach a consensus because anybody may join, and an adversary can create an infinite amount of pseudonyms (or "Sybils [44]").

Bitcoin and other cryptocurrencies attempt to solve this issue. Proof_of_Work guarantees that an adversary cannot benefit from using pseudonyms, with PoW requiring users to compute hashes to expand the blockchain repeatedly and only accepting the longest chain as authoritative. Forks, where two distinct blockchains have the same length but neither trumps the other, are possible with PoW. In order to prevent forks, users must wait for numerous blocks to ensure that their transactions are kept on the reliable chain; 6 blocks are recommended in Bitcoin [2]. The time it takes to add a new block must be sufficiently long, i.e., 10 minutes. As a result, Bitcoin transactions are confirmed in around an hour.

There are two types of forks:

- Soft Forks: in this case, the new versions of the software are compatible with the previous ones. Soft forks occur when part of the community wants to adopt

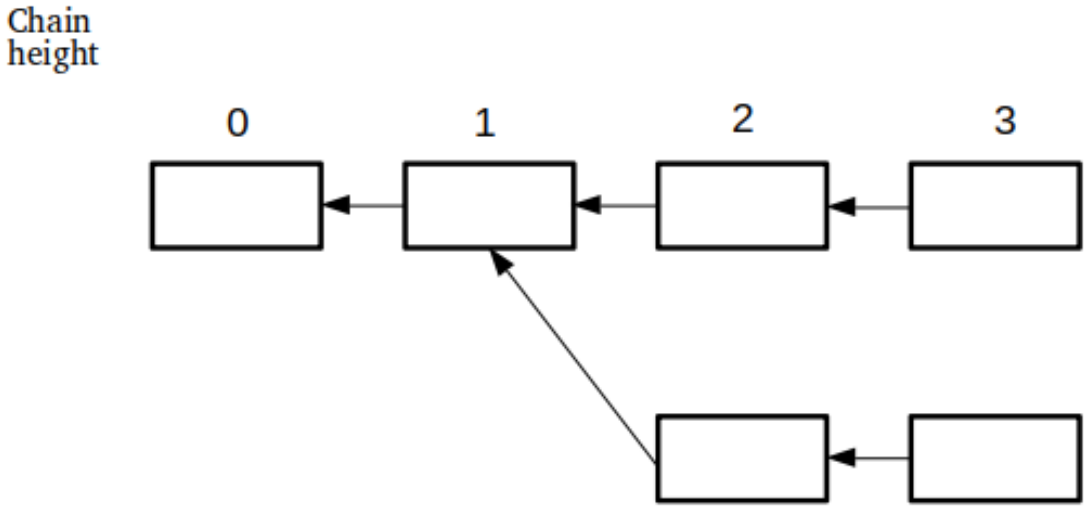


Figure 2.2: A blockchain fork.

a modification that is supposed to benefit users without contentious change: users are free to accept it.

- Hard Forks: Protocol changes that make old versions invalid. Hard forks usually happen when there is a disagreement between different communities: this is how Bitcoin Cash, an alternative version of Bitcoin, came into being in 2017.

Suppose a miner or a mining pool obtains a significant portion of the computing power. In that case, it can, in theory, use it to attack the consensus mechanism and endanger the security of availability of the Bitcoin network. But it's important to note that consensus attacks, called the 51%-attack: At best, it can only affect future trades, the most recent blocks. Beyond a certain depth, blocks become immutable, and the transactions they contain are impossible to censor. Do not affect the security of private keys and the Bitcoin signing algorithm: stealing bitcoins, spending unsigned bitcoins, redirecting bitcoins, or changing the blockchain history is impossible. But they can allow you to spend the same bitcoins twice: the famous double-spending. Indeed, if a miner controls more than 50% of the computing power, he can produce a chain of transactions longer than all the other miners combined. He can thus: Go back on your past transactions Refuse to validate transactions of other actors It is, therefore, vital to the ecosystem of any cryptocurrency that honest miners control more than 50% of the computing power. Figure 2.2 illustrates a blockchain fork.

2.2 Block Validity and Network Nodes

In most blockchain networks, the network is governed by two overlapping sets of players. : Miners (or block producers) and full nodes. The blockchain synchronizes at regular intervals by propagating a block created by one miner. There are numerous nodes of such continuous blockchains on the blockchain network. It operates as a decentralized ledger. Whenever a new block is created, the transaction receives a digital signature fingerprint that cannot be changed and comprises hashtag functions from the preceding block with a unique output. When the output is altered without being checked, the transaction holds no validity and becomes unverified. This implies that when the hash is executed, all network nodes must get the same result.

Blockchain networks generally fall into one of two categories, depending on the control of network participation, *permissionless* and *permissioned*.

- A permissionless blockchain allows nodes to join and leave the network without authorization as long as each node has a valid nickname (account address) and can send, receive, and validate transactions and blocks according to established regulations. They are known as a public blockchains. This means anyone can become part of the network and participate in the consensus. However, their voting power is often inversely correlated to the ownership of their network resources, such as their computing power, token value, storage space, etc. When using a permissionless blockchain, the operating environment is often considered zero-trust, which often cautions the community against using more efficient consensus schemes or expanding transaction processing capacity.

- A permissioned blockchain restricts specific actions to specified addresses. The network's participants can limit who can participate in the consensus mechanism and who can develop a smart contract and give specific participants the capacity to validate transaction blocks. The blockchain nodes are accessed through a control access layer. Their queries, though, are: Who can provide permission? A permission blockchain may give its owners confidence by providing the database with advanced security and privacy features. Still, it may be construed as a violation of the blockchain concept because only a few members have more control, implying that they can make changes regardless of whether or not the rest of the network agrees.

2.2.1 Cryptography in blockchain

The primary characteristics of blockchains are security, unchangeable records, and verification. The various blocks are linked together by linking hashtags, and each

block contains the hash code of the previous block, which is derived from the values produced when the new block comes up. This section presents the basic cryptographic building blocks that underpin blockchain protocols.

—**Hash Functions** A cryptographic hash function is a publicly known function H that takes in an arbitrary-sized input x and returns an output of fixed size H(x). There are three desirable properties for a cryptographic hash function:

- Pre-image resistance. Given a hash y, it is computationally hard to find any x such that H(x)=y.
- Weak collision resistance. Given an input x_1, it is computationally hard to find a different input x_2 such that H(x_1)=H(x_2).
- Strong collision resistance. It is computationally hard to find any x_1 and x_2 such that H(x_1)=H(x_2). This implies weak collision resistance.

—**Signature** Let's give an example with two participants, Alice and Bob. If Alice wants to send a transaction to Bob, Alice will create her transaction, which she will hash with a hash function and sign by her private key. Remember that the blockchain uses asymmetric cryptography, consisting of a public key to encrypt the plaintext and a private key to decrypt the ciphertext (Symmetric cryptography algorithms use the same key to encrypt and decrypt). The transaction is then transmitted to every network node. To obtain network approval, those nodes, known as miners, will gather the transactions in a block, verify the transactions in the block, and broadcast the block using a consensus mechanism (Proof_of_Work, proof_of_Stake, etc.). When other nodes verify that all transactions in the block are valid, the block can be added to the blockchain.

Only when the other nodes approve the block containing Alice's transaction and add the block to the blockchain does this transaction from Alice to Bob become finalized and legitimate. Figure 2.3 and Figure 2.4 provide an illustration of this process.

Digital signatures are practically impossible to falsify because they are based on number theory. Users who employ "public key cryptography" have a pair of keys, a public key, and a private key. Public key cryptography uses encryption to guarantee security and protect sensitive key information. Digital signatures cryptographically connect an identity to a massage, allowing digital messages to be authenticated. To summarize the digital signature process used by Alice and Bob in sending and receiving transactions, we represent each of the two processes, signature and verification, in three phases, as follows:

I Signature: Signing uses both asymmetric cryptography and hash functions.

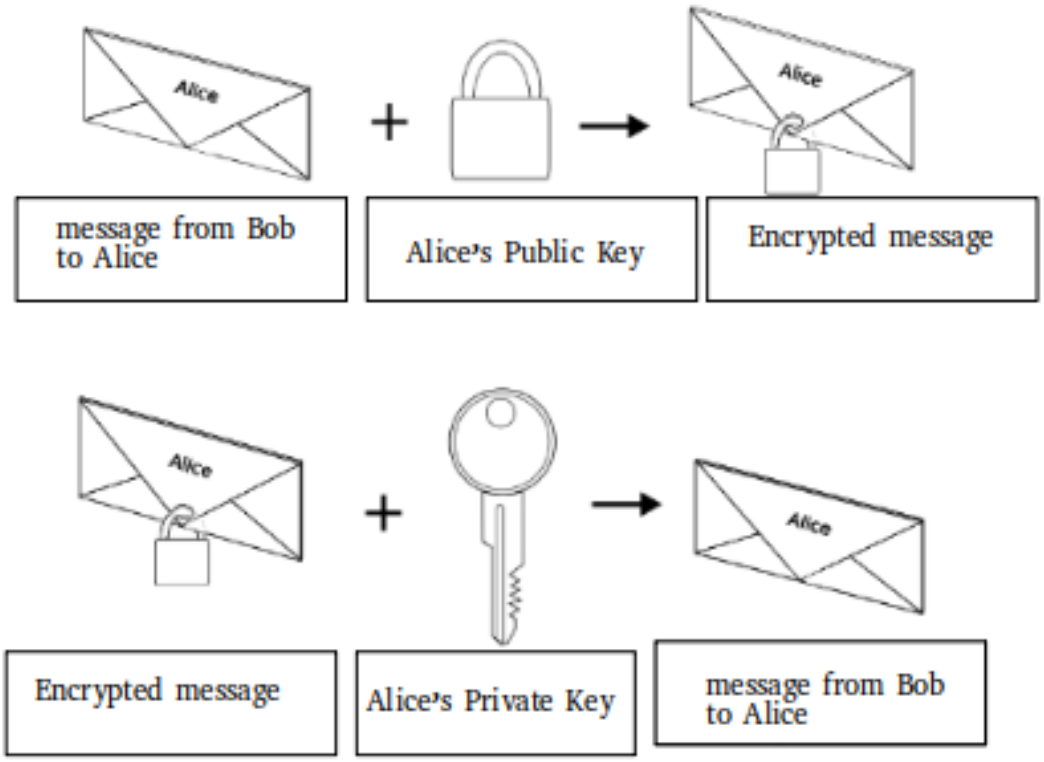


Figure 2.3: Encrypted Message.

In fact, we can derive the five features of a signature by combining these two processes (authentic, tamper-proof, non-reusable, unalterable, irrevocable).

i Create a transaction to sign

ii Generate the transaction's hash using the hash function.

iii Sign the hash with the private key.

II Verification

i Decrypt the signature using the issuer's public key.

ii Recreate hash from the original transaction using the same hash function as the issuer.

iii Compare recovered signer to claimed signer

If the two hashes are identical, the signature is validated.

The Elliptic Curve Digital Signature Algorithm (ECDSA) is the current signature algorithm used by Bitcoin. Compared to the RSA system, this uses shorter keys and requires lower computational while offering high security. Instead of finite fields, ECDSA makes use of "elliptic curves." A finite set of points on a curve known as an elliptic curve is one in which some operations are straightforward in one direction

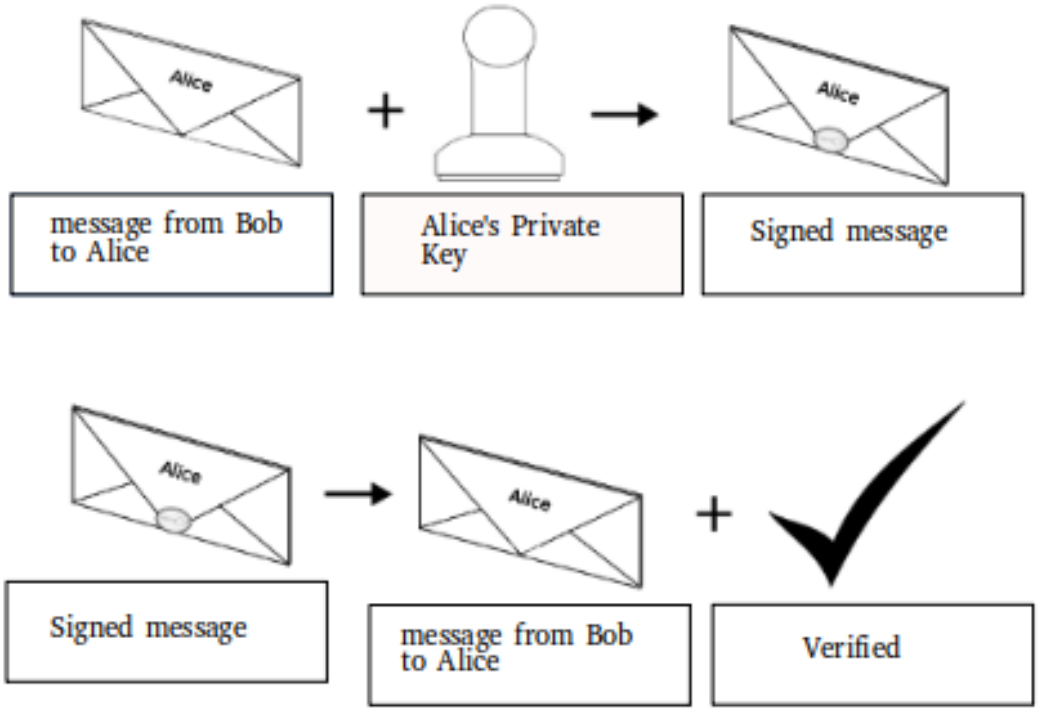


Figure 2.4: Signature.

but challenging in a reverse way. Instead of relying on the challenge of factoring primes, ECSDA uses the discrete log problem. The problem is formulated as follows:

"Let a, b, and c be integers such that ab = c. If you are given c and a, it is difficult to find b if b is a large enough number. Now apply this equation to an elliptic curve group and compute $\boldsymbol{Q} = k*\mathcal{G}$*, where n is the private key (some integer),* $\mathcal{G}$ *is a point on the curve, and* $\boldsymbol{Q}$ *is the public key (the result of the operation). The private key k is generated as a secure random integer in the range [0...n-1]. The public key pubKey* $\boldsymbol{Q}$ *is a point on the elliptic curve, calculated by the EC point multiplication: pubKey = privKey*$\mathcal{G}$*. It is easy to calculate* $\boldsymbol{Q}$ *given k and* $\mathcal{G}$ *in elliptic curves. It is difficult to find k given* $\mathcal{G}$ *and* $\boldsymbol{Q}$*. This is known as the "elliptic curve discrete logarithm problem."*

Bitcoin uses a specific curve called "secp256k1", standardized by the U.S. government agency, the National Institute of Standards and Technology (NIST) [8].

2.2.2 Nodes

Each node in the network follows the same set of rules, including consensus protocol, transaction processing, and block creation. Nodes communicate by sending messages through the Gossip protocol and form a P2P network topology on top of the internet. Each node replicates and saves an identical copy of the ledger on its computer, which updates itself according to the protocol's rules. Nodes act as peers connected over

a Peer_to_Peer network. Miners responsible for the chain's evolution by adding new blocks each round according to the rules followed by the network nodes can have two rules of *Uploader* and *Downloader*. It uploads to the network one block by round and downloads the blocks and transactions of other miners.

Nodes in Bitcoin act similarly to BitTorrent [117] [112]in that it is made up of computers and servers that host full nodes, making it resistant to attacks. In addition, full nodes contribute to network security by maintaining a complete copy of the blockchain.

2.2.3 Incentive Mechanism In Distributed System

Transaction in Bitcoin uses The Unspent Transaction Outputs (UTXO) data model, which can have a single or multiple inputs and outputs. Permissionless blockchains allow any node to join or leave the P2P network without authentication. When a new transaction is broadcast on the network, it is received and verified by a group of nodes to ensure that it is correctly signed and has not been previously recorded in the blockchain. Once verified, the transaction is added to a valid transaction block by miners competing and executing a consensus to extend the longest chain with blocks they generate.

Traditional blockchain has robust incentive mechanisms to encourage nodes to maintain the network running economically. For example, miners are rewarded for their labor by getting a fee each time their proposed block is accepted. Besides Bitcoin, file sharing is the most common use of P2P technologies such as BitTorrent. Nodes in Bittorent participate in a game called the 'optimistic tit-for-tat algorithm'[32]. In this game, nodes share data reciprocally with other nodes that share data with them. Network participants use information about favors to calculate peer rankings. The participants then use these rankings to determine how they will share their resources with other network participants preferring those who have higher rankings. Occasionally, nodes share data at random, interacting with each other without taking peer rankings into account. This game leads to a *Nash equilibrium* [110] where all nodes are incentivized to share resources (data) freely at the network's maximum capacity and perform prosocial behaviors.

Network models can generally be synchronous, asynchronous, or partially synchronous.

- **Synchronous**: In synchronous networks, the maximum network delay of message propagation delays between nodes has a predetermined upper bound ($\Delta-$synchrony) known to the participants in which messages are sent to all nodes. Semi-synchronous message delivery time can be defined by a random

variable whose probability distribution is known to the participants.

- **Partially-synchronous**: There is a predetermined maximum network delay unknown to network participants. However, they are confident that all messages will eventually reach their intended recipients. In a second scenario, there is also an unspecified global stabilization time (GST), ensuring that all communications between two honest nodes arrive on time after GST (the network will then become synchronous). Partial synchrony provides good adaptability to the natural network dynamics while simplifying network modeling.

- **Asynchronous**: The most challenging scenario is to reach a consensus. There is no maximum network delay in this configuration. That is, communications could take unlimited time to reach their intended receiver. A practical blockchain network is partially synchronous similar to most distributed networks, which can benefit from Internet synchronization services.

Bitcoin operates asynchronously up to an unknown global settling time (GST), ensuring that all communications between two honest nodes arrive in time after GST (the network will then become synchronous).

2.3 Blockchain properties

A block is said to be safe if there is a small probability that it will be rejected once it has been confirmed (since it is buried far enough deep in the longest chain). While safety is a core security property of a blockchain protocol, liveness is also crucial since it determines whether (honest) transactions are included in blocks and whether those blocks are part of the longest chain. Liveness guarantees that all transactions enter the ledger, while security guarantees that the transactions eventually remain permanently in the register (with high probability). Together, liveness and safety guarantee the blockchain protocol's security.

The first property, the *common prefix* property, states that, after eliminating some blocks from the chain, the chains of honest parties are always a common prefix of one another. It indicates that the oldest chain is a prefix of the most recent chain when two honest parties possess chains at two different rounds from [55]:

Definition 2.3.1 ***(Common Prefix).*** *The common prefix CP, parameterized by a value $p \in \mathbb{N}$ states that for two honest parties P_1 holding chain L_1 at rounds r_1, and P_2 holding chain L_2 at rounds r_2, with $r_1 \leq r_2$, the common prefix property states that $L_1 - p \leq L_2$.*

The second property, the *chain quality*, measures the number of real contributions in a sufficiently long and continuous segment of a party's chain. We analyze chains

of varying complexity. Therefore it is more appropriate to consider the parties' contributions in terms of the overall difficulty they bring to the chain rather than the number of blocks they add. The property asserts that adversarial parties have a maximum amount of difficulty they can add to any sufficiently long segment of the chain.

Definition 2.3.2 ***(Chain Quality).*** *The chain quality CQ of the longest chain L for a blockchain is the fraction of honest blocks in chain L.*

2.3.1 Longest Chain Rules

Since there is no central authority in Bitcoin's decentralized network, the longest chain rule implies that the stabilized prefix of the longest chain can be a common reference to the network's history. The hashing power comes from a real hardware investment, which makes it very difficult for attackers to tamper. A higher mining difficulty during block production requires more brute-force trials to find a suitable nonce. The estimated block interval is modified every 2016 block to maintain a consistent value regardless of changes in gross hashing power. This ensures that each block is sufficiently propagated before the next one is produced.

Bitcoin uses the longest chain rule to maintain a single blockchain and prevent conflicts between miners. Conflicts occur when several miners produce blocks simultaneously (in competition), with each miner considering their block as the legitimate one that should be put into the blockchain. For instance, if two miners, A, and B, attempt to add a block number n, A will create the block n_A, and B will create the block n_B. Both blocks have the generator address for the block reward and may have different transactions. Then, because blocks are not created and shared immediately throughout the network, every block assumes legitimacy. So, it adds it to its chain and begins building on the following one (block $n + 1$). According to the longest chain rule, if B is quicker than A and creates the block $n_B + 1_B$ before $n_A + 1_A$, then A must accept B's chain ($n_B + 1_B$) as the legitimate one and remove the shorter chain (n_A), which is referred to as an orphaned chain/block.

Suppose we assume that most of the computational power was honest for the duration of history. In that case, this ensures that, at all times during the execution, the longest chain represents the correct history of the world, except for the most recent k blocks of the blockchain ($k = 6$). The adversary cannot modify blocks before that due to the Common Prefix [55] property of blockchains. The longest chain represented the correct history if most of the computational power was honest, except for the k last blocks.

2.4 Consensus

It is an agreement to validate the correctness of a blockchain, which is linked to the order and timing of the block. Its goal is to achieve consistency of the nodes participating in blockchain. All honest nodes accept the same order of transactions as long as they are confirmed in their local blockchain views. The blockchain is constantly updated as new blocks are received.

The best-known consensus on active blockchains is Proof_of_Work. Each peer competes with other nodes to solve mathematical puzzles using its computing power, which involves discovering a nonce value that must be less than the current target value when hashed with additional block parameters (e.g., a Merkle hash, the previous block hash). When a nonce of this type is found, the miner builds a block and propagates it to the network layer using the Gossip protocol to propagate it around the network. Other nodes that receive the block halt mining and check the validity of the new block, I.e., peers validate the Proof_of_Work by computing the block's hash and comparing it to the current target value. If correct, they will keep mining the next block. The entire procedure was visible to all of the miners.

Consensus objectives. A blockchain consensus protocol's objective is to guarantee that all participating nodes concur on a single network transaction history that is serialized as a blockchain. We define the following objectives for blockchain consensus:

- **Termination**: A new transaction is rejected or accepted into the blockchain, recorded in a block at every honest node.
- **Agreement**: All trusted nodes should accept or reject each new transaction and its holding block. Every honest node should assign the same sequence number to an accepted block.
- **Validity**: A valid transaction or block should be accepted into the blockchain if every node receives it.
- **Integrity**: All accepted transactions should be consistent at every honest node (no double spending). All approved blocks must be correctly generated and hash-chained in order.

Strong consistency is handed up in favor of final consistency, which simplifies communication between nodes and is a characteristic of the standard Proof_of_Work algorithm.

2.5 Bitcoin and Proof_of_Work blockchains

The first cryptocurrency to appear was Bitcoin in 2009. Cryptocurrencies are digital currencies that rely on cryptographic algorithms to work correctly. Right now, Bitcoin is still the most secure cryptocurrency and the best illustration of how "blockchain" protocols. Bitcoin uses Proof_of_Work to create blocks. Blocks in Proof_of_Work blockchains satisfy the equation Hash(B)$\leq$ T, where T stands for the mining target [25] and B for the block. Block is a triplet that includes a nonce, the application's data and metadata, and a reference to the block before it by its hash. The function H is a hash function modeled as a random oracle, which outputs κ bits, where κ is the security parameter of the protocol and T $< 2\kappa$ [19].

(i) hash chained storage, (ii)digital signature, and (iii)commitment consensus are three fundamental and significant features supported by Bitcoin's blockchain.

The blockchain uses asymmetric cryptography to sign the transaction using a private key from the participant to one or more anonymous participants only known by the public keys.

Nakamoto enhances the termination requirement with the *probabilistic finality* statement compared to the consensus aim provided in Subsection 2.4. Every new block is rejected or accepted in the local blockchain of every honest node. An accepted block can always be rejected, but this probability rapidly decreases as the blockchain expands.

The hash-intensive Proof_of_Work mechanism is intended to prevent Sybil attacks. Attackers using Sybil [44] can easily create new identities or accounts because Bitcoin is permissionless and pseudonymous.

2.5.1 Nakamoto Protocol

Except for the genesis block, the block header also provides a hash of the preceding block and additional configuration data, such as a timestamp at block generation. Let $G(.)$ and $H(.)$ be cryptographic hash functions with output in $\{0,1\}^k$. A block with target $T \in \mathbb{N}$ is a quadruple of the form $B = (prev, txs, t, n)$ where $prev \in \{0,1\}^k$, $txs \in \{0,1\}^*$, and t and $n \in \mathbb{N}$ are such that they satisfy $validblock^T(B)$ defined as

$$H(n, G(t, prev, txs)) < T$$

The blockchain is the sequence of blocks where the last block is the one with the greatest height, denoted B_L. A chain L with $B_L = (prev, txs, t, n)$ can be extended to a longer chain by appending a valid block $B_{new} = (prev', txs', t', n')$ that satisfies $prev' = H(n, G(t, prev, txs))$ and $t' > t$, where t' is the timestamp of block B. We have an extended chain $L_{new}= LB_{new}$.

Each node will calculate whether the 2016 blocks were mined quicker or slower than 10 minutes on average after every 2016 block by examining the interval between the previous 2016 blocks. The target T would adjust downward to make it more challenging to fall below the target for the following period of blocks if blocks during this time were mined more quickly than every 10 minutes. On the other hand, if blocks were being mined more slowly than every 10 minutes, the target T would adjust upward to make it simpler to fall below the target for the upcoming block period. There is a function $D : \mathbb{Z}^* \implies \mathbb{R}$, which receives an arbitrary vector of round timestamps and produces the next target. A block's *difficulty* is determined by how many times it is more challenging to mine than the genesis block, the first block ever mined.

The Nakamoto protocol is suggested by Bitcoin-NG [52] to elect a leader who would then publish blocks of transactions, resulting in order of magnitude improvement in the latency of confirming transactions over Bitcoin.

2.5.2 Miners

The miners need to get the whole state to validate new transactions. Thus, miners download the *headers* block for the entire chain and full blocks only for blocks near the end of the chain. In Proof_of_Work protocol, the transaction is shown as unconfirmed until the transaction is 6 blocks deep, which takes about an hour. (e.g., for a transaction tx, if a node submits it at some time t and appears in the blockchain at time t', then t'-t is the transaction confirmation delay and corresponds to 6 blocks in Bitcoin.) Note here that the miner does not need to verify the integrity of all historical transactions:

2.5.3 Transactions

Every peer on the blockchain receives both transactions and blocks. Before building a new block, nodes must validate transactions; when a node receives a transaction that transfers bitcoins from an account, the transaction that made those bitcoins available on the account must be confirmed already in a previous blockchain block. The same bitcoins are only spent once; otherwise, it would be a double-spend attack. Valid transactions are grouped into a valid block. Transactions will be confirmed when mined in a block, and the network remains consistent in a common order replicated over the peers. The new block is distributed to all the nodes in the network, and each node stores the block. At this point, the state of the distributed replicas changes.

The transaction is a digitally signed statement (currency) indicating digital ownership (asset) transfer.

A peer stores the transactions they receive into a local buffer (unconfirmed transactions). Each transaction transfers previously received or newly minted bitcoins to one or more receivers through individual transaction outputs. To avoid double spending using a dedicated scripting language, full nodes must verify any claimed ownership of bitcoins for all pending transactions. Each transaction output sets a spend condition using this scripting language, which typically sends coins to a specific address corresponding to a cryptography pair of keys. The average number of transactions per second (TPS) is obtained by dividing the number of transactions in one block given by:

$$Number\_of\_transactions = block\_size/transaction\_size$$

; by block time (i.e., 600 seconds in Bitcoin).

A Merkle tree is frequently used to structure transactions. The Merkle tree is a common data structure for data storage and efficient data integrity checking. [104]. The block header contains the Merkle tree root, where a transaction hash identifies each leaf node.

2.5.4 UTXO model

Transaction in Bitcoin uses The Unspent Transaction Outputs (UTXO) data model, which can have a single or multiple inputs and outputs. The inputs are the previously unused transactions that belong to the sender, and the output is the receiver address which defines how many tokens go to which receiver. All nodes keep track of outstanding transaction outputs that have not yet been spent for efficiency reasons (UTXO set). When a transaction attempts to spend bitcoins that do not exist or have already been used, nodes may reject it. Notably, all spent transactions – those that no longer add to the UTXO set – are automatically removed from the transaction indexes of full nodes by default.

2.6 Arweave

In Arweave[138], authors propose a blockchain that provides permanent storage by incentive. Numerous blockchains (i.e., Solana [142]) that achieve scalability use Arweave for storing transactions and only keep a few blocks on their On_chain. Solana is a high-throughput blockchain that uses a network timestamp technique called Proof_of_History to achieve sub-second block times and high throughput. If Solana operates at maximum capacity for a year, it will generate 4 petabytes of data. At all times, the full blockchain must be stored and accessible. Arweave introduces a new block structure called "blockweaves" related to two previous blocks, i.e., in

addition to a pointer to the last block in the chain, blocks point to another randomly chosen "recall block" of the prior history of the blockchain. When miners solve the problem and find an appropriate hash, they can share the new block and recall block with the network. Arweave is constantly growing. Although its blockchain is designed to be archived forever, few nodes are incentivized to store old chain data. This problem gets worse as the blockchain network grows and has a storage problem.

2.6.1 Incentive Mechanism In Arweave

Arweave set out to solve the problem of long-term data storage on the internet. It used blockchain technology to store data and implemented an incentive system for peers to store data permanently. Arweave constructed a blockweave instead of a blockchain. The blockweave is held together by many links inside the whole data storage. Therefore, the only way to add new data to the blockweave is for a peer to recall a randomly selected block already on the blockweave. Only the peers that can recall the previous random block are allowed to store new data [138]. Miners have the incentive to store vast amounts of data to enhance their chances of finding a good recall block because the selection of recall blocks is random. This improvement in data storage and economic rewards for miners who keep the data create conditions for data to be stored for long periods. In addition, Arweave introduces a synchronization block generated once every 12 block, containing a full list of the balance of every wallet in the system and a hash of every previous block without the transaction. Synchronization blocks help new participants in the network to bootstrap, and there is no need to download all blocks from the genesis block. Node rates its peers based on two main criteria: the peer's generosity - sending new transactions and blocks - and the peer's responsiveness - responding quickly to information requests. Instead, it allows each actor to maintain private, local scores for other peers. Nodes are incentivized through the Adaptive Interacting Incentive Agent (AIIA) meta-game [139], i.e., nodes with low AIIA social rank risk their messages (including new candidate blocks) being propagated too slowly to be accepted by the network.

Definition 2.6.1 *Proof of Random Access. To mine or verify a new block, a node must have that block's recall block. Demonstrating proof that the miner has access to the recall block is part of block construction (conversely, verifying this proof is part of validating a new block). Proof_ of_ Access is an enhancement of Proof_ of _ Work in which the entire recall block data is included in the material to be hashed for input to the proof of work.*

2.7 Ethereum

Ethereum [140] popularised the concept of smart contract to implement any application on top of the blockchain or meant actual word certification to token validated by the blockchain. Furthermore, the consensus aims to provide finality, which means that no value inserted into the ledger can ever be withdrawn. Once a value is registered, it is hard to change it. Security is the responsibility of the nodes that maintain the blockchain. The more nodes store the blockchain, the more secure it is. The replication of the blockchain across all nodes that constitute the network is a hugely inefficient use of the memory storage of the system because the node in a blockchain system (such as the Bitcoin system) does not rely on a central organization, and each node stores a full copy of the transactions. This aspect, however, indicates that the size of blockchain transactions is overgrowing. Therefore, the node memory needs to be expanded to support the system running with continuous system operations. One of the issues with blockchain scalability is the growth in storage costs as the number of transactions increases, necessitating substantial memory capacity to store the blockchain and which excludes small nodes from the validation process. The accounting/balance models used in blockchain (UTXO model used in Bitcoin and the account model used in Ethereum) need to be stored for validation, called the validation state. Ethereum does not record all transactions but instead keeps track of the sequence number ("nonce") of the most recent transaction issued from a specific account [26]. Even if the account has no balance, this nonce must be saved and causes storage costs to grow linearly and unintentionally caused Ethereum issues when a smart contract establishes several zero-balance accounts. Furthermore, designing a cryptocurrency with storage costs that scale well with the number of users and transactions is difficult due to various constraints.

2.8 51% attacks

The fundamental issue with cryptocurrencies is double-spending, n which a miner or a group of miners trying to spend their cryptocurrency twice on a blockchain is known as a 51% or double-spend attack. The name comes from their attempt to "double" them. The purpose of this attack is not always the intent to double cryptocurrency spending; it is more common to undermine the integrity of a particular cryptocurrency or blockchain. For example, a transaction approved by a Bitcoin owner is added to a local pool of unconfirmed transactions. Miners create a block of transactions by choosing transactions from these pools. Then they must solve a difficult mathematical puzzle before adding that block of transactions to the blockchain, using computer power, and striving to find that answer. However,

a malicious miner may attempt to roll back transactions that have already been completed. When a miner discovers a solution, they should broadcast it to all other miners so that they can check it out. The block is then added to the blockchain (the miners come to a consensus). However, by not broadcasting the solutions of his blocks to the rest of the network, a dishonest miner can produce offspring of the blockchain. As a result, they create two versions of the blockchain. One version is tracked by uncorrupted miners and another by corrupted miners, isolated from the rest of the network. The corrupt miner develops their version of this blockchain and does not distribute it to the rest of the network. Because it is not aired, the rest of the network is unaware of this channel. Now that all other miners operate on the valid blockchain, the corrupt miner can spend all their bitcoins there. For example, suppose he uses the money to buy a house. Its bitcoins are currently spent on the trusted blockchain. Unfortunately, it is slow to include these transactions on its isolated version of the blockchain. Regarding its only version, it is still in the process of gathering blocks and performing its separate validation of the blockchain in the meantime. Blockchain is set up to respect majority rule or democratic governance. Because most miners add blocks to their version of the blockchain faster than the rest of the network (so, longest chain = majority), the blockchain does this by always following the longest chain. This is how the blockchain decides which version of its chain is valid and is the basis for all wallet balances. The corrupted miner will now attempt to add blocks to their isolated blockchain faster than other miners who add blocks to their respective blockchains (the truthful one). The corrupted miner immediately passes this larger version of the blockchain to the rest of the network as soon as they are done creating it. The protocol now forces them to switch to this chain if they realize that this (wrong) version of the blockchain is longer than the one they were working on. There will be an immediate reversal of all transactions not part of the corrupted chain. It is now accepted that this chain is the most reliable. The attacker once used his bitcoins to buy a house, but this transaction was not recorded in his isolated chain; however, since the chain is again the fundamental chain, he owns those bitcoins again. Again, he can use them.

2.9 Proof_of_Stake consensus protocols

A stake refers to a participant's coins or network tokens that can invest in the blockchain consensus process [114]. From a security perspective, PoS uses token ownership to mitigate Sybil attacks. The Bitcoin community created Proof-of-Stake (PoS) as an energy-efficient replacement for Proof_of_Work mining. Unlike miners in Proof_of_Work, which uses brute force, block validators in PoS use their stakes. The stake value of a PoS miner determines how probable it is to propose a block.

Authors in [141] summarize the PoS protocols in four classes: The first classe is the basic and the first alternative to Proof_of_Work, *blockchain-based PoS*. Its process is similar to the Proof_of_Work, except in how a validator bloc calculates the block hash. Unlike Proof_of_Work, it does not rely on expensive hashing to produce blocks. The estimated number of hashing attempts for a minter to solve the puzzle can be significantly decreased if her stake value is high because the complexity of the hashing puzzle decreases with the minter's stake value. PoS prevents the brute-force hashing competition that would occur if using Proof_of_Work, resulting in a considerable decrease in energy consumption. Most protocols using PoS are [85] and [9].

Committee-based PoS is an alternative mechanism. It creates and enables a committee of stakeholders to produce a new block according to their stakes. The committee-based PoS leverages a multiparty computation (MPC) scheme to determine a committee to orderly generate blocks. MPC is a type of distributed computing where multiple participants start with different inputs and output the same result [85]. The MPC process essentially implements functionality in the committee-based PoS that takes the current blockchain state, including the stake values of the stakeholders. It outputs a pseudo-random sequence of stakeholders (the leader sequence), which will subsequently populate the block-proposing committee. All stakeholder-sĺeadership positions should be equal, and those with higher stake values may occupy more spots in the sequence. Bentov's chain of activity (CoA) [21], Ouroboros [83], and Ouroboros Praos [38] are popular committee-based PoS protocols. These protocols were created concurrently by academics in the year 2017.

Part II.

Blockchain-based PKI system

Chapter 3

IDENTITY VERIFICATION BY SMART CONTRACT

3.1 Introduction

Internet services are increasingly being utilized in daily life for electronic commerce, Web-based access to information, and interpersonal interactions via electronic mail or voice. However, there is still significant concern about the trustworthiness of these services. Traditionally, public key certificates (PKIs) [34] is used for individuals, enabling secure communication protocol, e.g., Alice and Bob both need to have authentication certificates from a Certificate Authority (CA) to securely communicate with each other. The CA will then issue them digital certificates containing their public keys, which they can then use to encrypt the messages they send to each other. To encrypt a message, Alice and Bob will use the public key they received from the CA. The message will then be encrypted with the receiving party's public key and sent to the recipient. When the recipient receives the message, it will be decrypted with the private key they have, ensuring that only the intended recipient can read the message. Two main standards are used for public key encryption of messaging: S/MIME for Secure/Multipurpose Internet Mail Extensions and PGP. The main difference between S/MIME and PGP is how the user verifies that they have an authentic copy of another user's public key. In S/MIME, public keys come with a certificate from a CA. In PGP, users create a web of trust to establish the legitimacy of the tie that binds a public key and its owner. A web of trust is decentralized and serves as an alternative to its centralized CA. Still, they face multiple security threats, such as the Man In the Middle (MITM) [71, 33] attack and EFAIL attack [122, 113].

The blockchain is an innovative technology that overcomes these threats and allows for the decentralization of sensitive operations while preserving a high level of security. Moreover, it eliminates the need for trusted intermediaries. The blockchain is accessible to all network nodes; it keeps track of all transactions and ensures transparency. In this chapter, we propose a protocol for a blockchain-based messaging protocol. We explain why blockchain would make communications more secure. Our protocol maintains the performance and security of data recorded on the blockchain using a smart contract to verify the user's identities and associated public keys to validate the user's certificate. The protocol is decentralized and allows users to exchange messages securely.

3.2 State of the art in Blockchain-based PKI

This section presents the existing systems that use blockchain for secure user transactions. Blockchain has greatly interested engineers and investors because of its immense commercial potential and uses in applications as diverse as cryptocurrency.

Approaches	Trust Model	Decentralized	Protocol	On-chain storage	Update key
[135]	Hierarchical	No	N/A	Full	Yes
[90]	Hierarchical	No	Ethereum	Hash only	No
[35]	Hierarchical	No	Ethereum	Full	No
Cecoin[118]	WoT	Yes	Bitcoin	Full	Yes
CertCoin[53]	WoT	Yes	Namecoin	Full	Yes
Blockstack[13]	WoT	Yes	Bitcoin	Hash only	No
Our protocol [80]	Wot	Yes	Ethereum	Hash only	Yes

Table 3.1: Comparison

Since 2011, numerous approaches have been proposed. Namecoin [76](BitDNS) is the first to build a decentralized naming system using blockchain. It is the first altcoin from bitcoin. Satoshi believed Namecoin should use its independent blockchain, and the event offered the first proposal for merged mining to secure blockchain. With the use of distributed ledger technology, name-value pairs can be assigned and verified without the involvement of a third party. Moreover, human-readable identifiers can also be selected in a decentralized manner. Namecoin was the first solution to zooko's Triangle [136], producing a naming system that is simultaneously secure, decentralized, and human-meaningful.

Zooko's Triangle was thought of as a trade-off: Decentralized, Secure, Human-Meaningful Unfortunately.

Namecoin has its dedicated namespace (.bit) that maps a name to a value. Namecoin allows users to register a name and attach data, such as a public key fingerprint. Unfortunately, it suffers from 51% attacks because of its insufficient computing power.
Similarly, Blockstack [13] implements a naming service with human-readable names so that a Blockstack identity can be linked to their system using their blockchain name system. Again, it uses the existing internet transport layer (TCP [100], or UDP [116]) and underlying communication protocols and focuses on removing centralization points at the application layer. Blockstack employs the most straightforward approach, in which the user is responsible for key recovery and mobility, and the keys are kept on the device where the identity was generated. Twelve words are often used as a seed to generate the keys to make this method much more practical for mnemonic phrases. The work required to transfer keys from one system to another can be minimized using these phrases to reconstruct the private key.

Blockstack [13] and uPort [107] offer public profiles that include names, profile photographs, and signing keys. Blockstack primarily provides use cases of information dissemination to the public. Notably, data that needs to be publicly accessible

to reach its full potential, like social media accounts or PGP keys [130], can be securely stored on the blockchain. Emercoin [89] doesn't remove the central authorities but uses Blockchains to store hashes of issued and revoked certificates. Emercoin has the benefit of optimizing network access by performing key and signature verification on local blockchain copies.
Namecoin [76] had issues with name squatting, which was made worse by the lack of centralized control. In their analysis of the Namecoin namespace design, Kalodner et al. discovered that just 28 of the 120,000 registered domain names were not squatted or contained non-trivial content [76]. They contend that the names have some market value because human-readable identifiers are naturally rare compared to non-human-readable identifiers, such as key hashes or the public key, which are unlimited non-human readable identifiers.

Obviously, there is not and never will be a universal, global namespace with names meaningful to all possible users, as Carl Ellison emphasized in his 1996 work [36]. For a person to remember and associate meaning with all of the names, according to Ellison, is simply impossible.

> *it is clear that there is not a universal, global namespace with names that are meaningful to all potential users that do not exist and will never exist. [36]*

To reduce the issue of squatters, the Ethereum Nameservice (ENS) [140] these assessments to develop a decentralized bidding process. The uPort self-sovereign identity system [45] uses a Ethereum smart contract address is a reliable identifier for user identity. The address is derived from the smart contract's creator's public key. Christian Lundkvist of uPort believes ENS is a suitable naming layer to map the non-human-readable uPort identifier to a human-readable address [96]. Thus, Blockstack [13] uses its blockchain name system to offer a naming service with human-readable names to which a Blockstack identity can be linked for its system.
Unlike Emercoin, Certcoin [53] removes central authorities and uses the blockchain Namecoin as a distributed ledger of domains and their associated public keys. Every certcoin user stores the entire blockchain, and this causes two problems: the controller's latency and the security problems of merged mining used by Namecoin.

Blockchain based PKI: With the help of a decentralized solution provided by the blockchain, which offers certificate transparency, revocation, and trustworthy transaction records, central points of failure can be eliminated. Two strategies have been used in proposals for blockchain-based PKI from the perspective of the trust model.

Most approaches for blockchain-based PKI consider the blockchain as an append-only public journal and maintain CAs. [135] used the blockchain as public log-

in writing only to monitor CA certificate signing and revocation operations using TLS certificates. CA-signed certificates and their revocation status are submitted by the web server as a transaction and added to the blockchain by miners after verification of transactions. This solution still depends on a traditional CA role, and the blockchain primarily serves as the source for recording the certificate status. CertLedger [90] uses public blockchain as a public log to validate, store, and revoke TLS certificates. [35] employed the Ethereum platform for PKI management and public log to track keys. However, a centralized CA approach was used for adding and removing keys. The CA serves as the chainś root and issues the certificates. Cecoin [118] is a cryptocurrency in which digital certificates are treated like currency and are kept on a decentralized database whose states are updated by miners on a blockchain based on Bitcoin. To receive a certificate, the certificate owner must pay some coins to miners for their efforts and wait until they mine a new block using a PoW. To retrieve and verify certificates, they utilize a modified Merkle Patricia tree.

Table 3.1 compares our proposed system to similar research, where storage type indicates whether the full or hash of the public keys is maintained on the chain.

3.3 Background

In this section, we briefly describe the existing schemes we use in our contribution:

3.3.1 Public Key Infrastructure

A public key infrastructure (PKI) ensures that a specific entity is linked to its public key by relying on trusted key servers maintained by Certificate Authorities (CA) [64] Those authorities issue a certificate for a domain or a person that links that entity to a specific key publicly and verifiably, i.e., TLS ([119, 42] uses the X.509 certificate format [70]; establishing a secure communication channel and verifying the signature.

Traditional PKI installations are mainly centralized and, despite their use, have several vulnerabilities, such as fake certificates that would go unnoticed, allowing attackers to act as middlemen [146]. Public Key Infrastructures (PKIs) are a significant component of resolving network authentication and provide guarantees to trust a certificate signed by a certification authority(CA). PKI is used to offer confidentiality through encryption, authentication through signatures, and a web of trust through peer identity validation for his Pretty Good Privacy (PGP) [27] encryption technology. The certificates authenticate the public keys and allow to performance of cryptographic operations, such as encryption and digital signatures. CA is re-

sponsible for authentication and identity validation as a centralized system, which creates single points of failure.

Many authors have recently proposed blockchain technology for decentralizing key management in the context of Public Key Infrastructures; see Section 3.2.

3.3.2 Blockchain & smart contract

Public blockchain: The system is built on decentralized trust from the interactions of different participants on a peer-to-peer network. This system is a potential solution to achieve data integrity, relying on cryptography to provide tamper resistance. It is designed to make transactions more reliable. In such circumstances, it can be used to ensure secure communication and the integrity of data. Blockchain is decentralized, and no centralized authority can approve transactions. All network participants must reach a consensus to validate transactions securely. Previous transaction records cannot be modified because a very high cost has to be spent if an attacker wants to change historical data [57], especially in the blockchain using the Proof_of_Work protocol such as bitcoin and other altcoins [131, 31, 74]. Data immutability in this blockchain is strongest when the chain is long (see 1.9.2). Furthermore, external attackers would have to gain access to every computer in the network that hosts the blockchain database simultaneously to manipulate it, which is practically impossible. Recall that in blockchain, participants generate and propagate transactions to the rest of the blockchain network. Once validated, the transactions are added to a block, which will be appended to the blockchain by performing a mining process. In Proof_of_Work protocol, miners attempt to solve the hard cryptographic puzzle. The miner that solves the puzzle first adds the new block to the blockchain.

To summarize, What makes blockchain secure is:

- Public key infrastructure: It uses public/private key encryption and data hashing to store and exchange data safely.
- Distributed ledgers: There is no central authority to hold and store the data; it removes the single point of failure.
- Peer-to-Peer Network: The communication is based on the P2P network architecture and inherits the decentralized characteristics. P2P networks help overcome many problems that go beyond traditional client-server approaches.
- Cryptography: Blockchain uses cryptographic techniques, hash functions, and public and private keys. It is difficult to alter the blockchain; to make a modification, it is necessary to succeed in a simultaneous attack on more than 51% of the participants.

(A 51% attack on a blockchain occurs when a group of miners controls more than 50% of the network's mining hash rate. Attackers controlling most of the network can prohibit other miners from completing blocks, halting the recording of new blocks.)

- Consensus Algorithm: The rules which the nodes in the network follow to verify the distributed ledger. Consensus algorithms are designed to achieve reliability in a system involving multiple unreliable nodes. A consensus of the nodes validates the transactions. The choice of consensus algorithm has significant implications for performance, scalability, latency, and other blockchain parameters. The consensus algorithms must be fault-tolerant.

- Transparency: Each transaction is visible to anyone with access to the system. If an entry can not be verified, it is automatically rejected. The data is, therefore, wholly transparent. Each node of a blockchain has a unique address that identifies it. A user may choose to provide proof of identity to others.

Smart contract: A smart contract is a code stored and executed on a blockchain. The bytecode of the contract is replicated to all nodes. Ethereum is the most extensible public blockchain allowing the execution of smart contracts. A user can send a transaction that propagates the invocation information to the contact's address and triggers the smart contract execution. Each interaction with the smart contract is recorded as a "transaction" on the blockchain. This transaction also specifies the Gas price, which indicates how much the client is willing to pay for each computation unit carried out in a smart contract. The higher the Gas price, the faster the transaction propagates to the blockchain network. Once a smart contract is deployed, users cannot modify it. An agreement between two entities can be carried out automatically with the help of a smart contract.

DApp: The blockchain supports decentralized web apps, often known as DApps. A DApp typically refers to Java script software running on a website that accesses data on the blockchain by triggering a smart contract designed specifically for that DApp. For DApp web clients to interact with the Ethereum blockchain, remote procedure call (RPC) specifications are implemented. Every Ethereum client implements JSON-RPC. An RPC service accepts the JSON requests sent from a DApp client inside a web browser and translates them into queries or transactions.

3.3.3 PKI based blockchain

Blockchain technology can be used to replace Public Key Infrastructure (PKI) as a more secure and efficient way of storing and transmitting secure data. Blockchain technology is a distributed ledger technology that can be used to store encrypted

data securely and efficiently. This makes it ideal for storing and transmitting secure data such as secure communications, digital signatures, and other cryptographic data. Because the data is stored in a distributed ledger, it is more difficult for malicious actors to gain access to the data, making it a more secure option than PKI. Additionally, distributed ledger technology can provide faster secure data transmission than PKI, as it relies on a peer-to-peer network rather than a centralized server. This makes it ideal for applications where speed and security are important.

3.3.4 Cryptographic Primitives

Our protocol leverages some cryptographic primitives, such as digital signatures. It consists of three digital signature algorithms ($\mathcal{KG}$, $\mathcal{S}$, $\mathcal{V}$):

1. $(pK, sK) \leftarrow \mathcal{KG}(1^k)$: Key generation algorithm generates the public and secret keys given a security parameter k.

2. $\pi \leftarrow \mathcal{S}(sK, \mu)$: The digital signature on the message μ using the key sK generates π.

3. $\{0, 1\} \leftarrow \mathcal{V}(pK, \pi, \mu)$: The verifier checks if π is a valid signature for μ using the public key pK.

3.4 Overview

The proposed model uses the blockchain to validate the user's identity and ensure trust between users for exchanging messages with a high level of security. We are interested in building a decentralized, secure peer-to-peer messaging protocol using a PKI-based blockchain, which can be an email, a website, or some other form of message.

At the time of the digital economy, data is brought to transit more and more between companies, from a client to a supplier, moving from one cloud to another with the advent of virtualization and containers. This work describes the potential for applying blockchain to assure data traceability, certificate individual, and secure messaging based on blockchain technology. Our contribution aims to achieve the functionalities that PKI uses in the blockchain as the database to store public keys, digital signatures, and peer information, allowing each entity to validate information about every node in the network. Instead of relying on trusted key servers(i.e., centralized or decentralized), the confirmation and revocation of keys are distributed over multiple participants.

The overarching perspectives of the blockchain consensus ensure that once a name-value pair has been established, it cannot be changed without the proper authentication. More importantly, they can guarantee that the same identity cannot be issued more than once. The computational resources invested in the Proof_of_Work protocol are equivalent to votes on the correct version of the blockchain, so as long as more than 50% of the computational resources are in control of honest nodes, eventual consistency can be achieved [134].

3.4.1 Our protocol

The set of functions supported by our protocol includes:

(i) Key Generation: The algorithm generates two key pairs. The *offline* key pair is used to revoke old keys, whereas the *online* key pair is utilized to authenticate an identity. Users carry out these actions locally, save the secret key, and register the public key.

- Generate the two key pairs: the offline key pair and the online key pair.
- Store the secret key of the offline key pair securely and register the public key on the blockchain.
- When an identity needs to be verified, submit the public key of the online key pair to the blockchain.
- If the public key of the online key pair matches with the public key of the offline key pair, then the identity is verified.

(ii) Registration: Register a public key that corresponds to an identity.

(iii) Verifying: Check the relationship between a public key and a given identity.

(iv) Certification: When a user's public key is certified, the distributed network issues a certificate and delivers it to the user's client system.

(v) Updating: All key pairs must be updated regularly, that is, replaced with a new key pair, and new certificates must be provided.

(vi) Revoking: A protocol exchange may be required to support recovery if a user needs to retrieve these backed-up keys.

3.4.2 Keys generation

Users generate the key pairs offline, public and secret keys, and register the public keys on the blockchain. The attacker can access the network and participate using

Registration:

1. The participants add identities to their public keys and send the attributes to the blockchain. As input, it takes the following parameters:

   - **ID**: The identity derived from the public key, see section.
   - $\mathbf{pK}_{on}$ is the public key and π_{on} the digital signature, π_{on}=sign(sK_{on}, id) .
   - $\mathbf{pK}_{of}$ is the offline public key and π_{of} is the digital signature, π_{of}=sign(sK_{of}, id)

   Participants keep the private keys safely and request to register their identity with the corresponding public key into the blockchain after it is verified and validated by the network.

2. The miner checks:

   - The **ID** has never previously been registered.
   - The pK_{on} and pK_{of} have never previously been registered.
   - That **ver** (pK_{on}, π_{on}, id)=1, and **ver** (pK_{of}, π_{of}, id)=1. **ver** is the verification function.

3. If verified and validated by the network, the certificate's participants are then stored on the blockchain with the following verified attributes : (id, pK_{on}, π_{on}, pK_{of}, π_{of}, t), t is the timestamps.

   If any of the verification fails, discard the received block.

4. This is the registration process for all network entities.

Figure 3.1: Registration process

the stolen online key. The victim had to store the previous online key to recover from this incident. He can again calculate the stolen online key using the prior and current offline keys. He then upgrades the stolen key and revokes it. Then he revokes and updates the stolen key.

Figure 3.1 describes the registration protocol. After a user generates the key pairs locally, they store the secret keys and register the public keys.

3.4.3 Registration

Two public keys are contained in the transactions. The online key pair is associated with the first public key, and the offline key pair is associated with the second public key. The online key pair is used to validate an identity using a signature. The offline secret key is employed to revoke previous keys and sign new ones. Namecoin uses the

Update:

1. The owner sends *Update*(id, pK^{old}, pK^{new}, π_{owner}, π_{new})
   - **ID**: The identity derived from the public key, see section.
   - $\mathbf{pK}^{old}$ is the old public key.
   - $\mathbf{pK}^{new}$ is the new public key.
   - π_{owner} is the digital signature with the old secret key $\mathbf{sK}^{old}$ of the identity and the new public key π_{owner}= sign($\mathbf{sK}^{old}$, (id, pK^{new})). This proves that the owner possesses the old key pair and wants to update the public key for his **id**. The signature process contains two signatures with both online and offline secret keys.
   - π_{new}=sign($\mathbf{sK}^{new}$, id) is the new digital signature with the new secret key of the identity.
2. The miner checks:
   - The **ID** corresponds to the $\mathbf{pK}^{old}$.
   - That **ver** (pK^{old}, π_{owner} ,(id, pK^{new}))= 1.
   - That **ver** (pK^{new}, π_{new}, id)= 1.

   Otherwise, the miner omits this updated transaction.

   If any of the verification fails, discard the received block.
3. Every recipient would perform the same verification. If the verification succeeds, the updated transaction block is added or discarded.

Figure 3.2: Updating the old public key

online public key to authenticate users and servers simultaneously since Namecoin uses the trusted centralized server.

3.4.4 Revocation

In public key infrastructures (PKI), revocation is handled by using certificate revocation lists (CRLs), extending the validity of certificates, or a sophisticated combination of such mechanisms [50, 61, 65, 106]. Our protocol supports the revocation of the certificate, which is a real issue in traditional PKI systems. In our protocol, users use smart contracts to publish public, signature, and revocation keys for their identity on the Ethereum platform. The revocation function allows entities to revoke their signatures. The list of all revocations associated with a signature is stored and consulted by other entities. The signature is first checked for each revocation to ensure that the signatory matches the sender of the transaction requesting the

revocation. Certificates and their revocation status are appended to the blockchain after verifying the transaction.

The revocation process can be initiated by any arbitrary entity that is a part of the protocol. The revocation process begins when an entity needs to revoke an existing key from the ledger. This may occur due to improper or malicious behavior because the key is expired, the private key has been compromised, or for various other reasons.

The entity broadcasts a revocation request to all network nodes. A data structure known as Revocation Proof (RP), which we'll refer to as the revocation request's reason, is included in the revocation request. Before moving on to the next step of processing the revocation request, every unit in the consensus checks RP. The following are the steps for the revocation process.

Every node in the network that has received a revocation request analyzes the received RP and verifies the signature. If both are Verified, use its private key to sign the received Rp.

3.4.5 Updating a Public Key

A new public key can be generated by posting the identity in the transaction and the old and new public keys to the blockchain. A digital signature ensures that only the secret key owner corresponding to the old public key can post a new one. This update transaction will only be processed if the signature verifies with the old public key. The updating process is described in Figure 3.2

3.5 System's functioning

The smart contracts can be used to securely store and access public keys, while also providing authentication mechanisms to ensure that only authorized users can access the public keys. Additionally, smart contracts can be used to automate the exchange and validation of public keys between parties, reducing the potential for errors and malicious activity. Our protocol uses the Ethereum blockchain as a decentralized and transparent identity verification system managed by smart contracts. Any entity within the system can confirm the identities of other entities.

Ethereum addresses are unique identifiers whose ownership never changes. This make it possible to monitor and examine entities' behavior.

3.5.1 Smart contract architecture for PKI-based blockchain

A smart contract could replace a Public Key Infrastructure (PKI) by providing a secure, automated way to exchange digital assets and information securely. Smart

contracts are algorithms that are programmed to execute specific tasks when certain conditions are met. They can be used to store and transfer digital assets, such as cryptocurrencies, securely and transparently. Through smart contracts, users can securely establish trust without needing a third-party intermediary, such as a certificate authority. This could allow for a more efficient, streamlined, and secure way of exchanging digital assets and information.

By utilizing smart contracts and public/private key authentication, we create a secure and verifiable identity that is stored within the blockchain ledger. The steps of our protocol are as follows:

- Phase 1: Create a public/private key pair for each identity.
- Phase 2: Create a smart contract that is linked to the identity and that sets the rules for how the identity can be used.
- Phase 3: Store the public keys on the blockchain ledger and the private keys in a secure place.
- Phase 4: Develop an application that allows users to interact with the blockchain ledger and the smart contract.
- Phase 5: Use the public key to validate the identity of the user and access the information stored in the smart contract.
- Phase 6: Utilize a secure authentication protocol to ensure that only authorized users have access to the identity.

The protocol begins in phase (1) by creating a public/private key pair for each identity. The public key is then stored on the blockchain ledger and the private key is stored in a secure location. This ensures that only authorized individuals can access the identity information. In phase (2) a smart contract is created and linked to the identity. This smart contract has the ability to store, access and update the identity information. It can also be used to manage access rights and other security measures. In phase (3) an application is developed to interact with the blockchain ledger and the smart contract. This application will be used to verify the identity of each user and access the information stored in the smart contract. A secure authentication protocol is also used to ensure that only authorized users have access to the identity. Finally, in phase (4) the identity information is stored on the blockchain ledger and the private key is securely stored. This provides a secure and verifiable identity that is stored within the blockchain ledger. This will help to protect users from identity theft and other malicious activity.

Following these steps will ensure that we have a secure and verifiable identity that is stored within the blockchain ledger. This will help to protect users from identity theft and other malicious activity.

3.5.2 Communication Protocols

The protocol requires users to register their identities and public keys on the blockchain, and to validate the identity of the user they are communicating with. This ensures that only authenticated users are able to exchange messages, and that the messages are encrypted with the correct public key. Each user must communicate only with the user's identity validated by the smart contract and consider every other interaction as malicious.

A dapp for communication would use a blockchain to store and verify messages, as well as facilitate secure payments for goods and services. The smart contract would be used to set up rules for users, such as how messages are to be encrypted, what payment methods can be used, and any other conditions that need to be met in order for the dapp to function properly. Additionally, the smart contract would ensure that all users abide by the rules and cannot manipulate the system in any way. This would create a secure and reliable platform for users to communicate and interact with each other.

Overall, smart contracts are a powerful tool for creating secure and decentralized applications, such as WhatsApp. They can be used to ensure secure communication between users, as well as provide users with more control over their data and how it is used. Smart contracts can be used to set up a peer-to-peer network of users, where each user has their own account and is responsible for their own data. This allows users to communicate privately, without the need for a centralized server. Smart contracts can also be used to manage user accounts, ensuring that only authorized users can access the system and that all users are verified and authenticated.

The most frequently used notations in our protocol communication are in Table3.2.

1. Alice and Bob generate keys pair using the ECDSA algorithm, the public keys pK_{Alice} and pK_{Bob}, Alice's public key and Bob's public key, respectively. The secret keys correspond and derive the identities ID^{Alice} and ID^{Bob} as the public key hash. Users keep the secret key safely and request to register their identities with the corresponding public key into the blockchain after being verified and validated by the network.

   Each public key has an associated timestamp.

2. Alice signs transactions with the corresponding private key and transfers

Notation	Definition
Alice, Bob	Entities
pK_{Alice}	Alice's public key
pK_{Bob}	Bob's public key
sK_{Alice}	Alice's private key
sK_{Bob}	Bob's private key
ID^A	Alice's identity
ID^B	Bob's identity
S_{sk}^{Alice}	signature with private key of Alice
E_{pk}^{Alice}	encryption with public key of Alice
t	validity of the Public Key
T	time stamps

Table 3.2: Notations

$\mathcal{S}(ID^A, sK_{Alice}(pK_{Alice}, t, T))$ to the the blockchain.

3. The miner checks :

   (a) $pK_{Alice}(ID^A, sK_{Alice}(pK_{Alice}, t, T))$.

   (b) That the ID^A has never previously been registered.

4. If verified, The certificate's Alice $(ID^A, sK_{Alice}(pK_{Alice}), t, T)$, is then stored on the blockchain, with the following pieces of information: (Id Alice, Public key of Alice, the validity of the public key and timestamp).

5. This is the registration process for all network entities.

3.5.3 Smart contract-based verification

The smart contract is stored and executed on the blockchain. Once the smart contract is deployed, users cannot modify it since the contract was sent and validated. A user can send a transaction to the contact's address, and the transaction will be executed. Each interaction with the smart contract is recorded as a *transaction* on the blockchain. These transactions are grouped in a Merkle tree and stored in a block on the blockchain.

When user Bob wants to send a message to Alice, Bob only presents ID^B previously recorded on the blockchain and ID^A of Alice with a time-stamp T to the blockchain. Each message must include the time.

1. Bob sents a transaction: $T_B = [ID^B, ID^A, \text{T}, Sk_{Bob}(ID^B, ID^A, T)]$.

   The Smart contract receives the request from the user Bob:

2. The smart contract checks if the presented ID^B and ID^A exist on the blockchain. The Smart contract reads and parses the two Alice and Bob recordings.

   (a) It performs a Lookup of (pK, ID); if it exists, the output returns true.

   (b) Once the validity of the public keys is verified, the Smart contract checks the validity of the timestamp and the transaction's signature.

   (c) Finally, The smart contract validates the request and returns true.

3. Bob sends a transaction $T_B^1 = [ID^B, ID^A, \text{T}, E_{pub}^{Alice}(ID^B, ID^A, T, pK_{Bob})]$ to Alice's address.

4. Next, Alice checks the transaction with his private key and then sends to Bob's address $T_A^2 = [ID^A, ID^B, \text{T+1}, E_{pub}^{Bob}(ID^A, ID^B, T+1, pK_{Alice})]]$.

5. After receipt of the transaction by Bob, the same verification will be performed, and mutual authentication will be established between the two entities. 6.

6. Bob performs the same process for the registration of his identity.

7. Alice and Bob exchange their public keys and verify that they have both been registered in the blockchain.

8. Alice and Bob generate a shared secret key using a Diffie-Hellman Algorithm.

9. Alice and Bob encrypt the data they want to exchange with the shared key.

10. Alice and Bob exchange the encrypted data.

11. Alice and Bob decrypt the data using the shared key.

12. Alice and Bob verify the authenticity of the data using the digital signatures.

The shared secret can be generated using the Elliptic-curve-Diffie-Hellman (ECDH) to encrypt the Message. ECDH is a variant of the Diffie-Hellman algorithm for elliptic curves. Once the shared key is generated, it is used to encrypt messages using a symmetric key algorithm [75].

3.5.4 Key revocation smart contract

Our protocol defines expiration dates for keys and allows them to be revoked, significantly reducing the risk of unauthorized access and key theft. Figure 3.3 presents the KeyRevocation smart contract protocol, which is designed to help secure digital keys and grant access to resources or services. The protocol defines a Key struct, which consists of an owner address and an expiration date. It also defines three functions: revokeKey, isKeyValid, and expireKey. The revokeKey function allows the owner of the key to set an expiration date for the key. The isKeyValid function allows nodes to check if the key is still valid. Finally, the expireKey function allows

```
1
2
3  contract KeyRevocation {
4      struct Key {
5          address owner;
6          uint expirationDate;
7      }
8
9      mapping (address => Key) private keys;
10
11     function revokeKey(address _owner, uint _expirationDate) public
           {
12         require(msg.sender == _owner);
13         keys[_owner] = Key(_owner, _expirationDate);
14     }
15
16     function isKeyValid(address _owner) public view returns (bool)
          {
17         Key storage key = keys[_owner];
18         return (key.owner == _owner && now < key.expirationDate);
19     }
20      function expireKey(address _owner) public {
21         Key storage key = keys[_owner];
22         require(key.owner == _owner && now > key.expirationDate);
23         delete keys[_owner];
24     }
25 }
```

Figure 3.3: Key revocation

the owner of the key to delete the key if the expiration date has passed. This protocol can be used to ensure that keys are not used after their expiration date, ensuring security for the owner of the key.

3.6 Evaluation and Discussion

3.6.1 Security requirements

This protocol is an important contribution to the field of digital security, and it offers a viable solution for securely managing digital keys.

Our approach removes central authorities (CA) and uses the blockchain public as a distributed ledger of identity and their associated public keys. We use Blockchain to store public keys, digital a signature, and peer information.

Once published, the smart contract code works precisely as programmed. One of the platform's main advantages is that the code always interacts as promised, cannot be falsified, and never has downtime. The system is trust, transparent and traceable.

1. **Confidentiality:** Once the communication channel between users is secured, peer to peer encryption between endpoints can be set, and only authorized users can access the messages exchanged.

2. **Message integrity and Authentication:** The blockchain checks the validity of the signature before being stored. Another person can not change/-modify the signed agreement or alter exchanged message during the network transit. Each user has a certificate stored on the blockchain. The smart contract checks the certificate and proves the identity of the users. All exchanged messages are signed with private keys associated with the public key on certificates using the ECDSA algorithm.

3. **Reliability:** It is impossible to shut down all computers participating in the blockchain simultaneously. As a result, this database is always online, and its operation never stops.

4. **Availability**: The blockchain is resilient to denial of service attacks. Since there is no single point of failure, the blockchain is always available and functional.

5. **Nonrepudiation**: All the transactions in the blockchain are irreversible. Once a message is sent, it cannot be withdrawn or modified. This prevents a user from repudiating a transaction or denying having sent a message.

6. **Scalability**: Scaling the blockchain is possible by increasing the number of nodes in the network. This can be done by allocating more resources to the existing nodes or by adding new nodes.

7. **Security**: The blockchain is resistant to external threats since it is a distributed database. The data is encrypted and stored in the nodes in the network. Furthermore, cryptographic algorithms are used to ensure that only the sender and receiver can view the data.

3.6.2 Security requirements evaluation

The protocol provides an effective way to secure digital keys and restrict access to resources or services. It also provides a secure way to revoke keys, preventing unauthorized access. Additionally, the protocol allows for the expiration of keys, ensuring that they are not used after their expiration date. Overall, this protocol provides a secure and effective way of managing digital keys.

3.7 Open Issues

Our approach suffers from three main issues:

The first open issue of this protocol is the security of the private key. Although the private key is securely stored, there is still a risk of the private key being compromised, which could lead to identity theft or other malicious activities. Additionally, this protocol does not address privacy concerns, as the identity information is stored on the blockchain ledger. This means that anyone with access to the ledger can view the identity information, which could lead to privacy issues.

The second open issue of our protocol is that it does not provide a mechanism for preventing the reuse of expired keys. This means that an attacker could potentially use an expired key to gain unauthorized access to a resource or service. As such, additional measures should be taken to ensure that expired keys are not reused. Finally, there is currently no mechanism to automatically revoke keys that have been shared with other users. This means that users must manually revoke the key after it has been shared, which can be tedious and time-consuming.

3.8 Summary

This chapter proposes decentralized identity verification that eliminates the single point of failure of traditional PKIs and the laborious requirements for maintaining certificate authority and revocation lists. It distributes trust between entities, and new keys are registered or revoked, relying on a consensus protocol between blockchain nodes. Our protocol uses smart contracts to validate, store and revoke the certificate on a public blockchain. The individual's certificate will contain his address and public key, the address of the smart contract issued, and stored Off-chain. Only the hash is stored On-chain. Each operation is open to audit.

Our contribution benefits from an entirely decentralized architecture offering inherent fault tolerance, redundancy, and transparency. We have first proposed an approach that secures communication and its benefits of security properties of the blockchain public. It shows how to use the immutability of the blockchain to solve high problems in the field of centralized PKI.

Although the decentralized nature of blockchain-based naming brings about significant security advantages, some features of modern blockchains have technological limits. Individual blockchain entries can only carry a few kilobytes of data on average [108].

The blockchain's write propagation and leader election protocol limit the latency of creating and updating records, which is typically on a scale of 10 to 40 minutes [25]. The average bandwidth of network nodes in each round limits the number of new operations; for Bitcoin, this average currently stands at about 1500 new operations every new round [4]. Additionally, new nodes must independently audit the global log right away because it takes more time to bootstrap new nodes as the system advances.

Full nodes maintain large amounts of information to decentralize the validation system. The decentralization and scalability of the blockchain will be the focus of the following section of this manuscript so that the public blockchain may continue to manage data and store it on-chain without compromising those features.

Part III.

Sharding Protocol

Chapter 4

SECUSCA 1: the sharding storage protocol

The main question regarding the future of blockchain concerns its ability to execute a large volume of smart contracts and transactions while remaining decentralized. Indeed, with the growing popularity of traditional blockchain, each node's size has grown and become more extensive, with more than 400 GB for Bitcoin [4] and 6 TeraBytes for Ethereum, which makes it very complex for anyone to run a node. In addition, those blockchains continue to face serious scalability issues due to their ever-growing blockchain.

In order to solve these scaling issues, we propose a sharding protocol called *SECUSCA_1*. The proposed approach shows that the replication of the blockchain over the Peer-to-Peer network decreases as the blockchain's length increases to preserve security.

4.1 Introduction

Security and scalability are considered two major issues that are most likely to influence the rapid deployment of blockchains in businesses. We believe that the ability to scale up a blockchain lies mainly in improving the underlying technology rather than deploying new hardware. Though recent research works have applied sharding techniques in enhancing the scalability of blockchains, they do not cater to addressing the issue of both data security and scalability in blockchains.

Full nodes. The full nodes are the entities that independently maintain a full copy of the chain's ledger. They consistently validate new blocks and are responsible for accepting or rejecting blocks that block producers have submitted. Full nodes are not required to participate in the block productions (or Mining) process, see Figure 4.1. Most block producers run full nodes to check the chain history's validity. Even when most block producers are malicious, full nodes maintain system integrity and reject wrong blocks. If there are enough trustworthy full nodes, creating invalid blocks wastes time and resources. Full nodes store every data from the genesis block into the Proof_of_Work protocol to reach a consensus and ensure it follows the protocol rules for validity. In addition, considerable amounts of data are exchanged between blockchain nodes to synchronize or help new nodes bootstrap from the network for the first time, which is a part of the scalability difficulty.

There are two pieces of information in these data: First, the application data, which includes transactions, account balances, and smart contract [26, 140] state evolution, and everything else that is included in the block data itself. Secondly, the consensus data consists of information that makes the block header, including the Proof_of_Work [47] (or Proof_of_Stake) and nonces required to discover it. The consensus data allows us to determine the longest chain among many forks and makes consensus happen. The number of block headers that must be stored and

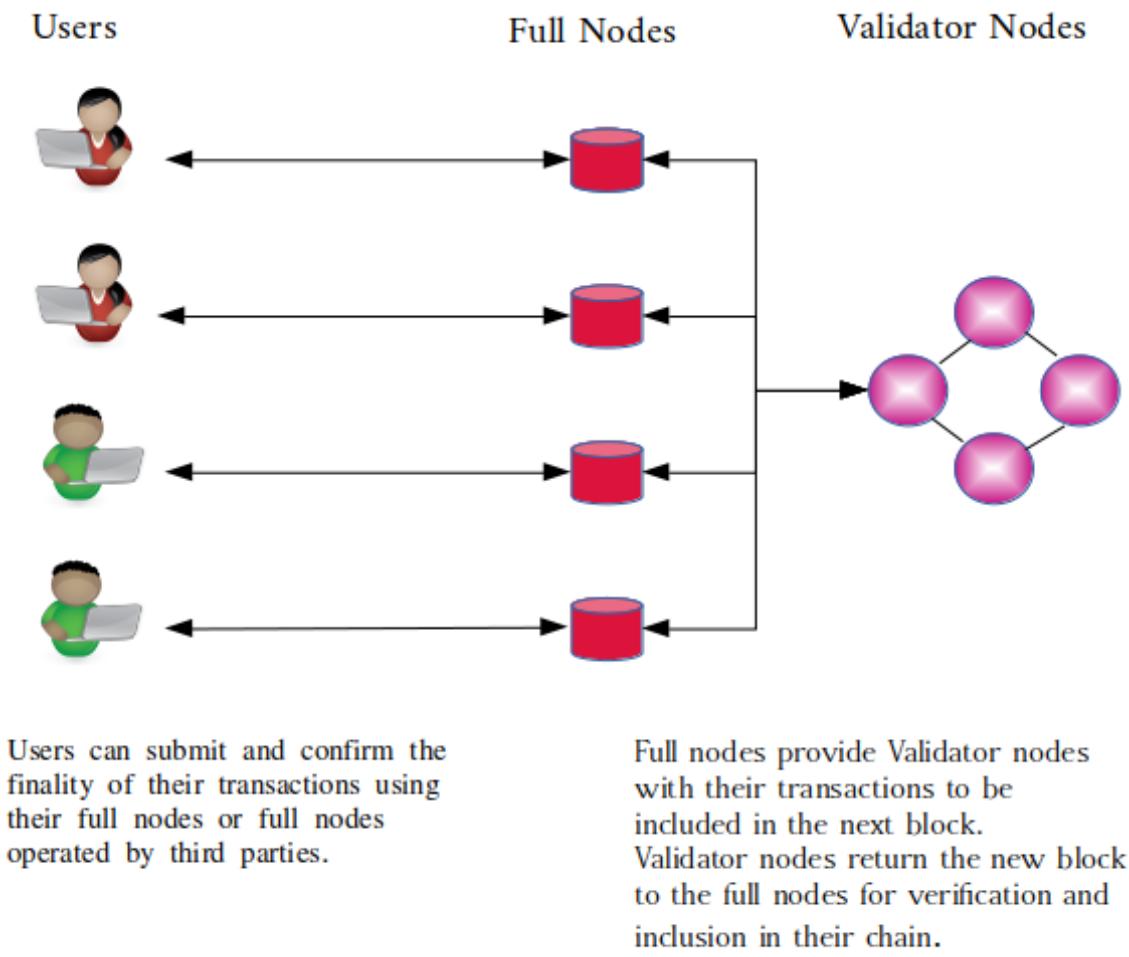


Figure 4.1: Validator Nodes send blocks to full nodes for validation and inclusion in the blockchain

sent to new bootstrapping nodes in Bitcoin increases at a constant rate of one block header every ten minutes [41], with size compared to i.e., while items can be added or removed from the UTXO [25]. Similarly, in Ethereum, adding or killing smart contracts [73] block headers still grow at a constant rate of 1 block header per 12.5 seconds.

In the present work, we focus on Proof_of_Work chains and application data (i.e., transactions). We proposed a mechanism to reduce the transaction's replication to maintain the blockchain's security without introducing any additional assumptions beyond the honest computational majority. Our protocol reduces the number of blocks that need to be stored and exchanged by nodes without losing any information. It reduces the transactions in blocks with a higher depth and keeps the block header. These reductions affect full nodes and miners alike.

Our protocol is the first to suggest that block replication will reduce without losing any information. Full nodes and miners collectively only have a small sample

of blocks. All blocks are replicated in a small set of nodes to achieve availability. The block headers are stored forever; they are necessary for achieving consensus and tracking data availability. Other solutions suggest pruning blocks but lose data availability as soon as the block is lost forever. The nodes of our protocol will store different data.

To bootstrap new nodes and reduce the amount of data exchanged, we can summarize all previous states in a block called *state block*. This technique is not very different from Proof_of_Work compression techniques that have been previously used [81, 77, 115]. This block appears every 12 blocks. This is not presented in this work.

4.2 Challenges of Scalability Storage in Blockchain

The data size challenge in scalability storage in blockchain is the need to store large amounts of data on the blockchain. First, as blockchain technology is designed to be a distributed ledger, the amount of data stored on the blockchain can quickly become large, which can lead to scalability issues, as the blockchain can become congested and slow down transaction processing. To address this issue, blockchain technology needs to be able to store and process large amounts of data without compromising speed or security. Solutions such as sharding, pruning, and off-chain storage can help to reduce the amount of data stored on the blockchain and improve scalability.

Secondly, the speed of transactions is limited by the number of nodes in the network and the amount of data that must be stored and processed. As the size of the blockchain increases, the time it takes to process transactions also increases, and the number of transactions that can be processed per second is limited, which can be a bottleneck for applications requiring a high transaction throughput. This leads to the following challenges:

1. Limited Storage Capacity: The limited storage capacity of blockchain presents a challenge to scalability. As the number of nodes and blocks increase, the storage capacity of each node decreases, leading to slower transaction processing and increased latency.

2. Processing Speed: As the amount of data stored in the blockchain increases, it can become difficult to process and verify all of the data. This can lead to a decrease in the speed of processing transactions on the blockchain, which can be a major obstacle to scalability.

3. Network Congestion: As the number of transactions on the blockchain increases, the network can become congested, leading to slower processing times

and increased fees.

4. Security Concerns: Scalability in the blockchain can lead to security concerns. As the network grows, so do the chances of malicious actors attempting to exploit the system.

4.3 Scalability storage in blockchain

Scalability storage in the blockchain is the ability for blockchains to increase their data storage capacity as the need for storage increases. This is achieved by different techniques such as sharding, off-chain storage, and data compression. These techniques allow blockchain networks to store more data without sacrificing performance or security.

Sharding is a technique for scaling out a blockchain network by breaking the network into multiple smaller parts, or shards, that can process transactions in parallel. Each shard essentially acts like its own mini-blockchain, with its own set of transactions and its own consensus mechanism. For example, if a blockchain network has 10 shards, each shard can process 10 transactions at the same time, allowing the blockchain to process 100 transactions in total in the same amount of time. This makes it much easier for the network to scale up its transaction throughput as the network grows.

Off-chain storage is a way of storing data outside of the blockchain to increase scalability. This means that instead of having to store large amounts of data on the blockchain, which could slow it down, this data is moved off the chain. An example of off-chain storage is a distributed file system such as IPFS (InterPlanetary File System). IPFS is a distributed file system that allows users to store large amounts of data off the blockchain. Users can then access this data on the blockchain, increasing the system's scalability.

Data compression is a technique used to reduce the amount of storage space needed to store data in a blockchain network. By compressing data, the blockchain network can reduce the amount of storage space required to store the data, resulting in improved scalability and performance. It also reduces the amount of data that must be transferred over the network, which can reduce costs associated with network bandwidth. An example of data compression for scalability storage in blockchain is the use of a Merkle Tree. A Merkle Tree is a data structure that stores the "fingerprints" of the individual transactions in a blockchain network. By using a Merkle Tree, the blockchain network can store the data much more efficiently, reducing the need for storage space.

4.4 State of the arts

This section will provide an overview of the latest developments in blockchain scalability and the challenges that have been identified in the field. We will discuss the various solutions that have been proposed, their strengths and weaknesses, and the potential implications of these solutions for the future of blockchain technology.

A long line of research proposes various techniques to make blockchain more scalable. For example, On_chain approaches, i.e., sharding nodes into multiple subsets [87, 98, 147], and Off_chain approaches, i.e., Lightning Network [66]. Partitioning data into separate shards managed by different subsets of nodes reduces performance as more messages are exchanged to build consensus without improving robustness; in such an approach, the data grows linearly with the number of nodes and transactions, the Off_chain scalability solutions can introduce their security threats to the blockchain because the data is not stored directly on the blockchain but rather through third-party protocols. These techniques provide scalability but affect decentralization and result in security vulnerabilities.

Table 4.1: Comparison between existing sharding blockchains in academia and industry based UTXO model and Account model

	UTXO			Account		
System	Elastico	Omniledger	Rapidchain	Ethereum 2.0	Monoxide	Zilliqa
Intra-Shard Consensus	PBFT	BFT (BizCoinX)	Sync BFT	BFT	POW	PBFT
Adversary Model	$\leq \frac{1}{4}$	$\leq \frac{1}{4}$	$\leq \frac{1}{3}$	$\leq \frac{1}{4}$	$\leq \frac{1}{2}$	$\leq \frac{1}{4}$
Cross-shard Consensus	-	2PC	Split	RT	RT	-
Performance	O(n^2)	O(n)		O(1)/O(n)	O(n)	O(n)
Decentralized	PBFT	BFT	Sync BFT	BFT	POW	PBFT
Security (Fault tolerance)	$3f+1$ 33%	$3f+1$ 33%	$2f+1$ 50%	$3f+1$ 33%	$2f+1$ 50%	$3f+1$ 33%
Scalability	No	No	Yes	Yes	No	No

Note:
[a]RT: Relay Transaction.
[b]The notation "-" means that the property does not apply to the system
[c]f is the number of admissible failures.

4.4.1 Off-chain blockchain

Executing transactions off-chain is another way to overcome the scalability limitations of blockchains. Untrusted peers can build direct payment channels on the Lightning network [22], allowing them to make off-chain micropayments without committing each transaction to the underlying blockchain. A payment channel comprises a blockchain-based contract that stores the funds of the peers involved in the transaction. Signatures are used in micropayments to ensure that peers accept a transaction. Furthermore, hash locks and timelocks are used in micropayments to ensure that a malicious node does not take advantage of a cooperative participant. Off-chain scalability options may come with their own set of security problems for the system; the decentralization and security of blockchain networks should not be compromised. To maintain the security of blockchain applications and the continuing growth of the blockchain architecture, scalability features must be applied at the base layer level.

4.4.2 Sharding

Generally used in databases, is proposed in cryptocurrency ledger. Sharding is the portioning of a network of N nodes into committees k with a small number of nodes c, with replication, $c = N/K$ - i.e., to yield smaller full replication systems. The node in each committee K stores only validated blocks inside its committee and does not manage the entire blockchain ledger. This allows the network to process transactions more efficiently by spreading out the workload over multiple nodes and used to increase throughput. As an example, [99, 88, 148] are sharding-based Proof_of_Work and Byzantine fault tolerance (BFT) [144].

i. Elastico

In [99] is the first sharding-based public blockchain proposed in 2016 that tolerates byzantine adversaries. It partitions the network into shards and ensures probabilistic correctness by randomly assigning nodes to committees, wherein a disjoint committee of nodes in parallel verifies each shard. It executes expensive PoW to form a committee, where nodes randomly join different committees and run PBFT [28] or Practical Byzantine Fault Tolerance for intra-committee consensus. In Elastico, all nodes maintain the blockchain ledger, but cross-shard transactions are not supported. In addition, running PBFT among hundreds of nodes decreases the protocol's performance, but reducing the number of nodes within each shard increases the failure probability. The network can only tolerate up to 25% of malicious nodes.

ii. Omniledger

In [88] improved upon *Elastico*. It includes new methods to assign nodes into shards with a higher security guarantee, as *Elastico*. It uses proof-of-work and BFT, an atomic protocol for across-shard transactions (Atomix). The intra-shard consensus protocol of OmniLedger uses a variant of ByzCoin [86] and assumes partially synchronous channels to achieve faster transactions. The network tolerates up to 25% of faulty nodes and 33% of malicious nodes in each committee as in [99].

iii. Rapidchain

Cross-shard in *Rapidchain* [148] relies on an inter-committee routing scheme which is based on the routing algorithm of Kademlia [102]. Rapidchain [147] also supports cross-shard transactions using Byzantine consensus protocols but requires strong synchronous communication among shards which is hard to achieve with resilience up to 33% and 50% of committee resiliency. In sharding-based systems, database performance scales linearly with the number of nodes, necessitating the creation of complicated protocols to enable shard connectivity.

Other approaches in the literature are based on private blockchain [129, 128, 16, 37, 15]. Even though sharding improves storage and throughput, K increases linearly with N with a low-security level, thus, leading to malicious node errors.

4.4.3 Pruning Blocks

Pruning is a method used to reduce the size of a blockchain by removing older blocks from the chain and storing only a portion of the most recent blocks. This method is used to reduce the amount of disk space and bandwidth needed to store and transfer the data. Pruning can also be used to improve data availability by ensuring that only the most up-to-date blocks are available for use in transactions and applications. By removing blocks that are no longer needed, the most recent blocks are more readily available and can be used more quickly, helping to reduce latency and improve the overall performance of the network.

Mbinkeu et al. [109] looked into the memory management and access time of the Bitcoin protocol, which uses SQLite databases. A memory optimization approach based on a redundancy system is developed by Guo et al. [63] to reduce the storage capacity of each node. It suggested a redundancy-based optimization strategy that significantly reduces the storage capacity of blockchain system nodes and creates a fault-tolerant mechanism. Wang et al. [14] looked at distributing data across a blockchain network. This work proposed a balanced user input solution for search time and space occupation. Gennaro et al. [56] developed a centralized threshold signature mechanism for more efficient Bitcoin systems in light of the challenges

with Bitcoin key management. El-Hindi et al. [48] added a database layer to the blockchain system to increase the performance and scalability of data sharing.

Non-interactive proofs of Proof_of_Work (NIPoPoW), a recent approach suggested by Kiayias et al. [82]., permits a light client to download and store just a polylogarithmic amount of block headers in expectation. Unfortunately, NIPoPoWs can only be employed in chains with fixed block difficulty and are succinct as long as no attacker impacts the honest chain. This is in contrast to most cryptocurrencies, which frequently modify block difficulty according to the network hash rate.

4.4.4 Mina

Mina [24] developed by O(1) labs is considered the smallest blockchain. It integrates zero-knowledge proofs ("succinct non-interactive argument of knowledge" or "Zk-SNARKs") to validate transactions, which were first used by the Zcash cryptocurrency [20]. The protocol generates a proof at each step to validate a new transaction without consulting the register of all the previous transactions; this proof drastically reduces the size of the data each user and validator need to download, which is never more than 22 KB. While validators need only a small amount of data, the block producers responsible for creating new blocks store the full state to achieve consensus. So Mina still has a storage problem.

4.4.5 Others Propositions

Vault [94] introduces fast bootstrapping to allow new participants to join the network without downloading the whole blockchain by reducing the transmitted state. *Vault* is Account-based for Algorand [29] and does not require all nodes to store the whole blockchain state.

In [95], authors propose a superlight client design to allow a light client to relay full nodes to read blockchain with a low read cost to predict (non) existence of a transaction in a blockchain. Therefore, blockchains can hold a large amount of data. However, each node requires storage space. Thus, the cost of storage and the required memory increase with the number of transactions.

4.5 SECUSCA_1 approach

4.5.1 Motivating example

The blocks that make up the blockchain are replicated on all nodes. They maintain local copies of all blocks (including the genesis block) for the following reasons:

1. To verify a transaction, the nodes read the history of all past transactions locally.

2. To provide replication as it enhances security against attacks and tampering and improves data availability.

3. To safeguard transactions - transactions are considered sufficiently safe from attacks when buried under enough blocks, and miners reach a consensus by selecting the longest. In proof_of_work cryptocurrencies, the longest chain is deemed honest by the network, regarded as the most invested chain

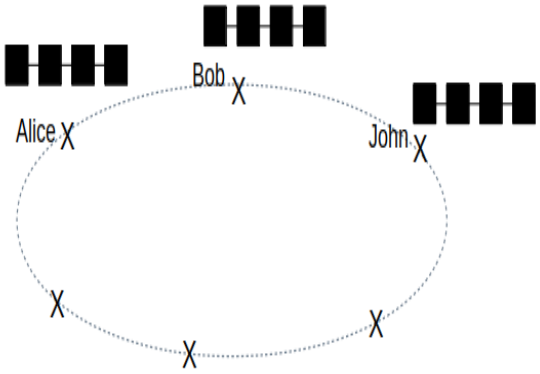
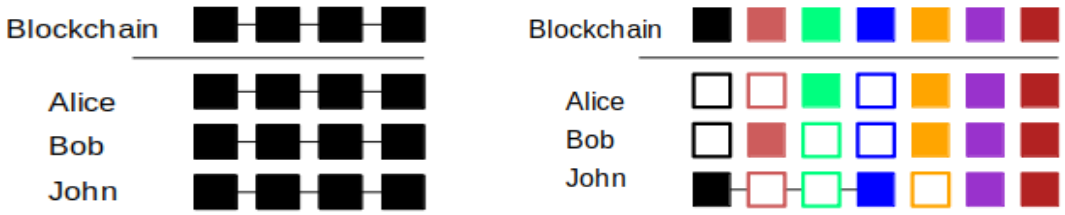


Figure 4.2: Full Replication

We consider Alice, Bob, and John as three participants among hundred nodes in the blockchain network with a storage capacity of 50 GB that holds a shared ledger. As shown in Figure 4.2, the blockchain is fully replicated on every node in the network. All nodes store whole blocks with all transactions, and the same block is replicated on all nodes.
Suppose that the size of a block is $1MB$ and that blocks are generated every $10minutes$, $1MB * 6 * 24 * 30 = 4320MB$ per month. Since each block is replicated in all nodes, the three nodes can store up to $50 * 10^3$ blocks. These nodes with a capacity of $50GB$ containing the blockchain will be saturated in less than a year. This shows that the current blockchain design would result in major scalability issues.

This example is demonstrating how blockchain distributed over networks is more scalable in storage than full replication. By reducing replication, it reduces the storage space of nodes so that they can continue to receive new blocks, allowing for a blockchain to contain more than 50,000 blocks.

Reducing replication reduces the storage space of nodes so they can continue to receive new blocks. Thus blockchain can contain more than 50,000 blocks. As shown



(a) Traditional blockchain (b) New replication model

Figure 4.3: Traditional blockchain *vs* the new replication model. In 2(a), each node maintains its chain which contains all previous transactions. Each given block is stored on the three nodes, Alice, Bob, and john. While in 2(b), the global blockchain shown at the top is shared across the three nodes. Colored blocks represent the entire block containing transactions, and framed blocks only have the block header - so the block buried in the chain is held on a few nodes.

in Figure 4.3, blockchain distributed over networks is more scalable in storage than that full replication.

4.5.2 System's functionning

We consider a network of peers. A peer generates operations in order to modify local data. These operations are executed immediately by the peer and then broadcast through the P2P network to all other peers using an anti-entropy protocol [69], such as epidemic propagation [51]. This protocol ensures that the new operation is disseminated throughout the network, even if there are process crashes, disconnections, packet losses, or changes in the network topology.

Data replication in the blockchain is a protocol that ensures that every transaction and block are recorded on each node in the network. It involves verifying and validating new transactions before broadcasting them to all other nodes for verification and validation. Once all nodes have validated the transaction, it is recorded in a new block and added to the chain. This new block is then replicated across all nodes of the network, allowing all users of the network to have access to an exact and up-to-date copy of the blockchain.

Quorum voting is a protocol for deleting data in a proof-of-work blockchain. It requires more than half of the network members to vote to approve the action, and once the quorum is reached, the network will need to be updated to reflect the change. This is done by miners who will check the quorum and incorporate the change into the network. To ensure that each node is deleting different information, a reference table should be created to list the information and indicate which node has deleted which information. Then, whenever a node needs to delete information, it must first check the reference table to see if another node has already deleted that information. If so, it can choose other information. If not, it can delete that

information. Lastly, consistency in distributed ledger technology is achieved through a consensus mechanism, which requires all nodes to validate each transaction with a high amount of computational power. When a valid block is discovered, it is broadcasted to all other nodes in the network and if accepted, it is added to the ledger. This process is repeated for each new block of data, ensuring that all nodes have an identical, up-to-date ledger.

Consistency in a peer-to-peer system is maintained by having each peer periodically exchange its data with each of its peers. This ensures that all peers in the network have the same data and that new data is distributed to all peers in a timely manner. To ensure data integrity, peers must also use digital signatures to verify the accuracy of data being exchanged. Finally, peers should also employ techniques such as conflict resolution and synchronization to ensure that data is consistent across the system.

Miners in blockchain can reduce the replication of blocks to save data by using technologies such as sharding, which allows for the creation of smaller blockchains linked to the main blockchain. This allows for more efficient storage of data by avoiding unnecessary duplication. Additionally, miners can use data compression techniques to further reduce the amount of data that needs to be stored in the blockchain.

A full node in the blockchain can reduce the replication of blocks to save memory by pruning the blockchain. Pruning is the process of deleting blocks from the blockchain that are not required for the node to operate. Pruning removes unnecessary data from the blockchain, which reduces the memory needed to store the blockchain. This can help to reduce the size of the blockchain and improve its performance.

One way full nodes in the blockchain can reduce the replication of blocks to save memory is by pruning the blockchain. Pruning is a process by which a node removes old blocks from its local copy of the blockchain and keeps only the most recent blocks that have not yet been confirmed by the network. This allows the node to save storage space without sacrificing security. Pruning also reduces the amount of data that needs to be propagated across the entire network, which makes the system more efficient.

4.6 The dynamic sharding approach

The SECUSCA_1 protocol of sharding is a method of distributing blocks across the network. It uses a replication function to determine the number of nodes that should store a particular block. In addition, it allows the network to scale more efficiently by reducing the amount of data that needs to be stored in order to validate transactions.

4.6.1 Overview of SECUSCA_1

The SECUSCA_1 protocol is an optimization technique in which old data is removed from the chain in order to reduce the blockchain size. This is done by removing the transaction data and only keeping the block headers. This protocol helps to improve the scalability of the blockchain by reducing the size of the chain, while still maintaining its security. It also helps to reduce the amount of storage space required by nodes to store the chain.

The process comprises two steps that operate at the time. An optimization function is introduced to ensure the sharding process runs smoothly. This function comprises two steps that operate simultaneously:

- *Efficient replication*: The Efficient Replication Protocol consists of distributing the global state of the blockchain to network nodes by dynamic sharding. The main goal is to preserve the security and scalability of the blockchain and store data. A node can participate in the protocol process even with small capacity. To ensure scalability, not all nodes store the full state of the blockchain. Instead, the blockchain is distributed among more nodes, each node being represented by n_i (where $i = 1, ..., N$). Blocks are propagated across the network and stored on nodes n with ($n <= N$). A node does not need to store any state to participate in transaction validation and block processing. The maximum number of malicious nodes q that the system can tolerate is less than half the number of nodes in the network. For honest nodes p to be resistant to malicious nodes q, their computing power must exceed the computing power of malicious nodes. Therefore, the security of the blockchain is guaranteed by honest nodes, which is why the replication of blocks must be large enough at the beginning of the process to ensure that malicious nodes cannot attack the network. The replication of the block also depends on the size of the blockchain in each node n_i. At the beginning, replication is at its maximum and decreases as the blockchain size increases.

- *Efficient reduction*: the block replication reduction protocol in the SECUSCA_1 reduces block replication by only storing the header of all blocks in the chain, but not the entire block. When a new block is added, its transactions are only confirmed when a specific size of other blocks is reached after it. At this stage, the full block is stored. When the block is confirmed by the network and buried in the blockchain, the transactions are deleted. As the blockchain size increases, the replication of the old blocks decreases, but the latest blocks remain at a high replication. This approach selects replication of all previously confirmed blocks in the blockchain across all nodes, with each node freeing

up memory space by erasing the transactions from these blocks until a minimum replication is reached. After this, replication remains constant. With SECUSCA_1, the availability of the global state is ensured and a minimum replication of the entire blocks is distributed among the nodes. Each node stores a part of the blockchain state (a subset of the full history) and block headers, including Merkle roots for the whole blockchain.

- *Liveness and availability*: Despite the state sharding between nodes, the blockchain continues to operate even when some nodes delete blocks from their local chain.

4.6.2 Block sharding protocol

This section will discuss the optimization function. We will explore the function's implications and its potential to improve blockchain scalability. let's introduce some useful definitions used in the setting of the function. We first introduce the notion of the depth of a block which is given by the number of blocks added to the chain after it. It is formally defined as:

Definition 4.6.1 ***(Block depth** d**).*** *Let's consider, t, the size of the blockchain, and j, the position of a block b_j in the chain, we define the depth of a block b_j as follows: $d = t - j$.*

For example, if a blockchain contains three blocks [b1, b2, b3], the depth of b1 is 2, the depth of b2 is 1, and the depth of b3 is 0.

Definition 4.6.2 ***(The boundaries thresholds** $\alpha_i N, \alpha_f N$ **for security.)*** *Let's consider, N, as the number of nodes in the network that host the blockchain,*

- *the upper bound $\alpha_i N$ in the replication phase is the estimated highest number of nodes where the blockchain should be fully replicated in all nodes such as $\alpha_i N < N$ to ensure the maximum security.*
- *the lower bound $\alpha_f N$ in the replication phase is the estimated lowest number of nodes where the blockchain should be replicated such as $\alpha_f N < \alpha_i N < N$.*

Definition 4.6.3 ***(The boundaries thresholds** $\gamma_i B, \gamma_f B$ **for scalability.)*** *Let's consider B as the higher size of the blockchain. In order to ensure security and optimize scalability we define:*

- *The upper bound $\gamma_f B$ in the replication phase is the estimated highest depth of a block from which we cannot go any lower to prevent the reduction of the blocks in different nodes, where $\gamma_f B < B$. B is the blockchain size.*

- *the lower bound $\gamma_i B$ in the replication phase from which the replication starts to be reduced, such as $\gamma_i B < \gamma_f B$.*

By using the definitions of block depth in a blockchain and the boundaries, let's define the sharding function:

Definition 4.6.4 ***(The sharding optimisation function $\mathcal{R}$.)*** *Let d, be a depth of block b in a blockchain. According to the steps of SECUSCA_1, the number of replications of any block $\mathcal{R}(b)$ over is defined as:*

$$\mathcal{R}(b) = \begin{cases} \alpha_i N & \text{if } d \leq \gamma_i B \\ \alpha_i N + \frac{\alpha_f N - \alpha_i N}{\gamma_f B - \gamma_i B} * (d - \gamma_i B) & \text{if } \gamma_i B \leq d \leq \gamma_f B \\ \alpha_f N & \text{if } \ d \geq \gamma_f B \end{cases}$$

The function defines the number of replications of any block in the blockchain depending on its depth in the chain, as well as the boundaries thresholds for security ($\alpha_i N$ and $\alpha_f N$) and scalability ($\gamma_i B$ and $\gamma_f B$). The function restricts the replication of the blocks in order to optimize the scalability of the network while still preserving the security of the blockchain. $\alpha_i N$ sets the upper and lower bounds for replication and $\alpha_f N$ respectively, while the upper and lower bounds for the blockchain size are set by $\gamma_i B and \gamma_f B$ respectively. The protocol then uses the optimization function to determine the number of replications of each block based on the depth of the block. The function results in a replication of $\alpha_i N$ if the depth of the block is less than or equal to $\gamma_i B$, a replication of $\alpha_i N + \frac{\alpha_f N - \alpha_i N}{\gamma_f B - \gamma_i B} * (d - \gamma_i B)$ if the depth of the block is between $\gamma_i B and \gamma_f B$, and replication of $\alpha_f N$ if the depth of the block is greater than or equal to $\gamma_f B$. By setting these parameters and using the sharding optimization function, the protocol is able to optimize scalability and security.

4.7 SECUSCA_1 algorithms

The key idea of SECUSCA_1 is to share the blockchain over nodes. We outline the two main replication and deletion algorithms according to the replication function in 4.6.4.

4.7.1 System model

We assume a finite group of participants Π whose composition may change over time. Although the clocks of the users are not synchronized, they all drift simultaneously. Participants communicate within the system by sending messages. They can have the roles of *miner*(m) that upload the block and *node validator*(N) that download the block. We assume that the cardinality of the latter node validators

is always finite. We assume that the network is *partially synchronous*, i.e., There is a predetermined maximum network delay Δ in this network, which is unknown to network participants; however, they are sure that all messages will eventually reach their recipients. *partially synchronous* offers good adaptability to the natural network dynamics while simplifying network modeling. Participants have access to essential cryptographic functions, including a cryptographic hash function and a signature scheme to sign all messages they send, and we assume that signatures are not forgeable.

The miner creates and broadcasts the new block in a quintuplet message containing the block's hash, target replication, and other information, signed with their private key. Nodes that receive the message verify the hash and replication and, if they have enough capacity to store the block, they send a message back to the miner. The miner then collects responses from the network after a delay and sends the block to randomly selected nodes. All nodes in the network process the block's replication for all blocks in their local blockchain. Once a block is cryptographically secure, its replication decreases. Once the block is deeply secure and its depth is greater than the depth defined by the function, its replication decreases.

New block is distributed over α_iN set of nodes:

1. Step 1: given the target replications from 4.6.4. A miner creates a new block b with a depth of $d_b = 0$, and target replication of $\mathcal{R}(b)$=$\alpha_i N$. The miner needs to connect with the network and prepares to upload the new block.

   (a) Message initialization: Before the miner transfers the block to the downloader, it broadcasts the new block hash $b.hash$ in a quintuplet message $(inv, pK, b.hash, target\_replication)$ signed with his private key sK, and starts the time parameter Δ.

   (b) Downloaders N receive the message and verify $b.hash$, then target replication $\mathcal{R}(b)$, i.e., even for the previous blocks if there are blocks in the chain. If chosen nodes have enough capacity to store the block, they send a message individually $(echo, get\_block(b))$ to the miner telling him to upload the whole block.

2. Step 2: Miner receives a response from the network. and collects them after an expired delay Δ. The *get_block* message is used in the Bitcoin broadcast [39]; broadcasting only the block header first reduces the flow and avoids sending the complete block to the whole network, given that the message crosses the entire network, and avoids sending the same block several times to the same node if a node has not received the block, it responds with an *echo message*. Otherwise, it is considered that the node has already received the block.

(a) Miner receives $(echo, get\_block(b))$ from the network. It selects randomly a number of nodes and sends them the block b.

(b) The nodes that store the block b communicate the $b.header$ to the other peers as proof that they get the right block from the miner.

3. Step 3: All nodes in the network process $\mathcal{R}(b)$ for all blocks in their local blockchain. Once a block is deeply secure, and its depth becomes greater than $\gamma_i B$, see 4.7.3.

4.7.2 Storage algorithm

This algorithm is used to store new block, it looks through the network for nodes with enough space to store the new block. If a sufficient number of nodes with enough space is found, then the new block is stored on those nodes.

Algorithm 1 Replication protocol

Input:
b: a block received from the network
$Network = \{n_1, n_2, ..., n_i\}$: the nodes of the blockchain

```
remaining_storage_capacity= storage_capacity - stored_blocks_size
if validate(b_j) then
    for n in Network do
        if remaining_storage_capacity>=new_block.size(b_j) then
            return list.nodes_with_enough_space
        else
            "Not enough storage to continue the protocol"
        end if
    end for
    if nodes_with_enough_space >= target_replication then
        Select nodes randomly for storage (Network, target_replication)
        for node in random_nodes do
            node.store_block(b_j)
        end for
    end if
    BC ← BC + {b_j}
end if
```

Initially, the block replication is comprised between the two targets $\alpha_i N$ and $\alpha_f N$. At each time, the block replication can be adjusted in two cases:

- If one node or more join the network, the replication $\mathcal{R}(\lfloor)$ is adjusted, and the replication increases.

- If one or more nodes go down or leave the network, in this case, if these nodes store blocks, then these blocks will be stored by other nodes.

4.7.3 Deletion algorithm

Recall that the block comprises a header and the body containing the transactions. In our reduction protocol, only the body of a block is deleted. To reduce the number of blocks, the protocol deletes a replica of each block having a depth of each epoch. (e.g., every 10 minutes corresponds to creating and adding a new block).

Algorithm 2 Deletion protocol

Reduce blocks: Each block has an identity and an index in the blockchain. The function reduces block replication until the final replication of a block, defined by the replication function.
Input:
nodes storing blocks(block.identity)

```
node= Select randomly node from nodes_storing_blocks(block.identity)
actual_replication = len(nodes_storing_blocks[block.identity])
if actual_replication > target_replication: then
    delete_block(block)
else if elif actual_replication < target_replication: then
    store_block(block)
end if
```

4.8 Simulation

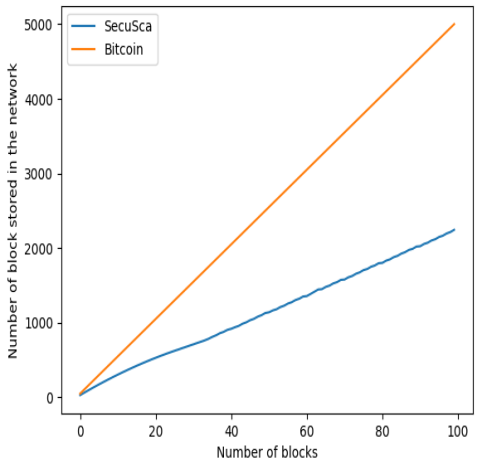
The code for the algorithms of SECUSCA_1 in Python is available on GitHub at "github.com/khacefkahina" [7] to validate how our proposed SECUSCA_1 scales out the blockchain technology.

We evaluate our approach and compare it with traditional blockchain (e.g., Bitcoin) by varying the parameters of the replication function. We run multiple simulations with different values of the α and γ parameters in order to define the *upper bound* $\alpha_i N$ and the *lower bound* $\gamma_f B$. We then determine the size of the whole blockchain shared over the network.

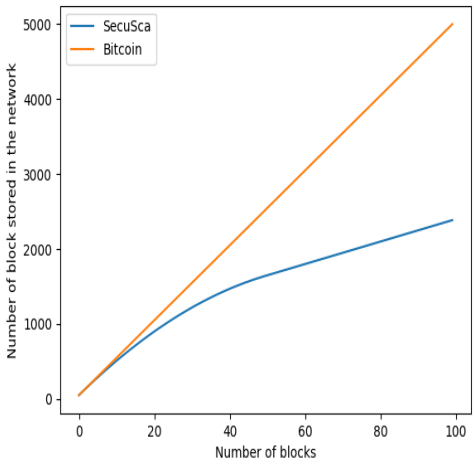
4.8.1 Blockchain Size Analysis

.

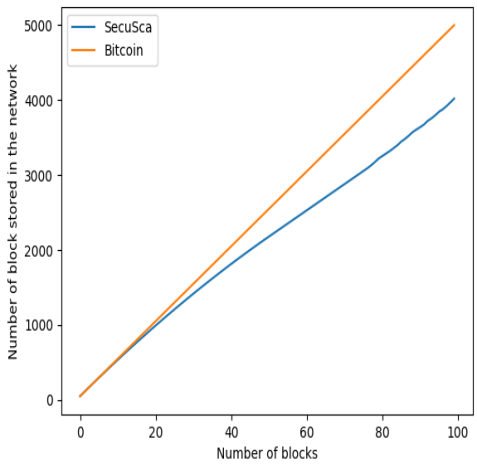
We run the experiment on generating 100 and 200 blocks of size 0.5 with 50, 100 nodes having capacities increasing linearly from 50 to 2500, with different parameters



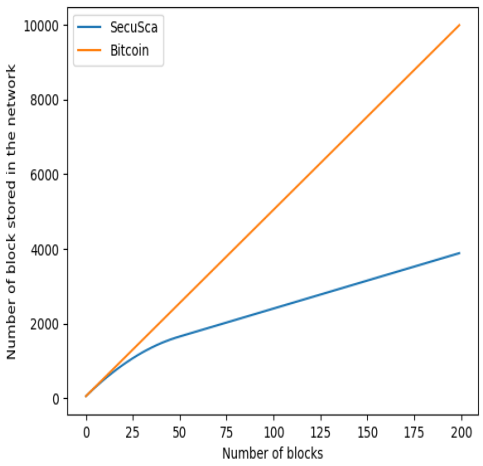
(a) 50 nodes, $\alpha_i = 0.6$, $\alpha_f = 0.1$, $\gamma_i = 0.5$, $\gamma_f = 0.9$



(b) 50 nodes, $\alpha_i = 1$, $\alpha_f = 0.1$, $\gamma_i = 0.3$, $\gamma_f = 1$



(c) 50 nodes, 100 blocks, $\alpha_i = 1$, $\alpha_f = 0.1$, $\gamma_i = 0.7$, $\gamma_f = 0.9$



(d) 100 nodes, 200 blocks, $\alpha_i = 1$, $\alpha_f = 0.1$, $\gamma_i = 0.3$, $\gamma_f = 1$

Figure 4.4: Number of blocks stored by SECUSCA_1

values for the target replication: α_i, α_f, γ_i, and γ_f. Each node performs the sharding optimization function $\mathcal{R}$.

Figure 4.4 gives an overview of the evolution of the blockchain size in both Bitcoin and SECUSCA_1. The experiment reveals that the overhead of blockchain becomes significant in Bitcoin, traced in red. The size of SECUSCA_1 is the sum of all blocks stored on each node, from the first to the last block, traced in blue. The size grows linearly.

We observe that the ratio of storage quickly converges to a limit. This comes from the fact that for the level of storage of each block is an homogeneous function of the depth of the block. (e.g. the replication ratio of the 60th block when the height of the blockchain is $H = 100$ is the same than the replication ratio of the 120th block

when the height of the blockchain is $H = 200$).

Therefore it is possible to select the parameters α_i, α_f, λ_i and λ_f with a target memory saving, and targets for initial storage security α_i and the "cold blocks" security α_f.

.

4.9 Summary

In this chapter, we have proposed a new blockchain design that makes a trade-off between security and scalability that allow for more capacity in terms of storage in the whole blockchain. We present a new blockchain design to reduce the storage volume on each node and allow dynamic sharding with different local states from node to node but without compromising on security properties.

Compared to the related work, SECUSCA_1 is the best solution for saving storage in a blockchain depends. It reduces the amount of data stored on the chain by removing old data, while sharding divides the chain into smaller pieces or shards to spread the load of processing and storage. Both of these approaches can help to reduce the storage requirements of a blockchain, however, sharding comes with several challenges. Firstly, it requires a complex consensus mechanism to ensure that all shards remain in sync. Secondly, it is difficult to ensure that all nodes remain secure in a sharded network. Finally, transactions can be vulnerable to a variety of attacks, such as Sybil attacks, double-spend attacks, and malicious data insertion attacks.

Pruning protocols are designed to reduce the size of the blockchain by removing blocks that are no longer necessary for validation. However, they may also reduce the overall security of a blockchain, as they can potentially remove data that is necessary for verifying the integrity of the ledger. SECUSCA_1 proposes preserving a minimal replication of data to preserve integrity. This way, if a node is compromised, the data can still be verified by the other nodes in the network.

The above analytical and numerical results show how our proposed approach promotes scalability and enables users to store more transactions by freeing up local disks. Nevertheless, SECUSCA_1 needs further improvements. For instance, it should allow the blockchain to continue functioning even when some nodes are saturated.

Chapter 5

SECUSCA 2: Optimized version of SECUSCA 1 to improve decentralization

5.1 Introduction

We present in this chapter our architecture for blockchain with stateless transaction validation. In SECUSCA_*, miners and validating nodes process transactions and blocks simply by accessing a short commitment of the current state found in the most recent block. We present an instantiate of SECUSCA_1 in the UTXO model that uses the ranking of nodes.

Instead of randomly choosing network nodes that have available memory to receive the new block, SECUSCA_2 favors nodes with large free memory. The difference between the two proposals consists of the distribution of the blocks. We will demonstrate in the 5.4, the fairness of SECUSCA_2

Bitcoin and Ethereum support light clients [108] [140]; however, they can only carry a small set of state and ledger subsets and cannot participate in consensus. By initially building a complete state data structure and then removing all but the last n ledger blocks, Bitcoin Core [1] and comparable projects [40] [30] maintain the capacity to validate transactions at the cost of increased disk capacity. However, these methods fail because they do not enable fine-grained control over which transactions to maintain and do not permit meaningful data structures to be formed from partially pruned blocks.

5.2 SECUSCA 1 limitations:

This section discussing the scalability of SECUSCA_1, a blockchain sharding protocol, and how it is perceived as unfair by randomly sampling nodes without knowledge of their characteristics. It also discusses how small nodes can be quickly saturated by the protocol and how this leads to centralization around large nodes.

SECUSCA_1 is a proposed blockchain scalability solution through sharding, which suggests a replication strategy based on the length of the chain. It attempts to ensure fairness in the selection of nodes by employing a random sampling of nodes with an equal probability of being chosen. However, our research demonstrates that this protocol is actually unfair, as nodes with small storage capacities can be quickly saturated if they are regularly chosen to store blocks, leading to a centralization around nodes with large storage capacities. SECUSCA_1 allows nodes with large storage capabilities to maintain the blockchain while discouraging small nodes from participating in the protocol. Furthermore, the random selection of nodes can lead to a rapid saturation of nodes with small storage capacities. Consequently, SECUSCA_1 does not effectively promote decentralization, which is vital for a successful blockchain protocol.

5.3 An optimal algorithm for random sampling without replacement

SECUSCA_1 is an approach to improve blockchain scalability by sharding the blockchain to nodes. It proposes a replication strategy that adjusts the number of replicas based on the block's depth in the chain. However, it has been shown to be unfair as it does not take into account the characteristics of each node, such as storage capacity, which can lead to centralization around large nodes. SECUSCA_2 is a fair alternative that tracks the capacity of each node and allocates the number of block replicas accordingly. This prevents the selection of nodes with insufficient capacity and ensures the target replication is achieved.

SECUSCA_2 is a protocol designed to ensure that blocks are stored in a fair and efficient manner. It achieves this by calculating a record of nodes' capacity and samples $0 < \alpha N \leq N$ nodes without replacement from the network of N nodes. to randomly select nodes with the required capacity for storing a block. This ensures that no node is overwhelmed by the amount of data it is required to store and that the desired replication level is always available. Furthermore, this approach prevents nodes with small capacities from becoming saturated and unable to accept blocks.

$\sum \tilde{C}_{n_j}$ is the total remaining capacities for all the nodes. $\tilde{C}_{n_i}$ is the capacity of the node n_i remaining after storage of the blocks. This capacity is calculated as the difference between the total capacity of the node and the size of the blocks already stored in the node, i.e. Let's take two blocks, b_j and b_{j+1}, mined in two time stamps t_j and t_{j+1}, and stored by node n_i. The remaining size of this node, Cni at time t_{j+1}, is given by $C_{n_i} = C_{n_i}(t_j) - Size.b_{j+1}$; we formulate the remaining storage capacity of a node by subtracting the size of the last block mined in the network:

$$\tilde{C}_{n_i} = C_{n_i} - \sum_{block \text{ stored}} block.size$$

Nodes are selected with probability proportional to their capacity. The probability of the sampling without replacement scheme that the node n_i will be chosen is given by:

$$\mathbb{P}_{n_i} = \frac{\tilde{C}_{n_i}}{\sum \tilde{C}_{n_j}}$$

$\tilde{C}_{n_i}$ is the weight of the i^{th} node sampled.

5.3.1 Selection Algorithms

Our probabilistic selection guarantees that nodes with greater capabilities are more likely to be selected after randomly choosing enough nodes while allowing smaller ones to be chosen. SECUSCA_2, we maintain the same replication target function protocol as outlined in4.6.4, and thus the same parameters α_i, α_f, γ_i, and γ_f.

Algorithm 3 SECUSCA_2 protocol

Sample($\alpha_i N, N$): Each node has the capacity information of nodes in the network. The function returns *true* only if the number of nodes that can receive the new block is sufficient for the number of replicas needed. Otherwise, the parameters are adjusted.

1: Calculate the remaining capacity of each node in the network $\tilde{C}_{n_i}$.
2: Calculate the total remaining capacity of all nodes $\sum \tilde{C}_{n_j}$
3: Select n node with the probability $\mathbb{P}_{n_i} = \tilde{C}_{n_i} / \sum \tilde{C}_{n_j}$
4: Remove the selected node from the set of nodes.
5: Repeat Steps 2-4 until $\alpha_i N$ nodes have been selected from N nodes.
6: **return** Selected node for storage

5.4 Evaluation

We code SECUSCA_* functions for sharding, selecting nodes, and adding and deleting blocks in Python, available on GitHub, with the unfairness metric.

5.4.1 Evaluation Metric

In this section, we experimentally evaluate our protocol. We denote $sto(n_i)$, the set of blocks stored by the node n_i. The experiments compare the performance of the naïve algorithm, SECUSCA_1, and SECUSCA_2 algorithms according to the following unfairness metric:

$$Var\left(sat(n_I)\right) \;=\; \sum_i \left(sat(n_i) - \frac{1}{N} \sum_j sat(n_j) \right)^2$$

Where $sat(n_i) := \frac{\sum_{b \in sto(n_i)} b.size}{C_{n_i}}$ is the saturation of the node i.

This metric measures the unfairness of partition of the load, as it measures, for each node, the deviation of its saturation from the average saturation. We also observe that it gives more weight to the small nodes, as all nodes have the same

$1/N$ weight. And recall that the stake is here to attract small nodes to maximize fairness for them.

The primary questions we want to evaluate are about the unfairness of SECUSCA_1 compared to SECUSCA_2 and whether it truly scales out.

5.4.2 Experimental Setup

We run the experiment on generating 5000 blocks of size 0.5 with 50 nodes having capacities increasing linearly from 50 to 2500.

We also take the following parameters for the target replication: $\alpha_i = 0.3$, $\alpha_f = 0.1$, $\gamma_i = 0.5$, and $\gamma_f = 0.2$.

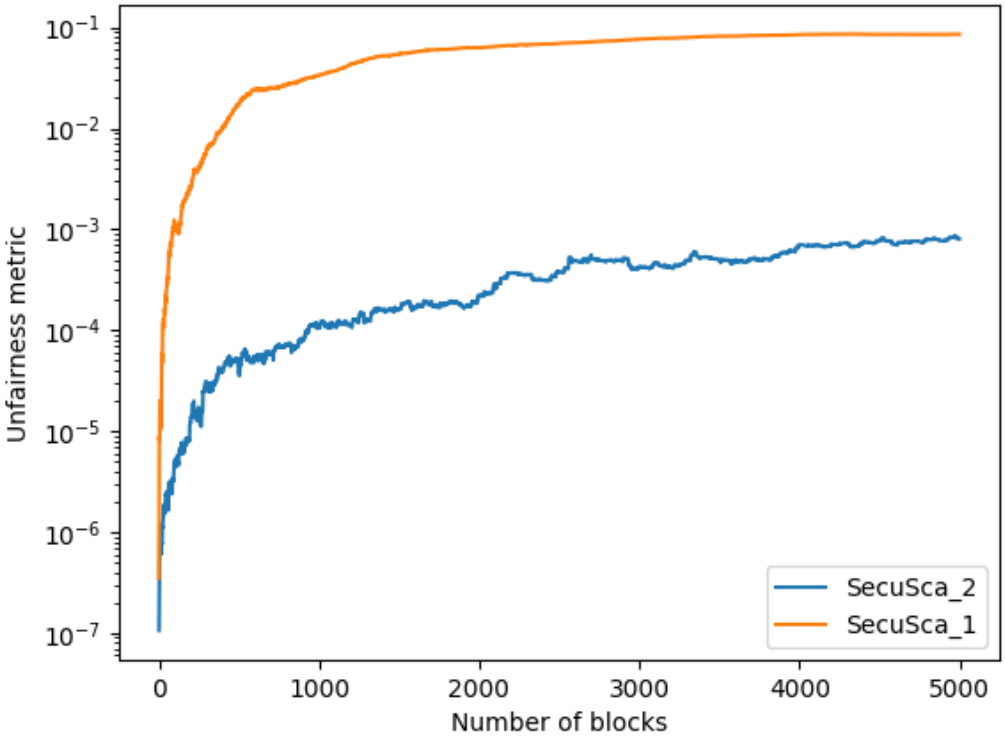


Figure 5.1: Comparison of the fairness of SECUSCA_1 and SECUSCA_2

We observe from Figure 5.1 that the saturation has a very small variance by a factor 10^{-2}. As we repeated a random choice 5000 times, this difference in variance size is of the order of magnitude $\sqrt{5000}$, which makes sense as there should be some central-limit theorem playing.

The important conclusion is that SECUSCA_2 guarantees a much fairer load distribution than SECUSCA_1 experimentally. We also observed in the experiment that the smaller nodes all got saturated. SECUSCA_2 is, therefore, a good solution to guarantee decentralization by attracting small participants in the network.

5.5 Benchmark with targeting the high capacity nodes

5.5.1 Alternative selection algorithm

An alternative approach would be sending the blocks to store from SECUSCA_1 to all the nodes with the highest capacity instead of sending it to a node selected randomly, like for Algorithm 3 of SECUSCA_2.

Algorithm 4 SECUSCA_1 highest capacity version protocol

Select nodes for storage: Each node has the capacity information of nodes in the network. The function returns *true* only if the number of nodes that can receive the new block is sufficient for the number of replicas needed. Otherwise, the parameters are adjusted.

```
Remaining_capacities= list of remaining_storage_capacity of all nodes
Total_capacity= ∑(Remaining_capacities)
for Capacity in Remaining_capacities do
  normalized_remaining_capacities = Capacity/Total_capacity
end for
Select target_number_of_nodes_for_storage nodes_selected_for_block_storage
from nodes with the highest normalized_remaining_capacities
return nodes_selected_for_block_storage
```

This approach seems simpler (no randomness needed) and allows us to exploit, at best, the capabilities of the most capable nodes.

5.5.2 Limitations of the alternative

The largest problem is the creation of centralization. It creates centralization if we rely only on the largest nodes to store the blockchain history. It makes the network too reliant on the largest nodes, defeating the purpose of using blockchain and decentralization.

Another limit that can be seen in a more quantitative way is in terms of the unfairness metric. As the protocol will only put the load on the largest nodes, the lightest nodes have not required any effort.

5.5.3 Comparaison of the unfairness metric

In the comparison of Subsection 5.4.2, we include the new protocol derived from SECUSCA_1 that uses Algorithm 4.

The result can be observed in Figure 5.2.

We observe that the version of SECUSCA with the highest capacities is better than SECUSCA_1 at the beginning. However, in a second time, SECUSCA with

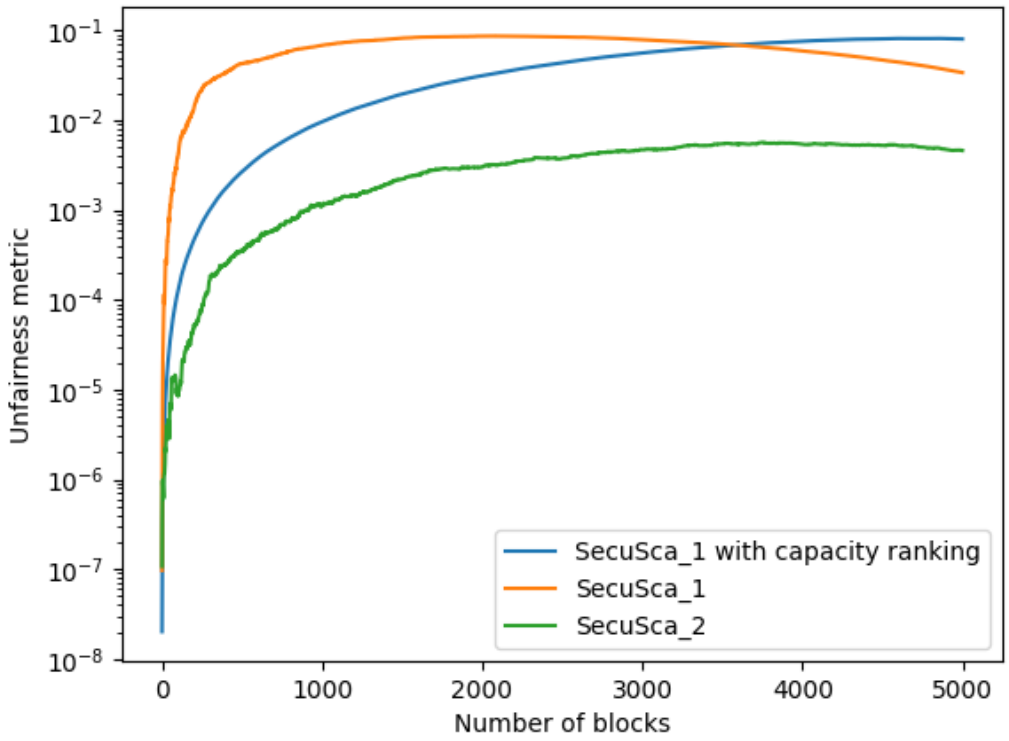


Figure 5.2: Fairness benchmark with SECUSCA_1, SECUSCA_2, and SECUSCA with highest capacities

the highest capacities becomes less fair, as there is more scarcity in terms of available memory.

What is worth mentioning is as well that SECUSCA_2 is by far beating this alternative in any setup. It seems, therefore, not worth considering this alternative, as SECUSCA_2 seems by far to be the best solution.

5.6 Summary

This chapter proposes a new algorithm for selecting nodes according to their storage capacity. We favor large-capacity nodes to be elected to store the blockchain. In Secusca 1, node selection is equal regardless of node capacity. This can lead to rapid saturation of some nodes with small capacity. The optimization we propose in this chapter allows the protocol to distribute the blocks according to the capacity of the nodes. Nodes with large remaining capacities are more likely to be chosen by the protocol.

We noticed in our simulations of the justice of the two protocols that the random choice can be favorable, provided, to have a high probability of the large nodes being selected while giving a chance to the small nodes to participate and ensure decentralization of storage over nodes.

Part IV.

Conclusion

Chapter 6

Conclusion and perspectives

6.1 Conclusion

The contributions presented in this manuscript relate first to the limits of the centralized systems that aim to secure users' identities. Second, we focus on designing a scalable permissionless blockchain in Proof_of_Work systems.

In our first contribution 3, we have improved Certificate Transparency to efficiently handle revocation, resulting in a secure, decentralized, and trustless Certificate Issuance and Revocation Transparency system. Its usefulness on the web is a result of its ability to provide secure communications through peer-to-peer encryption and no need for third-party approval. Public Key registration, revocation, and updating are based on consensus between participant entities. Any Blockchain node can act as an auditor and initiate revocation when suspicious activity is detected. Witnesses enable every node to effectively verify the public keys, eliminating the need for revocation lists. Our protocol is secure against attacks, unlike S/MIME, PGP, and CA. Overall, PKI based Blockchain provides an effective and reliable platform for data authentication and authorization, with the potential to revolutionize the way we interact with the digital world. It could provide a more secure online environment for businesses, consumers, and governments alike.

However, even with its potential, there are certain limitations to the blockchain technology. For example, the scalability of the blockchain is still an issue, as the network can become congested when too many transactions are being processed. Additionally, the costs associated with maintaining and operating the blockchain can be high, and it is often difficult to incentivize miners to stay on the network. Furthermore, blockchain technology is still relatively new, and there are still many areas which require further research before it can be adopted on a larger scale. Lastly, the lack of regulation and standardization may be a barrier to adoption, as different countries have different regulations and requirements when it comes to data privacy.

Blockchain based-systems have recently become appealing to several financial sectors and scientific communities. Currently, there exist various blockchains such as Ethereum [133], Hyperledger [10], Tezos[60] etc. Each blockchain has its operating mode and a different transaction validation consensus, making it attractive to various applications. A user broadcasts a new transaction to the network and adds it to the blockchain. A set of nodes then verify the new transaction to ensure that it is correctly signed and has not been previously spent (or recorded) in the ledger. A node with all these functions is called a full node - which maintains a complete copy of the blockchain and contributes to network security. Other nodes, supporting only a subset of functions, verify transactions using a simplified payment verification (SPV) method, known as lightweight nodes - they allow to send and receive

transactions without owning a full copy of the blockchain. But they download block headers, and transactions depend on full nodes.

Blockchain has won its spurs in data integrity, security, and immutability. However, security is achieved at the price of maintaining full nodes. Storage costs increase linearly with the number of transactions and may become one of the bottlenecks limiting blockchain's scalability. Traditional blockchain is based on full replication, where nodes rely on all past transactions locally. They check the state to validate a new transaction and then store each transaction to maintain the system. This represents proof of the correct state that consists of all block headers starting from the genesis's block. This processing is slow and requires a lot of storage capacity. Each node has to agree with that process. However, supporting many users and transactions results in a severe scalability problem.

On the one hand, the decentralization of blockchain on a Peer_to_Peer network and its replication on multiple nodes provide extra security by making it more difficult for an attacker to compromise the system. But on the other hand, these negatively affect the scalability of blockchain systems.

Reducing hardware requirements is essential to reduce the barrier of entry for running a full node, which is how blockchains have remained decentralized, a fundamental factor in building decentralized trust. The blockchain trilemma describes this dynamic, also called the scalability trilemma, which states that traditional blockchains can only maximize two properties: scalability, decentralization, and security. It is believed that the ability to scale up a blockchain lies mainly in improving the technology and not in deploying new hardware.

The contributions in III. present a new consensus mechanism that distributes the historical blocks to store among the active nodes. We are interested in the scalability of Blockchain, which is limited because of its décentralized and public characteristics; we formulate a method that could be used to solve the blockchain network scalability obstacle when the number of nodes and transactions increases without changing the main consensus of checking and validating new entries to the blockchain and without compromising security.

In 4, we proposed a SecuSca, a sharding approach that reduces block replication in the chain [79]. The idea is that the process acts as in the Bitcoin protocol, but transactions are not recorded in full replication. Nodes store only the block header to achieve consensus. Given a network with N nodes, the block replication is $\alpha * N$, with $(0 < \alpha < 1)$. Once a block is verified and confirmed by the network (i.e., a transaction is considered a success after six block confirmations), its replication decreases. We proposed an approach for a finite number of nodes C, and after

that, we underlined the major data availability problem. The malicious behavior of miners can cause the deletion of data from nodes.

6.2 Perspectives

PKI based blockchain: The protocol is an effective measure for ensuring the security of user identities. To further improve the limitations of our solution, several steps can be taken. Firstly, improving the scalability of the blockchain network drives us to propose SECUSCA. Secondly, it is important to ensure that the costs associated with the blockchain are kept to a minimum by employing technologies such as Proof-of-Stake. Thirdly, we must consider the privacy of the data stored on the blockchain, providing a scalable privacy infrastructure to support third-party applications and services in offering private access features to their users. These measures will ensure that the blockchain-based PKI is a secure and resilient solution for digital identity management.

SECUSCA_*

We chose to use basic transaction protocols (those of Bitcoin) without looking at smart contracts or short finality. Another important work that needs to be done is on the consensus approach; it may be necessary to replace Proof-of-Work with Proof-of-Stake, as well as implement a system of checkpoints. With a system of checkpoints, SECUSCA can execute the sharding protocol more quickly and securely. Checkpoints can be used to verify and secure certain points in the blockchain.

Part V.

ANNEXES

.1 SecuSca programming

```
1
2  class Block:
3      def __init__(self, size, identity):
4          self.size= size
5          self.identity= identity
6
7      def __repr__(self):
8          return f"Block(id={self.identity}, size={self.size})"
9
10 class Blockchain:
11     def __init__(self) -> None:
12         self.chain=[]
13         self.peers=Network()
14         self.nodes_storing_blocks = []
15         self.unfairness_values=[]
16
17
18     def add_new_block(self, block):
19         self.chain.append(block)
20
21     def mine_block(self):
22         new_block= Block(size=0.5, identity=len(self.chain))
23         self.delete_blocks()
24         self.add_new_block(new_block)
25         self.store_new_block(new_block)
26
27
28     def delete_blocks(self):
29         for block in self.chain:
30             self.delete_block(block)
31
32
33      def delete_block(self, block):
34         target_replication = self.peers.get_target_replication
35         (block=block, blockchain_size=len(self.chain)+1)
36         actual_replication = len(self.nodes_storing_blocks
37         [block.identity])
38         if actual_replication > target_replication:
39             self.delete_block_once(block)
40         elif actual_replication < target_replication:
41             self.store_block_once(block)
42
43
44     def store_block_once(self, block):
45         nodes_with_enough_space = [node for node in self.peers.
```

```
            nodes
46          if node.remaining_storage_capacity()>=block.size
47          and node not in self.nodes_storing_blocks[block.identity]]
48          assert len(nodes_with_enough_space) >= 1,
49          "Not enough storage to continue the protocol"
50          nodes_for_addition = self.select_nodes_for_storage
51          (nodes = nodes_with_enough_space, number_of_nodes=1)
52          if nodes_for_addition:
53              node_for_addition = nodes_for_addition[0]
54              node_for_addition.store_block(block)
55             self.nodes_storing_blocks[block.identity]
56             .append(node_for_addition)
57
58
59      def delete_block_once(self, block):
60          node_for_deletion = random.choice
61          (self.nodes_storing_blocks[block.identity])
62          node_for_deletion.delete_block(block)
63          self.nodes_storing_blocks[block.identity]
64          .remove(node_for_deletion)
65
66      def store_new_block(self, new_block):
67          replication_block= self.peers.get_target_replication
68          (block=new_block, blockchain_size=len(self.chain))
69          nodes_with_enough_space =
70          [node for node in self.peers.nodes if
71
72          node.remaining_storage_capacity()>=new_block.size]
73          assert len(nodes_with_enough_space) >= replication_block,
74
75          "Not enough storage to
76          continue the protocol"
77          random_peers = self.select_nodes_for_storage
78          (nodes = nodes_with_enough_space, number_of_nodes=
79          replication_block)
80
81          for node in random_peers:
82              node.store_block(new_block)
83
84          self.nodes_storing_blocks.append(random_peers)
85
86      def select_nodes_for_storage(self, nodes, number_of_nodes):
87          #Dummy function for the mother class
88          return []
89
90
91      def generate_blockchain(self, number_of_blocks):
```

```
92          for i in range(number_of_blocks):
93              self.mine_block()
94              self.unfairness_values.append
95              (self.peers.compute_unfairness())
96              if (i+1)*10 % number_of_blocks == 0:
97                  print("Block number ", i+1, "mined")
98
99
100 class Secusca1(Blockchain):
101     def select_nodes_for_storage(self, nodes, number_of_nodes):
102         return random.sample(nodes, number_of_nodes)
103
104
105 class Secusca2(Blockchain):
106     def select_nodes_for_storage(self, nodes, number_of_nodes):
107          remaining_capacities =
108          [node.remaining_storage_capacity() for node in nodes]
109          total_capacity = sum(remaining_capacities)
110          normalized_remaining_capacities =
111          [capacity/total_capacity for
112          capacity in remaining_capacities]
113          return [*np.random.choice(nodes,
114          size=number_of_nodes,*
115          replace=False, p=normalized_remaining_capacities)]
116
117
118
119 class Node:
120     def __init__(self, storage_capacity) -> None:
121         self.storage_capacity = storage_capacity
122         self.blocks_stored = []
123
124     def __repr__(self):
125         return f"Node(capacity={self.storage_capacity},
126         blocks_stored={self.blocks_stored}),
127         remaining capacity={self.remaining_storage_capacity()}"
128
129     def store_block(self, block):
130         self.blocks_stored.append(block)
131
132     def stored_blocks_size(self):
133         return sum([block.size for block in self.blocks_stored])
134
135
136     def remaining_storage_capacity(self):
137         return self.storage_capacity-self.stored_blocks_size()
138
```

```
139     def delete_block(self, block):
140         self.blocks_stored.remove(block)
141
142
143
144 class Network:
145     def __init__(self):
146         self.nodes =
147
148         Network._create_nodes
149                 (number_nodes=NUMBER_OF_NODES,
150                  min_capacity=MIN_CAPACITY,
151                  capacity_increment=CAPACITY_INCREMENT)
152
153     @staticmethod
154     def _create_nodes(number_nodes, min_capacity,
           capacity_increment):
155         return [Node(storage_capacity=
156
157         min_capacity+i*capacity_increment)
158                     for i in range(number_nodes)]
159
160     def get_target_replication(self, block, blockchain_size):
161         block_depth=blockchain_size -1 - block.identity
162         gamma_b= int(GAMMA*blockchain_size)
163         gamma_0= int(GAMMA_0*blockchain_size)
164
165      if block_depth <= gamma_0:
166         block_storage_replication= ALPHA*NUMBER_OF_NODES
167
168      elif block_depth <= gamma_b and block_depth >= gamma_0:
169          block_storage_replication=
170          (((ALPHA*NUMBER_OF_NODES - ALPHA_0*NUMBER_OF_NODES)
171          /(gamma_b - gamma_0))*
172           (gamma_b - block_depth)) +ALPHA_0*NUMBER_OF_NODES
173
174         else:
175             assert block_depth >= gamma_b
176             block_storage_replication= ALPHA_0*NUMBER_OF_NODES
177
178         return int(block_storage_replication)
179
180 def compute_unfairness(self):
181     saturation_expectation =
182     sum([node.stored_blocks_size()/node.storage_capacity for node
           in
183     self.nodes])/len(self.nodes)squared_saturation_expectation=
```

```
184     sum([(node.stored_blocks_size()/node.storage_capacity)**2
185     for node in self.nodes])/len(self.nodes)
186 
187     return squared_saturation_expectation-saturation_expectation**2
```

Bibliography

[1] Bitcoin core. https://bitcoin.org/en/bitcoin-core/. Accessed: 10/2/2022.

[2] Bitcoin wiki. confirmation. https://en.bitcoin.it/wiki/Confirmation. Accessed: 10/2/2022.

[3] bitcoin.org. https://bitcoin.org/en/full-node. Accessed: 10/2/2022.

[4] blockchain.com. https://www.blockchain.com/explorer/blocks/btc?page=1. Accessed: 10/2/2022.

[5] Décret n° 2009-834 du 7 juillet 2009 portant création d'un service à compétence nationale dénommé « agence nationale de la sécurité des systèmes d'information ». https://www.legifrance.gouv.fr/loda/id/JORFTEXT000020828212/.

[6] ethereum.org. https://ethereum.org/en/developers/docs/gas/. Accessed: 10/2/2022.

[7] Github-khacefkahina. https://github.com/khacefkahina/Blockchain-Sharding-Protocol. Accessed: 10/2/2022.

[8] National institute of standards and technology. https://www.nist.gov/. Accessed: 10/2/2022.

[9] Nxt whitepaper revision 4 v1.2.2, 2014. https://www.dropbox.com/s/cbuwrorf672c0yy/NxtWhitepaper. Accessed: 10/2/2022.

[10] Hyperledger Architecture Volume 1. Introduction to hyperledger business blockchain design philosophy and consensus. https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf, 2003. Accessed: 10/2/2022.

[11] Nadhem J. AlFardan, Daniel J. Bernstein, Kenneth G. Paterson, and Bertram Poettering. On the security of rc4 in tls and wpa. 2013.

[12] Nadhem J. AlFardan and Kenneth G. Paterson. Lucky thirteen: Breaking the tls and dtls record protocols. *2013 IEEE Symposium on Security and Privacy*, pages 526–540, 2013.

[13] Muneeb Ali, Jude Nelson, Ryan Shea, and Michael J. Freedman. Blockstack: A global naming and storage system secured by blockchains. In *2016 USENIX Annual Technical Conference (USENIX ATC 16)*, pages 181–194, Denver, CO, 2016.

[14] Saqib Ali, Guojun Wang, Bebo White, and Roger Leslie Cottrell. A blockchain-based decentralized data storage and access framework for pinger. *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 1303–1308, 2018.

[15] Mohammad Javad Amiri, Divyakant Agrawal, and Amr El Abbadi. On sharding permissioned blockchains. In *IEEE International Conference on Blockchain, Blockchain 2019, Atlanta, GA, USA, July 14-17, 2019*, pages 282–285. IEEE, 2019.

[16] Mohammad Javad Amiri, Divyakant Agrawal, and Amr El Abbadi. Sharper: Sharding permissioned blockchains over network clusters. *CoRR*, abs/1910.00765, 2019.

[17] Katrina A. Armstrong. If you can't beat it, join it: Uncertainty and trust in medicine. *Annals of Internal Medicine*, 168:818–819, 2018.

[18] David A. Basin, Cas J. F. Cremers, Tiffany Hyun-Jin Kim, Adrian Perrig, Ralf Sasse, and Pawel Szalachowski. Arpki: Attack resilient public-key infrastructure. *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014.

[19] Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *CCS '93*, 1993.

[20] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. *2014 IEEE Symposium on Security and Privacy*, pages 459–474, 2014.

[21] Iddo Bentov, Ariel Gabizon, and Alex Mizrahi. Cryptocurrencies without proof of work. In *Financial Cryptography Workshops*, 2016.

[22] Ferenc Béres, István András Seres, and András A. Benczúr. A cryptoeconomic traffic analysis of bitcoins lightning network. *ArXiv*, abs/1911.09432, 2020.

[23] Joseph Bonneau. Ethiks: Using ethereum to audit a coniks key transparency log. In *Financial Cryptography Workshops*, 2016.

[24] Joseph Bonneau, Izaak Meckler, Vanishree Rao, Evan, and Shapiro. Mina : Decentralized cryptocurrency at scale. 2021.

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

[26] Vitalik Buterin. A next generation smart contract & decentralized application platform. 2015.

[27] John L. Callas, Lutz Donnerhacke, Hal Finney, and Rodney Thayer. Openpgp message format. *RFC*, 4880:1–90, 1998.

[28] Miguel Castro. Practical byzantine fault tolerance. In *OSDI '99*, 1999.

[29] Jing Chen and Silvio Micali. Algorand: A secure and efficient distributed ledger. *Theor. Comput. Sci.*, 777:155–183, 2019.

[30] Alexander Chepurnoy, Mario Larangeira, and Alexander Ojiganov. Rollerchain, a blockchain with safely pruneable full blocks. *arXiv: Cryptography and Security*, 2016.

[31] Usman W. Chohan. A history of dogecoin. *Innovation Finance & Accounting eJournal*, 2021.

[32] Beth Cohen. Incentives build robustness in bit-torrent. 2003.

[33] Mauro Conti, Nicola Dragoni, and Viktor Lesyk. A survey of man in the middle attacks. *IEEE Communications Surveys & Tutorials*, 18:2027–2051, 2016.

[34] Matt Cooper, Yuriy Dzambasow, Peter Hesse, Susan Joseph, and Richard Nicholas. Internet x.509 public key infrastructure: Certification path building. *RFC*, 4158:1–81, 2005.

[35] Francisco Corella. Backing rich credentials with a blockchain pki. 2016.

[36] Carl M. Ellison Cybercash. Establishing identity without certification authorities. 1996.

[37] Hung Dang, Tien Tuan Anh Dinh, Dumitrel Loghin, Ee-Chien Chang, Qian Lin, and Beng Chin Ooi. Towards scaling blockchain systems via sharding. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, pages 123–140. ACM, 2019.

[38] Bernardo Machado David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *EUROCRYPT*, 2018.

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

[40] Richard Dennis, Gareth Owenson, and Benjamin Aziz. A temporal blockchain: A formal analysis. *2016 International Conference on Collaboration Technologies and Systems (CTS)*, pages 430–437, 2016.

[41] T. B. Developers. Developer guide - bitcoin. available at:. https://developer.bitcoin.org/devguide/. Accessed: 10/2/2022.

[42] Tim Dierks and Eric Rescorla. The transport layer security (tls) protocol version 1.2. *RFC*, 5246:1–104, 2008.

[43] Zheng Dong, Kevin Kane, and L. Jean Camp. Detection of rogue certificates from trusted certificate authorities using deep neural networks. *ACM Transactions on Privacy and Security (TOPS)*, 19:1 – 31, 2016.

[44] John R. Douceur. The sybil attack. In *IPTPS*, 2002.

[45] Paul Dunphy and Fabien A. P. Petitcolas. A first look at identity management schemes on the blockchain. *IEEE Security & Privacy*, 16:20–29, 2018.

[46] Ashutosh Dhar Dwivedi, Gautam Srivastava, Shalini Dhar, and Rajani Singh. A decentralized privacy-preserving healthcare blockchain for iot. *Sensors (Basel, Switzerland)*, 19, 2019.

[47] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *CRYPTO*, 1992.

[48] Muhammad El-Hindi, Carsten Binnig, Arvind Arasu, Donald Kossmann, and Ravishankar Ramamurthy. Blockchaindb - a shared database on blockchains. *Proc. VLDB Endow.*, 12:1597–1609, 2019.

[49] Carl M. Ellison and Bruce Schneier. Ten risks of pki. 2000.

[50] Farid F. Elwailly, Craig Gentry, and Zulfikar Ramzan. Quasimodo: Efficient certificate validation and revocation. In *Public Key Cryptography*, 2004.

[51] Patrick Eugster, Rachid Guerraoui, Anne-Marie Kermarrec, and Laurent Massoulié. Epidemic information dissemination in distributed systems. *IEEE Computer*, 37:60–67, 05 2004.

[52] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert van Renesse. Bitcoin-ng: A scalable blockchain protocol. In *NSDI*, 2016.

[53] Conner Fromknecht and Dragos Velicanu. Certcoin : A namecoin based decentralized authentication system 6 . 857 class project. 2014.

[54] Conner Fromknecht, Dragos Velicanu, and Sophia Yakoubov. A decentralized public key infrastructure with identity retention. *IACR Cryptol. ePrint Arch.*, 2014:803, 2014.

[55] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *EUROCRYPT*, 2015.

[56] Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. Threshold-optimal dsa/ecdsa signatures and an application to bitcoin wallet security. In *ACNS*, 2016.

[57] Arthur Gervais, Ghassan O. Karame, K. Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.

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

[59] Jennifer Golbeck. Computing and applying trust in web-based social networks. 2005.

[60] L.M Goodman. Tezos a self amending crypto ledger white paper. https://tezos.com/whitepaper.pdf, 2017. Accessed: 10/2/2022.

[61] Vipul Goyal. Certificate revocation using fine grained certificate space partitioning. In *Financial Cryptography*, 2007.

[62] Tyrone Grandison and Morris Sloman. A survey of trust in internet applications. *IEEE Communications Surveys & Tutorials*, 3:2–16, 2000.

[63] Zhaohui Guo, Zhen Gao, Haojuan Mei, Ming Zhao, and Jinsheng Yang. Design and optimization for storage mechanism of the public blockchain based on redundant residual number system. *IEEE Access*, 7:98546–98554, 2019.

[64] Peter Gutmann. Lessons learned in implementing and deploying crypto software. In *Proceedings of the 11th USENIX Security Symposium*, page 315–325, USA, 2002. USENIX Association.

[65] Jason J. Haas, Yih-Chun Hu, and Kenneth P. Laberteaux. Efficient certificate revocation list organization and distribution. *IEEE Journal on Selected Areas in Communications*, 29:595–604, 2011.

[66] Christopher Hannon and Dong Jin. Bitcoin payment-channels for resource limited iot devices. *Proceedings of the International Conference on Omni-Layer Intelligent Systems*, 2019.

[67] Susan J. Harrington and Cynthia P. Ruppel. Telecommuting: a test of trust, competing values, and relative advantage. *IEEE Transactions on Professional Communication*, 42:223–239, 1999.

[68] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. In *USENIX Security Symposium*, 2015.

[69] Joanne Holliday, R. Steinke, Dharma Agrawal, and A. Abbadi. Epidemic algorithms for replicated databases. *Knowledge and Data Engineering, IEEE Transactions on*, 15:1218– 1238, 10 2003.

[70] Russ Housley and Paul E. Hoffman. Internet x.509 public key infrastructure operational protocols: Ftp and http. *RFC*, 2585:1–8, 1999.

[71] Lin-Shung Huang, Alex Rice, Erling Ellingsen, and Collin Jackson. Analyzing forged ssl certificates in the wild. *2014 IEEE Symposium on Security and Privacy*, pages 83–97, 2014.

[72] S. M. Riazul Islam, Daehan Kwak, Md Humaun Kabir, Mahmud Shahriar Hossain, and Kyung Sup Kwak. The internet of things for health care: A comprehensive survey. *IEEE Access*, 3:678–708, 2015.

[73] Hudson Jameson. Renaming suicide opcode/ interface. available at:. https://github.com/ethereum/EIPs/blob/master/EIPS/eip-6.md/. Accessed: 10/2/2022.

[74] Marco Alberto Javarone and Craig Steven Wright. From bitcoin to bitcoin cash: a network analysis. *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, 2018.

[75] Donald Byron Johnson, Alfred Menezes, and Scott A. Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International Journal of Information Security*, 1:36–63, 2001.

[76] Harry A. Kalodner, Miles Carlsten, Paul Ellenbogen, Joseph Bonneau, and Arvind Narayanan. An empirical study of namecoin and lessons for decentralized namespace design. In *WEIS*, 2015.

[77] Kostis Karantias, Aggelos Kiayias, Nikos Leonardos, and Dionysis Zindros. Compact storage of superblocks for nipopow applications. In *IACR Cryptol. ePrint Arch.*, 2019.

[78] Steven P. Ketchpel and Hector Garcia-Molina. Making trust explicit in distributed commerce transactions. *Proceedings of 16th International Conference on Distributed Computing Systems*, pages 270–281, 1996.

[79] Kahina Khacef, Salima Benbernou, Mourad Ouziri, and Muhammad Younas. Trade-off between security and scalability in blockchain design: A dynamic sharding approach. In *The International Conference on Deep Learning, Big Data and Blockchain*, volume 309, pages 77–90. Springer, 2021.

[80] Kahina Khacef and Guy Pujolle. Secure peer-to-peer communication based on blockchain. In *Web, Artificial Intelligence and Network Applications - Proceedings of the Workshops of the 33rd International Conference on Advanced Information Networking and Applications, AINA 2019, Matsue, Japan, March 27-29, 2019*, volume 927 of *Advances in Intelligent Systems and Computing*, pages 662–672. Springer, 2019.

[81] Aggelos Kiayias, Nikolaos Lamprou, and Aikaterini-Panagiota Stouka. Proofs of proofs of work with sublinear complexity. In *Financial Cryptography Workshops*, 2016.

[82] Aggelos Kiayias, Andrew K. Miller, and Dionysis Zindros. Non-interactive proofs of proof-of-work. *IACR Cryptol. ePrint Arch.*, 2017:963, 2017.

[83] Aggelos Kiayias, Alexander Russell, Bernardo Machado David, and R. Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *CRYPTO*, 2016.

[84] Tiffany Hyun-Jin Kim, Lin-Shung Huang, Adrian Perrig, Collin Jackson, and Virgil D. Gligor. Accountable key infrastructure (aki): a proposal for a public-key validation infrastructure. *Proceedings of the 22nd international conference on World Wide Web*, 2013.

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

[86] 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. *CoRR*, abs/1602.06997, 2016.

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

[88] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. Omniledger: A secure, scale-out, decentralized ledger via sharding. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 583–598. IEEE Computer Society, 2018.

[89] Oleksii Konashevych. Emercoin blockchain anchoring as a way of singing contracts. In Víctor Rodríguez-Doncel, Pompeu Casanovas, Jorge González-Conejero, and Elena Montiel-Ponsoda, editors, *Proceedings of the 2nd Workshop on Technologies for Regulatory Compliance co-located with the 31st International Conference on Legal Knowledge and Information Systems*, 2018.

[90] Murat Yasin Kubilay, Mehmet Sabir Kiraz, and Haci Ali Mantar. Certledger: A new pki model with certificate transparency based on blockchain. *IACR Cryptol. ePrint Arch.*, 2018:1071, 2018.

[91] Ben Laurie, Adam Langley, and Emilia Käsper. Certificate transparency. *Queue*, 12:10 – 19, 2013.

[92] Ben Laurie, Eran Messeri, and Rob Stradling. Certificate transparency version 2.0. *RFC*, 9162:1–53, 2021.

[93] Arjen K. Lenstra, James P. Hughes, Maxime Augier, Joppe W. Bos, Thorsten Kleinjung, and Christophe Wachter. Public keys. In *CRYPTO*, 2012.

[94] Derek Leung, Adam Suhl, Yossi Gilad, and Nickolai Zeldovich. Vault: Fast bootstrapping for the algorand cryptocurrency. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019.

[95] Yuan Lu, Qiang Tang, and Guiling Wang. Generic superlight client for permissionless blockchains. *CoRR*, abs/2003.06552, 2020.

[96] Christer Lundkvist. https://media.consensys.net/state-change-10-christian-lundkvist-uport-65d08c7def1b, 2018. Accessed: 10/2/2022.

[97] Teresa F. Lunt, Dorothy E. Denning, Roger R. Schell, Mark R. Heckman, and William R. Shockley. The seaview security model. *Proceedings. 1988 IEEE Symposium on Security and Privacy*, pages 218–233, 1988.

[98] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and P. Saxena. A secure sharding protocol for open blockchains. *Proceedings*

of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016.

[99] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. A secure sharding protocol for open blockchains. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 17–30. ACM, 2016.

[100] Aleksander Malinowski and Bogdan M. Wilamowski. The transmission control protocol. In *The Industrial Information Technology Handbook*, 2005.

[101] Stephanos Matsumoto and Raphael M. Reischuk. Ikp: Turning a pki around with blockchains. *IACR Cryptol. ePrint Arch.*, 2016:1018, 2016.

[102] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the XOR metric.

[103] Marcela S. Melara, Aaron Blankstein, Joseph Bonneau, Edward W. Felten, and Michael J. Freedman. Coniks: Bringing key transparency to end users. In *USENIX Security Symposium*, 2015.

[104] Ralph C. Merkle. A digital signature based on a conventional encryption function. In *CRYPTO*, 1987.

[105] Ulrike Meyer and Susanne Wetzel. A man-in-the-middle attack on umts. In *in Proceedings of the 2004 ACM Workshop on Wireless Security*, pages 90–97. ACM Press, 2004.

[106] Silvio Micali. Novomodo : Scalable certificate validation and simplified pki management. 2002.

[107] Nitin Naik and Paul Jenkins. uport open-source identity management system: An assessment of self-sovereign identity and user-centric data platform built on blockchain. *2020 IEEE International Symposium on Systems Engineering (ISSE)*, pages 1–7, 2020.

[108] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, 2008. Accessed: 10/2/2022.

[109] Rodrigue Carlos Nana Mbinkeu and Bernabe Batchakui. Reducing disk storage with sqlite into bitcoin architecture. *Int. J. Recent Contributions Eng. Sci. IT*, 3(2):10–14, 2015.

[110] John Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951.

[111] Joann J. Ordille. When agents roam, who can you trust? *Proceedings of COM'96. First Annual Conference on Emerging Technologies and Applications in Communications*, pages 188–191, 1996.

[112] Michael Piatek, Tomas Isdal, Thomas E. Anderson, Arvind Krishnamurthy, and Arun Venkataramani. Do incentives build robustness in bittorrent. 2007.

[113] Damian Poddebniak, Christian Dresen, Jens Müller, Fabian Ising, Sebastian Schinzel, Simon Friedberger, Juraj Somorovsky, and Jörg Schwenk. Efail: Breaking s/mime and openpgp email encryption using exfiltration channels. In *USENIX Security Symposium*, 2018.

[114] Andrew Poelstra. Distributed consensus from proof of stake is impossible. 2015.

[115] Joseph Poon. Plasma : Scalable autonomous smart contracts. 2017.

[116] Jon Postel. User datagram protocol. *RFC*, 768:1–3, 1980.

[117] Johan A. Pouwelse, Pawel Garbacki, Dick H. J. Epema, and Henk J. Sips. The bittorrent p2p file-sharing system: Measurements and analysis. In *IPTPS*, 2005.

[118] Bo Qin, Jikun Huang, Qin Wang, Xizhao Luo, Bin Liang, and Wenchang Shi. Cecoin: A decentralized pki mitigating mitm attacks. *Future Gener. Comput. Syst.*, 107:805–815, 2020.

[119] Eric Rescorla. The transport layer security (tls) protocol version 1.3. *RFC*, 8446:1–160, 2018.

[120] Rodrigo Román, Carmen Fernández-Gago, Javier López, Hsiao-Hwa Chen, Stefanos Gritzalis, Tom Karygiannis, and Charalabos Skianis. Trust and reputation systems for wireless sensor networks. 2009.

[121] Quirin Scheitle, Taejoong Chung, Jens Hiller, Oliver Gasser, Johannes Naab, Roland van Rijswijk-Deij, Oliver Hohlfeld, Ralph Holz, David R. Choffnes, Alan Mislove, and Georg Carle. A first look at certification authority authorization (caa). *Comput. Commun. Rev.*, 48:10–23, 2018.

[122] Stefan Schumacher, René Pfeiffer, and Hanno Böck. Efail and other failures with encryption and e-mail outdated crypto standards and html mails as a security risk. 2019.

[123] Charles Shen and Feniosky Peña-Mora. Blockchain for cities—a systematic literature review. *IEEE Access*, 6:76787–76819, 2018.

[124] Ankush Singla and Elisa Bertino. Blockchain-based pki solutions for iot. *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, pages 9–15, 2018.

[125] Jiawen Su and Daniel W. Manchala. Trust vs. threats: recovery and survival in electronic commerce. *Proceedings. 19th IEEE International Conference on Distributed Computing Systems (Cat. No.99CB37003)*, pages 126–133, 1999.

[126] Pawel Szalachowski, Stephanos Matsumoto, and Adrian Perrig. Policert: Secure and flexible tls certificate management. *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014.

[127] Viroj Tangcharoensathien, Neville Calleja, Tim Nguyen, Tina D. Purnat, Marcelo D'agostino, Sebastian Garcia-Saiso, Mark Landry, Arash Rashidian, Clayton Hamilton, Abdelhalim AbdAllah, Ioana Ghiga, Alexandra Hill, Daniel Hougendobler, Judith van Andel, Mark Nunn, Ian Brooks, Pier Luigi Sacco, Manlio De Domenico, Philip Mai, Anatoliy A. Gruzd, Alexandre Alaphilippe, and Sylvie Briand. Framework for managing the covid-19 infodemic: Methods and results of an online, crowdsourced who technical consultation. *Journal of Medical Internet Research*, 22, 2020.

[128] Harmony team. Harmony. https://harmony.one/whitepaper.pdf, 2017. Accessed: 10/2/2022.

[129] ZILLIQA team and Others. The zilliqa. *Technical White paper*, 16, 2017.

[130] Alexander Ulrich, Ralph Holz, Peter Hauck, and Georg Carle. Investigating the openpgp web of trust. In *ESORICS*, 2011.

[131] Nicolas van Saberhagen. Cryptonote v 2.0. 2013.

[132] David K. Vawdrey, Tore Sundelin, K.E. Seamons, and Charles D. Knutson. Trust negotiation for authentication and authorization in healthcare information systems. *Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (IEEE Cat. No.03CH37439)*, 2:1406–1409 Vol.2, 2003.

[133] V.Buterin. Ethereum, white paper. https://ethereum.org/en/whitepaper/. Accessed: 10/2/2022.

[134] Marko Vukolic. Eventually returning to strong consistency. *IEEE Data Eng. Bull.*, 39:39–44, 2016.

[135] Ze Wang, Jingqiang Lin, Quanwei Cai, Qiongxiao Wang, Daren Zha, and Jiwu Jing. Blockchain-based certificate transparency and revocation transparency. *IEEE Transactions on Dependable and Secure Computing*, 19:681–697, 2022.

[136] Zooko Wilcox-O'Hearn. Names: Decentralized, secure, human-meaningful: Choose two. https://web.archive.org/web/20011020191610/http://zooko.com/distnames, 2003. Accessed: 10/2/2022.

[137] Uwe G. Wilhelm, Sebastian Staamann, and Levente Buttyán. On the problem of trust in mobile agent systems. In *NDSS*, 1998.

[138] Sam A. Williams, Viktor Diordiiev, and Lev Berman. Arweave: A protocol for economically sustainable information permanence. 2019.

[139] Sam A. Williams and Will Jones. Archain: An open, irrevocable, unforgeable and uncensorable archive for the internet. 2017.

[140] Daniel Davis Wood. Ethereum: A secure decentralised generalised transaction ledger. 2014.

[141] Yang Xiao, Ning Zhang, Wenjing Lou, and Y. Thomas Hou. A survey of distributed consensus protocols for blockchain networks. *IEEE Communications Surveys & Tutorials*, 22:1432–1465, 2020.

[142] A. T. Yakovenko. Solana : A new architecture for a high performance blockchain v 0 . 8. 2018.

[143] Kan Yang, Lan Wang, Jobin Sunny, Yingdi Yu, Alexander Afanasyev, David D. Clark, kc Claffy, Van Jacobson, and Lixia Zhang. Blockchain-based decentralized public key management for named data networking. 2018.

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

[145] Jiangshan Yu, Vincent Cheval, and Mark D. Ryan. Dtki: A new formalized pki with verifiable trusted parties. *Comput. J.*, 59:1695–1713, 2016.

[146] Jiangshan Yu and Mark D. Ryan. Evaluating web pkis. *IACR Cryptol. ePrint Arch.*, 2017:526, 2017.

[147] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. Rapidchain: A fast blockchain protocol via full sharding. *IACR Cryptol. ePrint Arch.*, 2018:460, 2018.

[148] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. Rapidchain: Scaling blockchain via full sharding. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 931–948. ACM, 2018.