A Novel Privacy-Preserving Data Sharing System based on Attributed-based Encryption and Zero Knowledge Proof

Shivani, MSc

SUBMITTED IN DEGREE FULFILMENT OF THE REQUIREMENTS FOR MASTER OF SCIENCE (RESEARCH) IN COMPUTING SCIENCE

SCHOOL OF COMPUTING SCIENCE

COLLEGE OF SCIENCE AND ENGINEERING

NOVEMBER 2024

To my family

Abstract

The exponential growth of digital data across various sectors, such as healthcare, finance, and e-commerce, has underscored critical concerns regarding data privacy, security, and ownership. Centralised data storage systems are inherently vulnerable to cyber-attacks, raising significant privacy risks and compliance challenges, despite regulatory frameworks like the General Data Protection Regulation (GDPR). This research introduces a decentralised, privacy-preserving data-sharing framework leveraging blockchain technology, Ciphertext-Policy Attribute-Based Encryption (CP-ABE), and Zero-Knowledge Proofs (ZKP).

By employing CP-ABE, the proposed system enables fine-grained access control, ensuring that only authorised entities can access sensitive data based on specified attributes. The integration of Zero-Knowledge Proofs preserves user privacy by allowing verification of access rights without revealing the underlying attributes. The system architecture is underpinned by decentralised storage, with smart contracts managing secure access verification.

Performance evaluations demonstrate that the system effectively handles dynamic policies and attribute sets, demonstrating its adaptability to real-world applications. This framework represents a significant advancement in privacy-preserving data-sharing technologies, offering a scalable and secure solution for safeguarding sensitive users' attributes in decentralised environments.

Contents

List of Tables

List of Figures

Acknowledgements

I would like to extend my deepest gratitude to my supervisor, Dr. Nguyen Truong, for his invaluable guidance, support, and encouragement throughout my research journey. His mentorship has not only enriched my research but has also greatly influenced my growth as a researcher, for which I am sincerely grateful.

My heartfelt appreciation also goes to my family and friends for their constant encouragement and patience throughout this journey. Their belief in my potential provided the foundation upon which this work was built.

I would also like to acknowledge the School of Computing Science at the University of Glasgow for creating an inspiring and supportive academic environment. The resources and opportunities provided by the university have been instrumental in facilitating my research.

Finally, I extend my appreciation to all those who, directly or indirectly, contributed to this research. Your support and encouragement have been invaluable in bringing this work to fruition.

Declaration

I declare that, except where explicit reference is made to the contribution of others, that this dissertation is the result of my own work and has not been submitted for any other degree at the University of Glasgow or any other institution

Shivani

Abbreviations

Abbreviation	**Description**
ABE	Attribute-Based Encryption
ABAC	Attribute-Based Access Control
AES	Advanced Encryption Standard
BaaS	Backend-as-a-Service
BSW	Bethencourt-Sahai-Waters
CID	Content Identifier
CDN	Content Delivery Networks
CP-ABE	Ciphertext-Policy Attribute-Based Encryption
CRS	Common Reference String
DApps	Decentralized Applications
DA0	Decentralized autonomous organization
DO	Data Owner
DP	Data Processor
DP	Differential Privacy
DLT	Distributed Ledger Technology
DPoS	Delegated Proof of Stake
ECDSA	Elliptic Curve Digital Signature Algorithm
ERC	Ethereum Request for Comments
ETH	Ether currency
EVM	Ethereum Virtual Machine

FHE	Fully Homomorphic Encryption
GDPR	General Data Protection Regulation
HIPAA	Health Insurance Portability and Accountability Act
HE	Homomorphic Encryption
HTTP	Hypertext Transfer Protocol
IPFS	InterPlanetary File System
JWT	JSON Web Token
KGS	Key Generation Server
KP-ABE	Key-Policy Attribute-Based Encryption
LDP	Local Differential Privacy
MSK	Master Secret Key
MUI	Material-UI
NFT	Non-Fungible Token
NIZK	Non-Interactive Zero-Knowledge
P2P	Peer-to-Peer
PHE	Partially Homomorphic Encryption
PK	Public Key
PoA	Proof of Authority
PoH	Proof of History
PoS	Proof of Stake
PoW	Proof of Work
RABE	Rust Attribute-Based Encryption
RBAC	Role-Based Access Control
RSA	Rivest-Shamir-Adleman
SC	Smart Contract
SDK	Software Development Kit
SMPC	Secure Multi-Party Computation
TLS	Transport Layer Security
WASM	WebAssembly

zk-SNARKs	Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge
zk-STARKs	Zero-Knowledge Scalable Transparent Arguments of Knowledge
ZK	Zero-Knowledge
ZKP	Zero-Knowledge Proof
zkServer	Zero Knowledge Server
ZK CP-ABE	Zero-Knowledge Ciphertext-Policy Attribute-Based Encryption
zkVM	Zero-Knowledge Virtual Machine

Chapter 1

Introduction

The exponential growth of digital data in recent years has brought significant advancements in various fields, from healthcare and finance to social networks and e-Commerce. However, this rapid digital transformation has also introduced critical challenges related to data privacy, security, and ownership [5]. Centralised systems often consolidate vast amounts of personal data, which makes them attractive targets for cyber-attacks and breaches. The privacy and security risks associated with the storage of large amounts of personal information in central databases have become a major concern [106].

The implementation of data protection and privacy regulations including the General Data Protection Regulation (GDPR) in Europe has imposed strict requirements on companies regarding the collection, storage, and processing of personal data. Although these regulations are designed to protect user privacy and ensure data security [179], adhering to these regulations has become expensive. For instance, compliance costs for GDPR are estimated to be around $9 billion for Fortune 500 companies [93].

By leveraging the transparency, decentralisation and immutability characteristics of blockchain technology, we can develop a system that enables individuals to retain control over their data and reap direct benefits. This decentralised approach enhances data privacy and promotes economic fairness by allowing users to have greater autonomy and ownership of

their personal information. Despite its potential, the widespread adoption of blockchain technology faces significant hurdles particularly concerning privacy and scalability. A study by Deloitte found that 68% emphasise data privacy amongst the top three greatest areas to facilitate adoption of blockchain and digital assets [92]. While blockchain's transparency ensures data integrity, it also raises concerns about data confidentiality, especially in scenarios where sensitive personal information is involved. To address these issues, researchers have explored various privacy-preserving techniques, such as Zero-Knowledge Proofs (ZKP) [144], Attribute-Based Encryption (ABE) [82], Secure Multi-Party Computation (SMPC) [202] and couple more [94].

ZKP techniques, in particular, have gained attention for their ability to prove the validity of a statement without revealing the underlying information[80]. This cryptographic technique is considered crucial to improve privacy in blockchain-based systems, enabling secure data sharing without compromising confidentiality [126, 94]. Ciphertext-Policy Attribute Based Encryption (CP-ABE), a type of ABE enhances by allowing fine-grained access control to encrypted data based on user attributes, providing a flexible and secure mechanism for data management. This technique is aimed at decentralised environments which enable secure and controlled data sharing without revealing sensitive information [22]. These privacy-enhancing technologies form the foundation for systems that help address the inherent trade-offs between transparency and confidentiality in blockchain technology [94].

1.1 Problem Statement

Existing frameworks for secure data sharing, mainly those that handle sensitive personal or organisational data, predominantly employ traditional role-based or attribute-based access control mechanisms with cryptographic schemes such as ABE techniques [102, 70]. Although ABE provides a dynamic method for controlling access based on user-defined at-

tributes (e.g., age, professional role, or security clearance), **these systems often require users to disclose their attributes during validation.** This mandatory disclosure introduces significant privacy concerns, as it exposes potentially sensitive information to third parties, even when it is solely intended for access verification [152, 132].

For instance, in healthcare data-sharing platforms, a doctor may be required to present their credentials and medical specialisation to gain access to specific patient records [102, 69]. These attributes are shared with a data provider or a centralised authority for validation which creates possible scope for vulnerabilities. Such disclosures increase the risk of attribute leakage, unauthorised surveillance, and misuse, where external entities could track attribute usage or profile the data processor.

The current reliance on attribute revelation for decryption validation **not only compromises privacy but also magnifies the potential for unauthorised access and tracking of user behaviour.** Therefore, there is an urgent need for an enhanced solution that ensures secure data sharing while preserving the privacy of the user's attributes, eliminating the need for their disclosure during access validation.

1.2 Research Gap and Research Questions

Literature review is carefully conducted that help revealing a significant gap in addressing attribute disclosure risks within CP-ABE systems. Although ZKP integration offers a theoretical solution, practical implementations are scarce and existing approaches often lack scalability and efficiency [81]. There is a need for a comprehensive framework that seamlessly combines CP-ABE and ZKP in blockchain-based data-sharing systems.

That leads to the research questions:

- **RQ1:** ***How to preserve privacy in decentralised Blockchain-based Data Sharing System***
- **RQ2:** ***How to utilise ABE to strengthen decentralised Blockchain-based Data Sharing Systems***
- **RQ3:** ***How to integrate ZKP in CP-ABE based blockchain-based data sharing systems to mitigate the risks of attribute disclosure***

1.3 Research Aims and Objectives

Addressing the risk of attribute disclosure in CP-ABE systems is crucial to the adoption of blockchain-based data sharing solutions in privacy-sensitive domains. In recognition of the need to safeguard sensitive personal data and attributes, this research **aims to propose a novel Zero-Knowledge Ciphertext-Policy Attribute-Based Encryption (ZK CP-ABE) system** for data-sharing platforms based on blockchain technology.

The combination of ZKP variants and CP-ABE would result in a promising avenue to enhance privacy without sacrificing the benefits of fine-grained access control. However, the combination does not come straightforward, as it requires non-trivial tasks to seamlessly integrate user' attributes with cryptographic primitives into ZKP proofs, particularly when employed into a Blockchain system.

The **ultimate objective of the research** is to propose a solution for bridging the research gap by **seamlessly combining ZKP with ABE employed in a Blockchain platform**. The solution would be the foundation for developing an efficient and practical framework for a blockchain-based data sharing system.

The proposed solution would inherit the strengths of blockchain technology, ZKPs, and ABE. The solution should ensure that only authorised parties can access specific records whilst concealing the data processor's attribute required for successful retrival of data. As a result, it prioritises the privacy of data owners and guarantees the preservation of data requester attributes. The results of the research would contribute to the advancement of secure and privacy-preserving data sharing technologies.

Summarising research objectives as follows:

- To develop a scalable privacy-preserving data sharing system.
- To demonstrate how ZKPs can eliminate the need for attribute disclosure during access validation.
- To evaluate the effectiveness of the proposed ZK CP-ABE system in real-world scenarios with dynamic policies.

1.4 Contribution

This research makes the following contributions:

- Proposed a scalable privacy-preserving data-sharing system that combines ZKP and CP-ABE to address attributes' disclosure risks.
- Implemented a decentralised storage solution leveraging blockchain and IPFS for enhanced data integrity and confidentiality.
- Performed comprehensive performance evaluations to demonstrate system adaptability with dynamic policies and attribute sets.

1.5 Thesis Organisation

The remainder of this thesis is organised as follows.

- *Chapter 2* provides a comprehensive understanding of blockchain technology, its fundamental characteristics such as decentralisation, transparency and immutability. It delves into the technical aspects of blockchain technology which focuses on the Ethereum platform, smart contract functionality, and associated security and privacy challenges. The chapter also explores IPFS as a decentralised storage solution.
- *Chapter 3* reviews various privacy-preserving techniques and literature on blockchain-based systems, including mixing services, ring signatures, and homomorphic encryption, and presents the literature review.
- *Chapter 4* presents the problem statement and proposed solution for a ZK CP-ABE system, detailing its architecture and technical stack.
- *Chapter 5* covers system implementation and analysis, including encryption, key generation, decryption, and performance testing.
- *Chapter 6* concludes the thesis, summarising the findings and suggesting future research directions.

Chapter 2

Fundamental Background

2.1 Blockchain Technology

Satoshi Nakamoto [131] is recognised as the originator of Bitcoin and its foundational blockchain technology. A groundbreaking decentralised system designed to enable secure operations in networked environments without a central authority. All transactions or messages exchanged between network nodes are recorded in a blockchain system. Once network nodes verify these transactions through a consensus protocol requiring majority agreement, they are permanently stored on the blockchain, ensuring the immutability of the data and providing a complete traceable history of all transactions [184]. From a data management point of view, the blockchain functions as a distributed database, maintaining a continuously growing repository of transaction records [89]. These records are systematically organised into interconnected blocks [94].

A blockchain system is typically a P2P network characterised by two types of nodes: miners and users. The freedom to assume either role lies with each node. Miners are pivotal to the blockchain infrastructure that maintains the operation of the P2P network [43]. The following provides an in-depth exploration of the salient components of blockchain technology.

- *Blockchain as a Distributed Ledger:* A distributed ledger is a decentralised database that records all blockchain data in a standardised format and is managed by all participating miners. It encompasses a series of interconnected blocks in a chain linked using a **cryptographic hash function** h. Each block B_i is added and maintained chronologically and is identifiable by its unique **hash value** $h(B_i)$, which serves as the block address. The hash of each block B_i depends on the previous block B_{i-1}, forming a chain, which can be expressed as:

$$h(B_i) = h(B_{i-1} \| T_i)$$

  where T_i represents the transaction data in block B_i, and $\|$ denotes concatenation. The block header incorporates the current version number, the previous block's hash value, its block address, the Merkle root hash, and the creation timestamp, as shown in figure 2.2 [89]. A consensus mechanism, such as PoW [131], PoS [133], are employed, containing a nonce that confirms the computational correctness of the block generation. The block body permanently records all confirmed transactions in the blockchain, which are structured as a Merkle tree, organising all transactions for efficient querying and verification [155].

  The onus of maintaining the distributed ledger falls on the miners. They can both access and write data into the ledger. Before any data are recorded, their validity must be confirmed via the consensus mechanism. While users can access data, they can only write and append data to the blockchain with miners' assistance [94].

- *Consensus Mechanisms:* The consensus mechanism is a fault-tolerant process that allows multiple parties to agree on a single data value or network state [189]. It sustains the blockchain data's authenticity, consistency, and order across the network. The consensus mechanism resolves the Byzantine general's problem [29] that in a distributed system with numerous nodes. These nodes must reach an identical decision despite disloyal nodes. As illustrated in 2.1,the consensus process involves participating nodes broadcasting transactions while validator nodes verify and validate them before reaching an agreement.

Consensus mechanism guarantees the blockchain's verifiability and tamper-resistance by obtaining approval from these nodes in which mining is the core process to reach consensus on a newly created block via the blockchain, providing system liveness and safety [94]. Familiar consensus mechanisms include PoW [131], PoS [33], DPoS [180], PoA [51], and PoH[199].

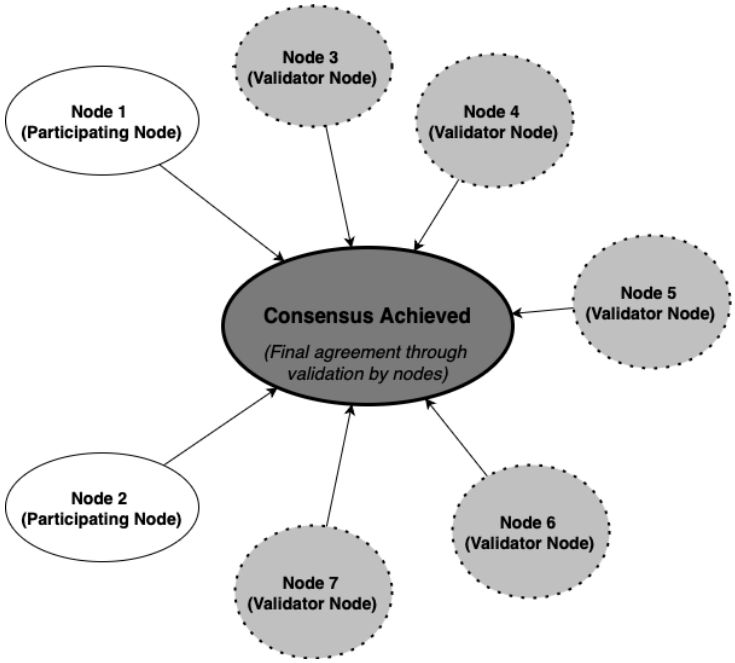

Figure 2.1: Consensus mechanism where Participating Nodes broadcast transactions, and Validator Nodes verify them. Consensus is achieved when a majority of Validator Nodes validate the transaction.

Typically, the security of a blockchain system, like Bitcoin, is contingent upon its consensus model. The security of consensus is premised on the assumption of an honest majority which means the majority of consensus voting power is honest. Bitcoin incorporates an incentive mechanism to motivate miners to create new blocks which contributes to the blockchain's resilience and, based on game theory, augments the blockchain's security and privacy [94].

2.1.1 Blockchain Characteristics

This subsection discusses the uniqueness of blockchain with respect to its core features such as decentralisation, autonomy, transparency, non-repudiation, and immutability.

2.1.1.1 Decentralisation and autonomy

Decentralisation and autonomy are quintessential features of blockchain technology, breaking away from conventional centralised systems.

The decentralised nature of blockchain technology implies that no single entity controls the entire network [7]. Instead, the network is maintained collectively by all nodes participating, often known as 'peers'. Each peer has a copy of the entire blockchain and participates in the validation of transactions.

This decentralised architecture contributes to the robustness and security of the system. Since no central authority can be compromised, the system is resilient to various attacks, including denial-of-service attacks. Moreover, the absence of a central authority eliminates the need for trust in a single entity, as the integrity of the system is maintained by the consensus of the network participants [94].

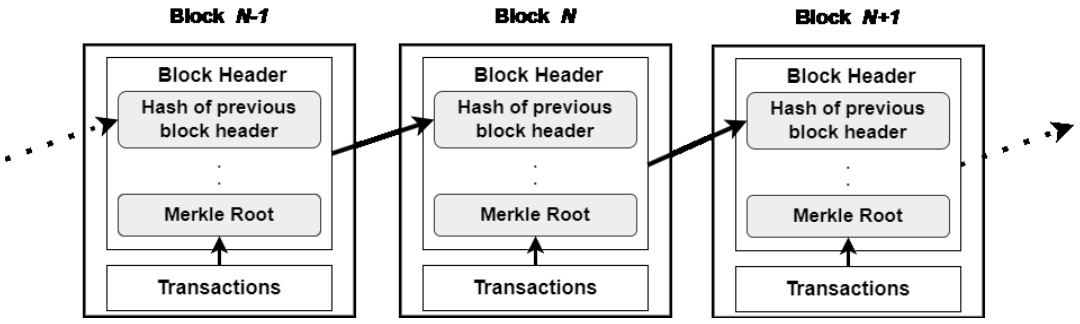

Figure 2.2: Overview of the blockchain data structure consisting of blocks linked to each other using cryptographic hashes of the previous blocks [94].

The autonomous nature of blockchain technology emerges from its use of smart contracts [94]. They are self-executing contracts with the terms of the agreement directly written into code. The code and agreements exist on a decentralised blockchain network, ensuring their execution is automatic , traceable and irreversible [116]. This minimises manipulation, bias, or error and improves efficiency by removing intermediaries [94].

Decentralisation and autonomy offer significant advantages in security and efficiency, but they also present challenges [94]. The autonomous execution of smart contracts can lead to unintended consequences if there are flaws in the code [159]. These challenges necessitate ongoing research and development in the field [94].

2.1.1.2 Transparency

Information recorded in a blockchain (i.e., on-chain data) is transparent to all nodes in the blockchain network, and users can conveniently access this on-chain data by querying miners [94, 89]. Transparency enhances data immutability and verifiability because all nodes can detect illegal modification and data. Transactions are **cryptographically linked** using **hash functions**. If one block B_i is altered, the hash of the block B_{i+1} will change. This results in creating a chain reaction that affects all subsequent blocks. The relationship can be expressed as:

$$h(B_{i+1}) = h(B_i \| T_i)$$

where any alteration in B_i or T_i will invalidate the hash of block B_{i+1}, making tampering detectable. Nevertheless, privacy leakage due to transparency has become a crucial issue that dramatically limits blockchain applications [21, 94].

Each transaction on a blockchain is time-stamped, immutable, and linked to the previous transaction, forming a chain of blocks. Any alteration in a block impacts all subsequent blocks, making unauthorised changes nearly impossible without detection [21, 94]. This immutability feature enhances transparency and trust in the system, as participants can verify the integrity of transactions without relying on third-party intermediaries [214, 94].

2.1.1.3 Non-repudiation

Non-repudiation is an inherent property of blockchain technology that ensures irreversible and undeniable proof of participation in a transaction or event [66, 94]. The **cryptographic digital signature** used in transactions provides this guarantee. A transaction signed with a private key k_{priv} can be verified using the corresponding public key k_{pub}:

$$\text{Verify}(h(T), \sigma, k_{pub}) = \text{True}$$

where T is the transaction, $h(T)$ is the hash of the transaction, and σ is the signature generated using k_{priv}. The non-repudiation of Blockchain refers to (i) No one can deny transaction contents created by himself and (ii) No one can repudiate the transaction time generated by himself. This characteristic is vital for creating a trustless environment where participants can confidently interact, knowing their actions cannot be denied later. Non-repudiation is achieved in blockchain technology through the use of cryptographic digital signatures [216, 94]. In a transaction process, the sender signs the transaction with their private key, which anyone can verify using the sender's public key. This signature serves as strong proof of the origin and integrity of the transaction. Once the transaction is included in a block and the block is appended to the blockchain, the transaction becomes immutable and undeniable [216, 94].

Due to the characteristic of non-repudiation, if a transaction exists in the Blockchain, it must be initiated by its signer itself. The node cannot deny that it has published the transaction [94]. The distributed nature of blockchain technology ensures that all nodes in the network hold a copy of the blockchain, and this duplication further bolsters non-repudiation. Suppose a participant attempts to deny their action. In that case, other participants can refer to their local blockchain copies to verify the action's occurrence [216].

Non-repudiation, while offering an added layer of security and trust in blockchain transactions, raises several implications, particularly privacy. Once a transaction is committed, it is permanently recorded and openly verifiable. This feature presents challenges for privacy-concerned users who may need to obfuscate their activities to maintain their privacy, bringing forth anonymity and pseudonymity in the blockchain [94].

2.1.1.4 Verifiability and immutability

Verifiability means that the validity of each transaction in the blockchain can be verified and cannot be modified or removed from the blockchain [134]. Since all miners confirm the blocks in which transactions are recorded via the consensus mechanism, invalid transactions will not be recorded in the blockchain. Any data modification in the blockchain will be denied unless the adversary compromises the whole system. Also, blocks are organised in a chain using the hash function, which makes any modification to the data easily detected. This characteristic benefits security but also results in the problem that sensitive data cannot be removed from the blockchain [94]. Immutability means whose state cannot be altered after its creation. Immutable transactions make it impossible for any entity to manipulate and falsify data stored on the network [134, 94]. This is due to the **chaining of blocks** via hash functions:

$$h(B_{i+1}) = h(B_i \| T_i)$$

If B_i is modified, the hash $h(B_{i+1})$ would change, and the blockchain would reject the altered data, ensuring the integrity of past transactions. Since historical transactions can be audited at any point, immutability enables high data integrity [94].

2.2 Comparative Analysis: Centralised vs. Decentralised Systems

Traditional centralised systems have long served as the foundation for data storage and access management [64]. These systems rely on a central authority such as a cloud provider, database, or identity management service to store, verify, and regulate access to sensitive information. While centralized solutions provide efficient processing and familiar architectures, they introduce several challenges, including *single points of failure, vulnerability to cyberattacks, lack of transparency, and high operational costs* [64].

In contrast, decentralised systems decentralise data storage and access control by distributing records across multiple nodes, ensuring *tamper-proof security, automated policy enforcement through smart contracts, and enhanced privacy mechanisms.*

The following table provides a detailed comparison of key aspects of centralised and decentralised systems:

Table 2.1: Comparison of Centralised Systems and Decentralised Systems

Feature	Centralised Systems	Decentralised Systems
Architecture	Operable under a single entity (e.g., cloud provider or enterprise database) [64].	Decentralised ledger distributed across multiple network nodes [94].
Access Control	RBAC/ABAC enforced by a central authority [64].	Smart contracts enable fine-grained, self-enforcing access control without intermediaries [94].
Security	Prone to cyberattacks, insider threats, and data leaks due to central control [64].	Enhanced security through cryptographic hashing, consensus mechanisms, and distributed trust.
Data Integrity	Admins or malicious actors can modify or delete records [64].	Immutable records prevent unauthorised changes and all modifications are cryptographically verifiable [94].
Transparency and Auditability	Logs can be modified or erased, making tracking and verification difficult [64].	Blockchain records are tamper-proof and fully transparent, ensuring a reliable audit trail [94].
Scalability and Interoperability	Data silos limit seamless integration across different platforms, requiring complex and costly infrastructure upgrades to scale [64].	Decentralised architecture enables efficient cross-platform data sharing and scaling without reliance on a central authority [94].
Non-Repudiation	Actions can be denied or manipulated due to centralised control [64].	Cryptographic signatures ensure non-repudiation which makes all transactions irrefutable [94].

2.3 Ethereum platform

Ethereum, launched in 2015, is an open-source, blockchain-based platform that enables the development and execution of smart contracts [33]. The earlier cryptocurrency, Bitcoin, influenced the design of Ethereum, but with significant enhancements to extend its functionalities beyond just a peer-to-peer electronic cash system.

Ethereum has revolutionised the blockchain landscape by introducing the concept of programmable contracts known as ”smart contracts”. This platform enables developers to create decentralised applications that operate on a blockchain,thereby leading to trustless and transparent interactions over the internet [50].

In Ethereum, users can create accounts, transfer their native ETH cryptocurrency, and interact with smart contracts. Smart contracts are immutable and autonomous scripts stored on the Ethereum blockchain that execute predefined functions when certain conditions are met [50]. These smart contracts are pivotal in allowing DApps to be built on Ethereum, making it more than just a platform for cryptocurrency transactions. One notable feature of Ethereum is the introduction of the EVM, a runtime environment that executes smart contracts. This makes Ethereum a general-purpose blockchain, unlike Bitcoin, designed with a specific use case [50].

In parallel to Bitcoin, Ethereum offers its users a degree of anonymity through its use of pseudonymous addresses linked to each account. These addresses bear no explicit identifiers tying them to the users' real-world identities, offering a level of privacy that is essential for many [182]. Nevertheless, this feature can be a double-edged sword as it opens avenues for potential illicit activities [209].

2.3.1 Ethereum Virtual Machine (EVM)

One prominent feature of Ethereum is the EVM, which is a Turing-complete runtime environment designed to execute smart contracts. The EVM abstracts the underlying hardware, enabling developers to write and deploy smart contracts using various programming languages, including Solidity [50].

The EVM operates as a decentralised global computer, processing and executing smart contracts securely and deterministically across the entire Ethereum network. Each operation within the EVM requires a certain amount of gas, a unit of computational work which must be paid for with Ether. This mechanism ensures that computational resources are allocated efficiently and that contracts are executed to prevent infinite loops or other forms of abuse [34]. Mathematically, if a contract execution requires n operations, the total gas cost can be represented as:

$$\text{Total Gas} = \sum_{i=1}^{n} \text{Gas}_i$$

where Gas_i is the gas required for the i^{th} operation.

2.3.2 Smart Contracts

A smart contract platform is software that runs on a blockchain, extending its functionality and broadening its application. Smart contracts are programs stored on a blockchain that execute when certain predetermined conditions are fulfilled [33]. They are used to automate the execution of an agreement so that all participants can instantly ascertain the outcome, thus eliminating any need for an intermediary or wasting time. Workflow automation is possible, triggering the subsequent action when conditions are met [33].

Smart contracts are defined in many ways. Szabo first proposed that *'Smart contracts are a computable transaction protocol to execute contract terms'* [171]. Ethereum's smart contract is a digital asset control program based on the blockchain [23]. In a narrow sense, a smart contract is a program code involving business logic, algorithms, complex relationships among people, legal agreements, and networks. A smart contract is a computer protocol that can self-execute and self-verify after deployment [171].

The operation of smart contracts involves three procedures: contract creation, contract publishing, and contract execution [191]. During contract creation, the contract participants will negotiate to clarify parties' rights and obligations, determine the standard contract text, and then program them into a smart contract program [191]. The contract program generally requires auditing for secure execution. In contract publishing, the contract creator signs and requests a miner to record the signed contract into the blockchain. The contract execution is based on an event-triggered mechanism on the blockchain, which encompasses transaction processing and preservation mechanisms and is a complete state machine [191]. Specifically, the external nodes can interact with a smart contract program by sending transactions. The transactions can change the status of the contract [118]. All miners monitor the status, and once they detect its change, they execute the smart contract based on its design [33, 191].

2.3.3 Tokens and ERC Standards

In addition to enabling smart contracts, Ethereum also supports creating tokens and digital assets built on the Ethereum blockchain. These tokens can represent a wide variety of assets, from currencies to property ownership, and can be easily transferred between users on the blockchain .

Ethereum has introduced several standards to facilitate the creation and management of tokens, the most prominent of which is the ERC-20 standard [138]. The ERC-20 standard defines a common set of rules that an Ethereum token must implement, making it easy for tokens to be traded, exchanged, or used within Ethereum-based applications. The key functions defined by ERC-20 include [138]:

- *totalSupply():* Returns the total supply of tokens.
- *balanceOf(address):* Returns the token balance of a specific address.
- *transfer(address, uint256):* Transfers a specified number of tokens to a given address.

- *approve(address, uint256):* Approves another address to spend a specified amount of tokens on behalf of the token holder.
- *transferFrom(address, address, uint256):* Transfers tokens from one address to another, typically used in conjunction with the *approve* function.

The ERC-721 standard defines the rules for non-fungible tokens (NFTs), which are unique digital assets that can represent ownership of a specific item or piece of content [139]. Unlike ERC-20 tokens, which are fungible and identical to each other, each ERC-721 token is unique and can have unique properties and value making it suitable for applications such as digital art, collectibles, and real estate.

In ERC-721, each token is represented by a unique ID, and the ownership and transfer of these tokens are managed through functions such as [139]:

- *ownerOf(uint256):* Returns the owner of a specific token ID.
- *transferFrom(address, address, uint256):* Transfers ownership of a specific token ID from one address to another.
- *approve(address, uint256):* Approves another address to transfer a specific token ID on behalf of the owner.

Mathematically, if a user owns a token with ID t, the ownership can be represented as:

$$\text{ownerOf}(t) = A$$

where A is the address of the current owner. If the token is transferred to another address B, the ownership changes to:

$$\text{ownerOf}(t) = B$$

2.3.4 Security and Privacy in Ethereum

This section examines the security and privacy concerns on Ethereum platforms. While the fundamental structure of blockchain provides notable security measures, Ethereum's unique characteristics present specific challenges in protecting user data from malicious attempts.

2.3.4.1 Security

Security in the blockchain is based on the following factors. First, blockchain technology relies on a decentralised ledger to keep track of all financial transactions. The "master" ledger would be a point of vulnerability [13]. If the ledger was compromised, then it could lead to a system breakdown. Secondly, the ledger exists as a long chain of cryptographically encrypted sequential blocks, reducing the risk of data tampering. Blockchain consists of hundreds to thousands of unique nodes. Every node has a complete copy of the digital ledger. The nodes can work independently for the verification of a transaction. If the nodes do not agree, then the transaction is cancelled [94].

Thirdly, The cryptographic keys and two key systems used in blockchain exchanges are very long, complex and difficult to decipher unless one has the authorisation to view the keys. The public key and private key in public-key cryptography. Both of these keys are generated using the Elliptic Curve cryptography method [94]. Firstly, it creates the private key, and then a public key from the private key is created using ECDSA [99]. Therefore, private and public keys are cryptographically and mathematically linked. Therefore security in blockchain ensures *confidentiality* , *availability*,*integrity* and *ledger consistency* [94].

However, although the security embedded, blockchain is prone to attacks such as double spending, wallet-based attacks (i.e., client-side security), network-based attacks such as DDoS, Sybil, and eclipse and mining-based attacks such as *51%* [203], block withholding and bribery [84, 55, 165]. As a prominent platform for decentralised applications, Ethereum presents unique security challenges stemming from its characteristics. These challenges can broadly be categorised into protocol-level challenges, smart contract vulnerabilities, and network-level threats as follows [94]:

- *Protocol-Level Challenges*: Like all blockchain systems, Ethereum is subject to attacks that attempt to manipulate the consensus protocol. Examples include 51% attacks, where a malicious entity with control over the majority of the network's hashing power can manipulate the blockchain, and eclipse attacks, where a node is isolated from the rest of the network and fed false information [65].
- *Smart Contract Vulnerabilities*:Smart Contract known as Ethereum's distinctive features is also known to introduce unique security issues. These powerful programmable contracts have been a common target for attackers due to vulnerabilities in their code. Notable attacks exploiting smart contract vulnerabilities include the DAO and Parity wallet incidents. Detecting and eliminating such vulnerabilities before contract deployment remains challenging due to the Turing completeness of Ethereum's programming language, Solidity, and the immutability of deployed contracts [110].
- *Network-Level Threats*: Ethereum is also subject to common network-level attacks as part of the Internet. These include DDoS attacks, Sybil attacks, and routing attacks. Effective defense mechanisms are needed to maintain the robustness and reliability of the Ethereum network [14].

2.3.4.2 Privacy

Data privacy of blockchain refers to the property that blockchain can firstly provide *Anonymity* where it is the state of being anonymous and unidentified [84, 87, 67] and secondly ensure *Unlinkbaility* where users' transactions related to themselves cannot be linked [163, 158, 67, 94]. Despite its innovative applications and significant potential, Ethereum grapples with inherent privacy challenges. The issues pertain to the system's pseudonymous nature, transaction transparency, and the interaction of smart contracts mentioned below:

- *Pseudonymity and Linkability*:Ethereum accounts are pseudonymous and transactions are publicly visible,leaving room for the potential deanonymisation of user accounts through data analysis. An attacker can link addresses to identify a unique user or organisation and then analyse their financial behaviour. Although this transparency is necessary for ensuring the system's integrity, it presents significant privacy concerns [34].
- *Transaction Transparency*: Every transaction on the Ethereum blockchain is visible to every participant in the network [136]. While this provides essential advantages, it can lead to privacy leaks. Information about the value transferred, the parties involved, and even the timing of the transaction can reveal sensitive data [124].
- *Smart Contract Privacy*: Smart contracts, a distinctive feature of Ethereum, have exacerbated privacy concerns [103]. The logic and state of a smart contract are visible to all participants, potentially revealing sensitive business logic or private data. Moreover, interaction with smart contracts can leak information about the parties involved.

While these challenges exist, it is essential to note that continuous efforts are being made by Ethereum developers and the wider blockchain community to identify and mitigate these security and privacy threats. Addressing these issues is crucial for adopting and surviving Ethereum as a secure platform for decentralised applications [94].

2.4 InterPlanetary File System

IPFS is a peer-to-peer distributed file storage protocol designed to create a more resilient, efficient, and decentralised internet [18]. Introduced by Juan Benet in 2015 [18], IPFS addresses the limitations of traditional HTTP-based web architecture by providing a content-addressable, versioned, and decentralised approach to file storage and sharing [18, 100].

At its core, IPFS allows users to store and share files in a distributed manner without relying on centralised servers. This is achieved through a global network of nodes, each contributing to the storage and retrieval of data [49]. Files in IPFS are broken down into smaller chunks, which are then cryptographically hashed and distributed across multiple nodes in the network [18]. Each file is identified by its unique cryptographic hash, a CID and a permanent, immutable link to its content which irrespective of location [18].

Key features of IPFS include:

- *Hashing:* Files in IPFS are divided into smaller blocks, and each block is hashed using a cryptographic hash function, typically SHA-256. If a file F is split into n blocks $\{B_1, B_2, \ldots, B_n\}$, each block is hashed to produce a set of hashes $\{h(B_1), h(B_2), \ldots, h(B_n)\}$ [18]. The overall CID for the file can be represented as a Merkle Root of these block hashes:

  $$\text{CID} = h(h(B_1) \| h(B_2) \| \ldots \| h(B_n))$$

  where $\|$ denotes concatenation and $h(\cdot)$ denotes the hash function.
- *Content Addressing:* Instead of using location-based addressing like URLs in HTTP, IPFS uses content addressing. The file is retrieved using its CID, derived from the content's cryptographic hash. The CID ensures that the content is unique and remains accessible as long as at least one node in the network hosts the file [18].

- *Distributed Storage:* Files are distributed across multiple nodes in the IPFS network. Let N be the set of nodes, and B_i be a block of the file. Each node $n_j \in N$ that stores B_i contributes to the redundancy and availability of the file. The distribution of the file blocks follows a probabilistic model to ensure fault tolerance and minimise data loss[18].
- *Versioning and Immutability:* IPFS supports versioning by linking different versions of a file through their respective CIDs. If a file is updated, the new version receives a new CID, while the previous versions remain accessible. The immutable nature of IPFS ensures that once data is stored, it cannot be altered or tampered with, which can be mathematically represented by the immutability of the hash function [18]:

$$h(F) \neq h(F') \quad \text{for} \quad F \neq F'$$

  where F and F' are two different versions of a file.

IPFS has been widely adopted in various applications, particularly with blockchain technology [18, 100, 108]. Its ability to store large amounts of data off-chain while maintaining verifiable links to on-chain records makes it an ideal solution for DApps. For example, IPFS is used in conjunction with Ethereum to store large files like documents, images, and videos while only storing the cryptographic hash on the blockchain. This reduces on-chain data storage costs and improving scalability [108].

Moreover, IPFS plays a crucial role in decentralised CDNs, enabling efficient and censorship-resistant content distribution worldwide [169]. It also supports collaborative platforms where multiple users can contribute to and access shared datasets without relying on a central authority.

Despite its many advantages, IPFS faces challenges such as incentivising node participation, ensuring data availability, and managing content retrieval speeds in large-scale networks [54]. Projects like Filecoin , a decentralised storage network built on top of IPFS, aim to address these challenges by introducing economic incentives for users to provide storage and retrieval services [169].

IPFS is a major advancement in decentralised file storage and content distribution, providing a scalable, secure, and resilient alternative to traditional web architectures. Its integration with blockchain platforms enhances the potential for creating truly decentralised applications that rely on distributed storage and immutable data [18, 100].

Chapter 3

Literature Review on Privacy-Preserving Blockchain Systems

Blockchain technology, introduced by Satoshi Nakamoto in 2008 [131], has emerged as a revolutionary framework for decentralised data management and secure transactions. Its core features—decentralisation, immutability, and transparency—provide a robust foundation for building trustless systems where data integrity and provenance are guaranteed without the need for intermediaries [189]. Moreover, since the introduction of Blockchain, various privacy-preservation techniques have continuously been adopted and integrated to enhance the confidentiality and anonymity of users' identities and transaction data on the networks [94].

3.1 An Overview

In this chapter, we delve into the significant milestones in the evolution of privacy preservation in blockchain systems, particularly Ethereum, from the implementation of Ring Signatures in 2015 to more advancements, such as Mixing Services and Differential Privacy.

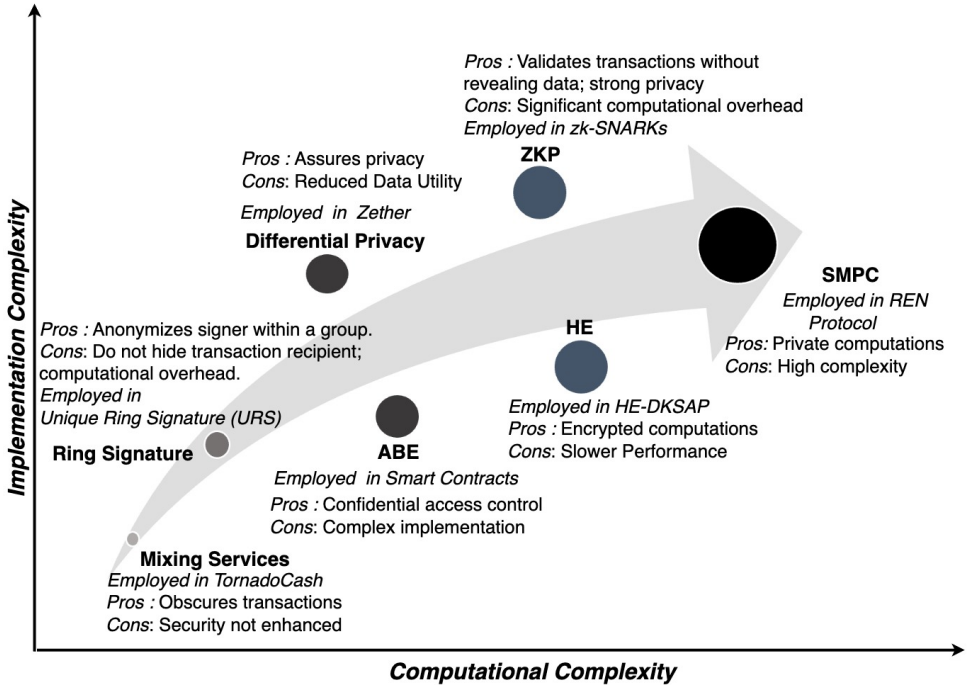

Figure 3.1: Utilisation of privacy-preserving techniques in Ethereum has been analysed through the lens of computational and implementation complexities since its inception.

In the journey towards enhancing privacy on the Ethereum blockchain, as seen in Figure 3.1 [94], several implementations of these privacy-preserving techniques stand out. To name a few, introducing a unique ring signature scheme using *secp256k1* elliptic curve that provides anonymity for signers within a group [123]and integrating ZKPs to bolster scalability and confidentiality in transactions via zk-SNARKs [128]. Ethereum's privacy mechanisms further expanded with Tornado Cash, a decentralised, non-custodial mixing service that uses smart contracts and zero-knowledge proofs to enhance transaction privacy [145]. Tornado Cash breaks the on-chain link between sender and receiver addresses, providing a higher degree of anonymity and privacy for Ethereum transactions [145, 94].

Additionally, SMPC is used in REN Protocol, an open protocol built to provide interoperability and liquidity between different blockchain platforms [172]. ABE technique in Ethereum is the integration of ABE with smart contracts for fine-grained access control, where data access can be precisely controlled according to a set of attributes, such as user roles, permissions, or other criteria, without revealing the identities of the users involved, particularly relevant for use cases where sensitive data is involved, such as in healthcare [98]. Lastly, the introduction of HE techniques allowed for computations on encrypted data, preserving the privacy of the underlying information [200, 94]. The Zether protocol brought Differential Privacy to Ethereum, providing statistical privacy by obscuring individual data within aggregated data sets [32]. Collectively, these advancements have been seen to contribute to the robust framework of privacy within the Ethereum ecosystem. These privacy-preservation techniques are briefly summarised in Table 3.1 [94] with their pros and cons.

3.2 Mixing Services

Mixing services are essential tools in the blockchain ecosystem for enhancing user privacy, particularly in addressing the non-anonymous nature of cryptocurrencies such as Bitcoin. Despite employing pseudonymous addresses, the public nature of Bitcoin transactions allows for the analysis and correlation of a user's transactions. Mixing services, such as Tornado Cash on the Ethereum platform [145], obfuscate transaction trails to impede address linkage. However, they do not protect against coin theft [40]. Notable mixing services include MixCoin [30], CoinJoin [52], and Private CoinJoin as implemented in DASH [57]. These services play a pivotal role in preserving user anonymity in the blockchain. Several notable mixing services have been proposed as follows:

Technique	Pros	Cons
Mixing Services [145, 30, 52, 119, 57, 9, 24, 161, 194, 195]	Efficiently obscures transaction trails, making linking addresses and tracing transactions back to users difficult.	Enhances privacy but does not necessarily increase security against theft or loss of cryptocurrency.
Ring Signature [17, 72, 166, 166, 115, 3, 208, 181, 47, 135, 42]	Ensures anonymity of the signer within a group, thereby effectively shielding the sender's identity in a transaction.	Does not conceal the transaction amounts or the recipient's address. Computational overhead from larger ring sizes can affect transaction processing times.
ABE [197, 192, 98, 28, 39, 117, 129, 85, 149, 90, 198, 201, 140]	Supports both confidentiality and access control and has potential for decentralised multi-authority systems.	Limited adoption of certain DApps due to complexity and implementation challenges.
SMPC [111, 122, 212, 215, 20, 6, 20, 219]	Enables confidential computations involving multiple entities while ensuring data privacy among participants.	Computationally complex, potentially increasing network latency, which affects blockchain network efficiency.
ZKP [144, 31, 79, 144, 147, 83, 26, 88, 128, 141]	Allows validating blockchain transactions without revealing sensitive data and offers strong cryptographic privacy guarantees.	Introduces computational overhead, especially in non-optimised implementations.
DP [8, 60, 61, 1, 101, 73, 120, 86, 71, 109, 97, 95, 142]	Offers strong mathematical assurances against data breaches, crucial for protecting individual privacy in sensitive DApps data processing or sharing.	May reduce data utility, impacting effectiveness of data-driven decisions in blockchain applications. Privacy levels depend on careful parameter configuration to balance privacy without significantly reducing data usefulness.
HE [76, 53, 130, 185, 107, 185, 160]	Enables computation on encrypted data while preserving privacy, ideal for DApps managing and processing sensitive data.	Complex and computationally intensive, resulting in slower performance, less scalability, and impracticality for smart contracts.

Table 3.1: Comparative analysis of privacy-preservation techniques in blockchain platforms [94].

1. MixCoin: Introduced by Bonneau [30], MixCoin aims to provide anonymous transactions for Bitcoin and similar cryptocurrencies. It counters passive adversaries by broadening the anonymity set, facilitating simultaneous coin mixing by all users. Against active adversaries, MixCoin offers anonymity akin to traditional communication mixes. Significantly, MixCoin incorporates an accountability mechanism, deterring coin theft by aligning user incentives and fostering rational usage.

2. CoinJoin: Conceived in 2013, CoinJoin offers an alternative for anonymising Bitcoin transactions [52, 119]. Rooted in the concept of joint payments, CoinJoin allows a user to collaborate with another user to execute a collective payment within a single transaction. Such joint payments considerably diminish the likelihood of tracing input-output links or discerning a specific user's monetary flow direction. Early mixing services employing this approach, like SharedCoin [9], relied on centralised servers. However, such centralisation, while simplifying the process, introduced trust issues. Users had to trust these service operators to safeguard the bitcoins and not retain transaction logs, which would undermine the privacy efforts.
3. Private CoinJoin: DASH has implemented an advanced version of CoinJoin [57] providing privacy features through its *PrivateSend* function [57, 24, 52]. *PrivateSend* is an optional feature in DASH that blends multiple transactions, making it substantially challenging to determine the source, destination, and amounts in individual transactions. This CoinJoin inspired method allows users to utilise the increased privacy feature without making it mandatory for all DASH transactions [52].

Mixing services represent a pivotal advancement in preserving user anonymity within the blockchain realm, especially given the transparency challenges posed by cryptocurrencies like Bitcoin. While they mask transactional histories, inherent vulnerabilities remain, particularly regarding coin security. As blockchain technology progresses, a deep understanding of these services' pros and cons becomes crucial for developers and users [161].

3.3 Ring signature

Ring signatures, introduced by Rivest, Shamir, and Tauman [151], represent a sophisticated approach to digital signatures that allow a member of a group to sign messages anonymously. An example of ring signatures in practical use is in anonymous voting systems [17], where they ensure voter anonymity while maintaining vote integrity. As Figure

3.2 illustrates, the ring consists of several members, any of whom could be the actual signer, depicted by the figures around the "RING". This ensures that while the signature validates the message as originating from the group, the actual author's identity remains concealed, depicted by the detached figure holding the *"Signature Value"* [94].

Such a mechanism of signer ambiguity, as shown in the diagram, ensures privacy and is computationally formidable to penetrate, making it a valuable tool for maintaining the signer's anonymity within a set [72, 166]. The practicality of ring signatures is evident in various domains, most notably within blockchain technology and cryptocurrencies, where they play a pivotal role in safeguarding transactional anonymity [166, 115, 94]. In the

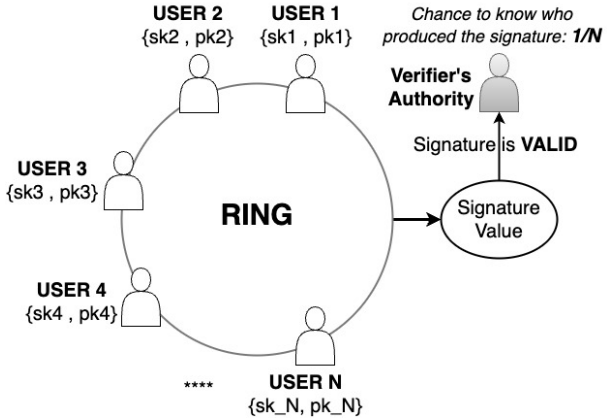

Figure 3.2: Ring Signature, a privacy preservation technique where the level of anonymity is restricted to the number of users (i.e., size of the ring).

realm of cryptocurrencies, ring signatures find a significant application in Monero [181], a prominent privacy-centric cryptocurrency. By leveraging the ring signature mechanism, Monero can obscure the actual sender of a transaction by blending their identity with decoy inputs, ensuring the origin of funds remains confidential. Technically, Monero combines *Ring Signatures*, *Stealth Addresses* [47], and *Confidential Transactions* [135] to achieve transactional privacy. Ring Signatures involves the creation of a signature by a group such that it becomes computationally infeasible to determine the actual signer. Stealth Addresses ensure transaction outputs are directed to one-time addresses, render-

ing them unlinkable to the recipient's public address. This prevents potential linkage attacks where transaction patterns are analysed to determine sender-receiver relations. These techniques combined make Monero transactions opaque, ensuring the details of the sender, receiver, and transaction amounts remain concealed [181].

However, while ring signatures offer enhanced privacy, it is imperative to understand their limitations and the trade-offs involved [42]. Specifically, the size of the ring (i.e., the number of potential signers) can impact both the level of anonymity provided and the computational overhead required. Larger rings offer higher anonymity but at the cost of increased computational resources and transaction sizes. Moreover, while ring signatures shield the sender's identity, they do not inherently conceal transaction amounts or the recipient's address. To achieve a more comprehensive privacy solution, ring signatures are often combined with other cryptographic techniques, such as *Confidential Transactions* [135] or *Stealth Addresses* [47] as pointed out in Monero. This combination will complement Ring Signatures as a robust cryptographic tool, balancing transactional privacy and computational efficiency [94].

3.4 Attribute-based encryption

ABE is an advanced cryptographic method where user attributes are defining and regulatory factors in the encryption process that provide the framework for the ciphertext produced by a user's secret vital [156, 22, 38]. A practical application can be seen in healthcare blockchain platforms, where ABE controls access to sensitive patient data [98]. Successful decryption of the encrypted data using a user's secret key occurs only if the user's attributes align with the ciphertext attributes [28]. This arrangement highlights the crucial security aspect of ABE known as collision resistance, ensuring that malicious

users cannot access data beyond what their private keys allow, even in instances of collusion [39, 117, 94]. ABE was first introduced by Sahai and Waters in 2005 [156], and it has since evolved into two main types: Key-Policy ABE (KP-ABE) and Ciphertext-Policy ABE (CP-ABE).

A unique feature of ABE is its capacity to support confidentiality and access control through singular encryption [82, 186, 211, 94]. This is facilitated by the involvement of four key parties, namely, the cluster head or data owners, blockchain miners, attribute authorities (AA), and the distributed ledger (typically represented by blockchain with blocks of transactions) [129, 85, 149]. Despite its potential, the application of ABE remains limited, primarily due to a general lack of comprehension surrounding its core principles and efficient implementation strategies [90, 198].

Furthermore, ABE does not necessitate a fixed authority, enabling the potential for multiple authorities within a decentralised network, providing a robust and adaptable approach for network management [38, 201]. Technologies such as the IPFS [19] and Storj [190] leverage this, permitting witnesses to assume the role of these authorities within a blockchain. Despite the advances, implementing ABE within a blockchain-centric approach presents ongoing challenges. It is a topic of active research [140, 94].

We now delve into the types of ABE namely Key-Policy Attribute-Based Encryption (KP-ABE) and Ciphertext-Policy Attribute-Based Encryption (CP-ABE)

3.4.1 Key-Policy Attribute-Based Encryption (KP-ABE)

KP-ABE is a specialised encryption scheme that enables fine-grained access control to encrypted data based on the attributes of users and predefined policies. This cryptographic technique, introduced by Goyal, Pandey, Sahai, and Waters in 2006, extends the basic principles of Attribute-Based Encryption (ABE) by enabling the decryption process to be governed by an access structure defined over attributes rather than by the ciphertext itself [82].

In KP-ABE, the data owner encrypts the data using a set of attributes, and the access policy is embedded within the user's private key during the key generation process [82]. This access policy specifies which combinations of attributes allow a user to decrypt the data. Users are issued private keys with embedded access policies, and they can decrypt the ciphertext only if the attributes associated with the ciphertext satisfy the access policy encoded in their private key [82, 10]. This model is particularly effective in environments where access control is based on user privileges or roles and is highly relevant for applications such as data sharing, secure communications, and digital rights management [10, 11].

- *Attributes:* Attributes are descriptive labels attached to the ciphertext by the data owner, representing characteristics such as roles, permissions, or other defining features [82].
- *Access Policy:* The access policy is a logical structure embedded within the user's private key. It defines the specific combination of attributes necessary for decryption the ciphertext. This policy can range from simple conjunctions of attributes (e.g., "Department: A&E AND Role: Nurse") to complex Boolean expressions [82].
- *Ciphertext:* The ciphertext is the encrypted data associated with a set of attributes but not directly containing the access policy.

- *Key Generation:* A trusted authority generates private keys for users based on their access policies. These keys allow users to decrypt data only if the attributes linked to the ciphertext meet the criteria of the access policy embedded in their keys [82].

3.4.2 Ciphertext-Policy Attribute-Based Encryption (CP-ABE)

CP-ABE is a specialised encryption scheme that allows fine-grained access control to encrypted data based on the attributes of users and predefined policies [22]. This cryptographic technique, introduced by Bethencourt, Sahai, and Waters in 2007, extends the basic principles of ABE by enabling the encryption process to be governed by an access structure defined over attributes rather than individual users [22, 157].

In CP-ABE, the data owner defines an access policy embedded within the ciphertext during the encryption process. This access policy specifies which attributes a user must possess to decrypt the data. Users are issued private keys associated with their attributes, and they can decrypt the ciphertext only if their attributes satisfy the access policy encoded in the ciphertext [22]. This model is particularly effective in environments where access control is based on user roles or characteristics rather than individual identities. It is highly relevant for applications such as cloud computing, secure data sharing, and healthcare information systems [22].

The main components of CP-ABE include the following:

- *Access Policy:* The access policy is a logical formula defined by the data owner, specifying the attributes required to decrypt the data. It can be a simple conjunction of attributes (e.g., ”Department: Cardiology AND Role: Surgeon”) or a more complex Boolean expression [22].

- *Attributes:* Attributes are descriptive labels assigned to users, representing their roles, permissions, or other characteristics. Users' private keys are generated based on these attributes [22].
- *Ciphertext:* The ciphertext is the encrypted data incorporating the access policy. Only users whose attributes match the policy can decrypt the ciphertext [22].
- *Key Generation:* A trusted authority generates private keys for users based on their attributes. These keys enable users to decrypt data if their attributes satisfy the access policy [22].

CP-ABE provides several advantages, such as flexible access control and reduced overhead in managing encryption keys, as it eliminates the need to manage individual user keys [156, 22, 82]. However, it also presents challenges, particularly regarding scalability and efficiency, as the complexity of the access policy increases. The encryption and decryption processes can become computationally intensive, and managing large sets of attributes may introduce performance bottlenecks.

CP-ABE is particularly useful in decentralised environments like blockchain, where it can be integrated with smart contracts to enforce fine-grained access control to data stored on the blockchain [218, 12]. For example, in healthcare systems, CP-ABE can ensure that only authorised personnel with the necessary attributes can access sensitive patient records [188].

While CP-ABE offers robust access control mechanisms, it faces challenges such as key management complexity , potential performance issues with large-scale attribute sets , risk of attribute disclosure.

3.5 Secure multi-party computation

SMPC has emerged as a cornerstone cryptographic protocol, enabling multiple entities to collaboratively compute a function over their respective inputs while maintaining the confidentiality of these inputs [56, 48]. Rooted in the seminal work by Yao in 1982 [202, 94], SMPC distributes data or programmatic states, such as those inherent to blockchain-based smart contracts, across N parties via advanced secret sharing techniques. Under this paradigm, a subset of M parties from the total N. N is indispensable for the joint computation of a designated input, producing the output and revealing the underlying data or programmatic states [94]. Notably, each participating entity is privy only to an input segment, thereby possessing a unique position on a distinct polynomial, instrumental for discerning a specific data segment [202, 94].

Integrating SMPC in blockchain frameworks epitomises an evolutionary step towards enhancing user privacy [212, 215, 20]. Bitcoin, for instance, has harnessed variations of multi-party computation for the augmentation of transactional privacy, a significant illustration being its deployment in the generation of threshold signatures [6]. Central to this privacy-centric methodology is the imperative for the majority of participants to maintain probity, thereby ensuring the sanctity and security of the collective computations [94]. Nonetheless, it is pertinent to acknowledge the computational intricacies introduced by SMPC, especially in terms of network latency attributed to the inherent inter-node data exchanges requisite for MPC computation [20].

In the decentralised computational landscape, the Enigma platform, unveiled in 2015, is a paragon of SMPC implementation [219]. Enigma harnesses a verifiable secret-sharing scheme, fortifying its computational model's integrity and privacy. Further, instead of engendering a nascent blockchain, Enigma employs an auxiliary blockchain as an immutable event ledger, concurrently facilitating peer-to-peer network governance, which addresses identity management and access control nuances [219].

3.6 Zero-Knowledge Proof

ZKP is an advanced cryptographic technique designed to enhance privacy in the blockchain [144]. A relevant use case of ZKP is in identity verification systems, allowing users to prove specific attributes, like age or citizenship, without revealing other personal details [167, 94]. The origins of this privacy-preserving technique trace back to the 1980s when it was proposed as a zero-knowledge proof [68, 79]. At its core, the concept provides formal proof that verifies a program's execution using an input secretly known by the user [78]. These proofs allow one entity to convincingly demonstrate to another that it possesses a particular piece of information without divulging any specifics about the data. The execution results in a publicly available output, ensuring no sensitive data or attributes are exposed [94].

For ZKP to function effectively, it must satisfy three critical criteria as follows [27]:

1. *Completeness*: Given a true statement, the prover should invariably be able to present a successful proof [27, 94].
2. *Soundness*: If the statement under scrutiny is false, a dishonest prover should find it near-impossible to deceive the verifier into believing its authenticity, barring a minuscule probability [27, 94].
3. *Zero-Knowledge*: There should be an algorithm, limited by polynomial time, that can autonomously create transcripts of the protocol which cannot be differentiated from a genuine proof shared between a prover and a verifier. This is pivotal because if an external entity, unaware of the statement's veracity, can fashion a legitimate protocol transcript, neither a verifier nor any interceptor can glean additional information from the real interaction [27, 94].

As depicted in Figure 3.3, the process involves two parties: a Prover and a Verifier. The Prover sends a confidential information, generates a proof of their knowledge using `func(makeProof)`, and sends this proof to the Verifier. The Verifier then uses `func (checkProof)` to verify the proof's validity and concludes the process by obtaining the results. This succinctly illustrates the ZKP methodology where the Prover can affirm the possession of confidential information without revealing any information itself, thus preserving privacy [94].

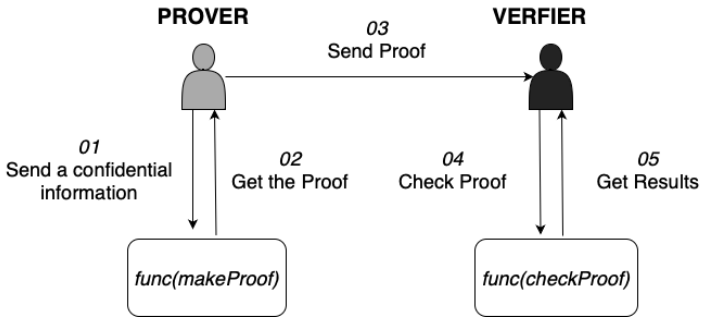

Figure 3.3: Zero Knowledge Proof, a privacy preservation technique where a formal proof performed by a Verifier helps verify program's execution from a Prover.

Types of Zero-Knowledge Proofs There are several variations and advancements of ZKP, each catering to different use cases and providing various levels of privacy and efficiency:

- *Interactive Zero-Knowledge Proofs*: In the original form, ZKPs were interactive, requiring multiple rounds of communication between the prover and the verifier. This interaction could make the protocol impractical for some applications, especially in scenarios where low latency or minimal communication is crucial.
- *Non-Interactive Zero-Knowledge Proofs (NIZK)*: Non-Interactive ZKPs eliminate the need for multiple rounds of communication between the prover and the verifier. Introduced by Blum, Feldman, and Micali in 1988 [193], NIZKs rely on a CRS that both parties have access to. Once the CRS is established, the prover can generate a proof that the verifier can validate without any further interaction. NIZKs are particularly useful in decentralised environments, such as blockchains, where interaction between parties may not be feasible or desirable.

- *zk-SNARKs (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge)*: zk-SNARKs are a type of NIZK that provides both succinctness and non-interactivity. They enable the generation of very short proofs that can be verified quickly, making them highly efficient. zk-SNARKs were popularised by the Zcash cryptocurrency, which uses them to provide privacy-preserving transactions. One key advantage of zk-SNARKs is that they allow for the verification of computational integrity without requiring the verifier to perform the computation themselves, preserving both privacy and efficiency [25].
- *zk-STARKs (Zero-Knowledge Scalable Transparent Arguments of Knowledge)*: zk-STARKs are an evolution of zk-SNARKs, designed to improve scalability and transparency. Unlike zk-SNARKs, zk-STARKs do not require a trusted setup phase, making them more secure and transparent. They are also more scalable and capable of handling larger computations with lower computational overhead. zk-STARKs are particularly appealing in scenarios where transparency and scalability are paramount, such as in large-scale blockchain applications [15].

Applications of Zero-Knowledge Proofs in Blockchain ZKPs have found numerous applications within the blockchain ecosystem, where privacy, security, and scalability are critical concerns [94]. Some notable use cases include:

- *Privacy-Preserving Transactions*: ZKP are integral to privacy-focused cryptocurrencies like Zcash, which employs zk-SNARKs to enable shielded transactions. In these transactions, the details of the sender, receiver, and transaction amount are concealed while still allowing the network to verify the validity of the transaction, ensuring both privacy and security, addressing the transparency issues inherent in public blockchains [88].

- *Scalable Verification*: ZKP enable scalable verification of complex computations without revealing sensitive data. For instance, zk-SNARKs can be used in smart contracts to verify the correctness of off-chain computations or data without exposing the underlying information. This can significantly reduce the computational load on the blockchain, improving scalability while maintaining privacy[112].
- *Identity Verification*: ZKP can be used for privacy-preserving identity verification, allowing users to prove certain attributes (such as age, citizenship, or credentials) without revealing their full identity. This is particularly useful in decentralised identity systems, where users control their own data and can selectively disclose information as needed[217].
- *Decentralised Voting*: ZKP can enhance the privacy and security of voting systems by allowing voters to prove that their vote is valid and counted without revealing their vote. This ensures the integrity of the voting process while protecting voter privacy, making it suitable for secure electronic voting systems[37].
- *Data Sharing and Access Control*: ZKP can be applied to create secure data-sharing mechanisms where parties can prove they meet certain conditions or have certain rights without revealing sensitive data. For example, in healthcare, ZKP can be utilised to prove they have a valid prescription without revealing their entire medical history[217].

3.7 Differential Privacy

Differential privacy, a paradigm conceived by Cynthia Dwork [58], has been acknowledged for pioneering in safeguarding individuals' privacy in computational processes [60]. At the heart of this framework is a mathematically rigorous guarantee: an adversary's knowledge about an individual remains invariant, whether or not the individual's data is part of the computational dataset. Given two adjacent databases, $D1$ and $D2$, that differ by precisely one record, the probabilities of obtaining any specific output from these databases should

be nearly identical [58, 61, 94]. Consequently, an adversary remains indecisive about the database of origin for a particular output, which implies that, even when equipped with supplementary data, the adversary needs to gain additional insights about the individual in question [94].

The concept is exemplified in Figure , which outlines the fundamental operation of differential privacy. The two databases, $D1$ and $D2$, differ by only one record. When each is subjected to an equivalent analysis or computation, the resulting answers, A for $D1$ and B for $D2$, should be nearly indistinguishable. This ensures that the participation of an individual's record in either database does not provide significant information that could lead to their identification.

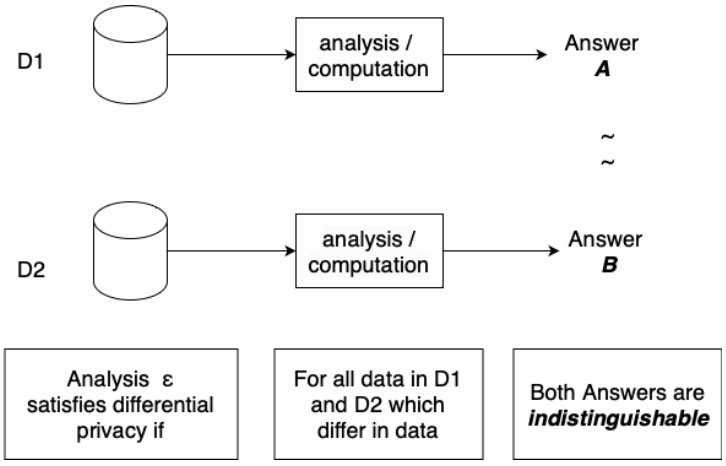

Figure 3.4: Differential Privacy, a privacy preservation technique that safeguards blockchain users' data published on-chain with nearly identical outputs when processed.

The efficiency of differential privacy is parameterised by the privacy constant, ε, and the cumulative queries executed over a duration [58, 61, 59]. The smaller the value of ε, the stronger the privacy guarantee; however, it often comes at a utility cost. Notable applications of differential privacy include its integration in systems like the U.S. Census Bureau's OnTheMap [1], Google's RAPPOR, Apple, and Microsoft [101], showcasing its practical usage so far. Moving from a theoretical approach to real-world implementation presents challenges, including requiring a skilled person and an appropriate computing development environment and determining the most favourable ε parameters [73].

In recent years, the blockchain landscape has identified the potential of incorporating differential privacy mechanisms [120, 86, 71, 94]. For instance, research has been conducted on using differential privacy and blockchain together with Bitcoin [109]. These studies [109] aim to find ways of preserving privacy in the overall structure of the blockchain and to evaluate the effectiveness of differential privacy in improving anonymity. Such integrations are paramount when adversaries exploit blockchain records to deduce sensitive user information, especially during federated learning processes or while chronicling crowd-sourced endeavours [86, 71, 97, 95].

A notable stride in this direction was the introduction of a blockchain-centric data-sharing framework [71]. This novel approach empowers data proprietors with the capability to oversee anonymisation procedures, thereby thwarting potential data mining-centric threats targeted at blocking information. Using local differential privacy (LDP) in conjunction with blockchain has also been an approach. LDP is a privacy-preserving technique that ensures individual privacy while allowing statistical data analysis [60]. By combining blockchain and LDP, a secure genomic data management system was built that addresses privacy concerns associated with sharing gene data using LDP [142, 94].

3.8 Homomorphic Encryption

HE, a significant advancement in cryptographic techniques, facilitates operations directly on encrypted data, obviating the need for decryption prior to computation [76, 53]. For example, in blockchain-based healthcare services, HE enables data analytics on patient records without exposing individual data [185]. There are primarily two forms of HE: Fully HE (FHE) and Partially HE (PHE). FHE allows for addition and multiplication operations on encrypted data, whereas PHE restricts operations to addition or multiplication [130]. This capability paves the way for more secure data processing and holds particular promise for blockchain applications [94].

These FHE and PHE encryption schemes have been progressively integrated into blockchain frameworks, fortifying data privacy even during computational phases. Such implementations become indispensable for sectors handling critical and sensitive data [94]. Within the blockchain domain, executing private smart contracts or undertaking computations on confidential data without compromising data integrity is invaluable [107]. Also, in blockchain-based healthcare services, where patient records demand utmost confidentiality, HE allows data analytics and research without exposing individual patient data [185, 160, 94].

Implementing HE in blockchains poses several challenges due to blockchain systems' specific needs and properties. The computational demands of operating on encrypted data, mainly using FHE, can hinder the blockchain's performance and scalability [164]. Designing effective HE schemes should balance robust security and practical computation with processes to ensure accurate decryption [94]. Integrating encryption into established blockchain architectures demands attention to compatibility and maintaining the decentralisation structure. Beyond technical aspects, there are regulatory and legal hurdles since privacy norms differ globally, emphasising adherence to global regulations. The complex nature of HE requires enhanced educational awareness and expertise among blockchain developers and researchers [160, 94].

3.9 Related Work

In decentralised data-sharing environments, blockchain's consensus mechanisms, such as PoW and PoS,ensure that all nodes agree on the state of the ledger, thus preventing double-spending and fraudulent activities [104, 89]. Smart contracts, introduced by Ethereum, extend blockchain's capabilities by enabling programmable, self-executing agreements that facilitate complex transactions and data exchanges without human inter-

vention [33]. Industries such as healthcare, finance, and supply chain management have leveraged blockchain to enhance data security, privacy, and interoperability [150, 36]. For instance, in supply chain management, blockchain provides end-to-end visibility and traceability of goods, reducing fraud and improving efficiency [178].

Despite blockchain's advantages, its inherent transparency poses significant privacy concerns, especially when handling sensitive data [44].All transactions are recorded on a public ledger, which can lead to the exposure of user identities and transaction details through blockchain analysis techniques[121]. In privacy-sensitive domains like healthcare, this transparency conflicts with regulations such as the HIPAA and the GDPR, which mandate strict controls over personal data disclosure[63, 4]. Moreover, blockchain's immutability complicates the "right to be forgotten" stipulated by GDPR, as data, once recorded, cannot be altered or deleted[148]. This necessitates the development of privacy-preserving mechanisms that can coexist with blockchain's immutable and transparent nature.

To address privacy concerns, off-chain storage solutions have been proposed, wherein sensitive data is stored outside the blockchain, and only references or hashes are kept on-chain [196]. IPFS is a distributed file system that facilitates decentralised storage and retrieval of data [18].By integrating IPFS with blockchain, systems can achieve a balance between data accessibility and privacy [187]. IPFS's decentralised structure enhances privacy by eliminating centralised points of failure common in traditional cloud models. In systems that combine blockchain and IPFS, sensitive data is stored off-chain in encrypted form, while only the data's hash is stored on-chain, preventing exposure of private information on public ledgers [100, 41, 96].

ABE is crucial in securing data sharing by enabling fine-grained access control. In ABE systems, users' secret keys and the ciphertext are tied to attributes such as roles or access levels rather than to specific identities [82]. This flexibility allows for more precise control over who can access the data.

Among ABE schemes, CP-ABE has garnered attention for embedding access policies directly within ciphertext. In CP-ABE, the data owner defines an access policy based on attributes, ensuring that only users with the requisite attributes can decrypt the data. This method is especially advantageous in cloud computing, healthcare, and finance sectors, where secure, policy-driven data sharing is critical [102, 70].In healthcare, CP-ABE enables secure sharing of electronic health records, allowing data to be accessed only by authorised personnel, thereby preserving patient confidentiality [102, 69]. The fine-grained access control that CP-ABE facilitates is crucial in such sensitive environments. Recent advancements in CP-ABE have introduced features like attribute revocation and policy updates, enhancing the system's flexibility in dynamic environments where access rights may change frequently [152, 132].

Despite its benefits, CP-ABE faces challenges related to key management, scalability, and attribute revocation [210].Revoking user access or updating attributes requires complex key updates or re-encryption of data, which can be computationally intensive and impractical in dynamic environments [91].

Recent advancements have introduced privacy-enhancing mechanisms that integrate ZKP with CP-ABE. [35]proposed a blockchain-based payable outsourced decryption scheme using responsive ZKPs to verify outsourced results without adding redundant information to ciphertexts. This approach ensures both verifiability and fairness while maintaining efficient decryption mechanisms. Another notable development is a ZK-CPABE framework

that integrates Fiat-Shamir transformation to enhance scalability and privacy in CP-ABE transactions. By demonstrating improved transaction throughput and secure off-chain computation, this Ethereum-based implementation addresses one of the critical challenges in blockchain-integrated CP-ABE systems [213].

A critical yet often overlooked issue in CP-ABE systems is the risk of attribute disclosure. During the decryption process, users may inadvertently reveal their attributes, which can be exploited for profiling or unauthorised data access [114]. This is particularly concerning when attributes encode sensitive information, such as health conditions, roles within an organisation, or personal identifiers.

Existing solutions primarily focus on hiding access policies rather than user attributes, which does not fully mitigate the risk of attribute exposure [62]. There is a pressing need for mechanisms that can protect attribute privacy while maintaining the functionality of CP-ABE systems.

ZKP are cryptographic protocols that allow one party to prove knowledge of a secret without revealing the secret itself [77].They have been employed in various applications to enhance privacy and security, such as authentication systems, anonymous transactions, and blockchain protocols [16].In blockchain systems,ZKPs enable users to prove the validity of transactions without revealing transaction details, as demonstrated in privacy-focused cryptocurrencies like Zcash [88]. zk-SNARKs provide efficient non-interactive proofs that are suitable for blockchain environments [143].

Integrating ZKPs with CP-ABE could potentially address the attribute disclosure problem by allowing users to prove possession of required attributes without revealing them [74, 213]. This combination would improve privacy while retaining the fine-grained access control capabilities of CP-ABE.

By generating zero-knowledge proofs during the decryption process, users can demonstrate that they possess the necessary attributes without revealing any specific attribute information [113].

For example, in a healthcare data-sharing scenario, a doctor could prove that they have the attributes required to access a patient's medical records (e.g., medical license, department affiliation) without disclosing these attributes to the system or other parties [168].This not only preserves the doctor's privacy but also strengthens the overall security of the system by preventing attribute-based attacks.

Chapter 4

A Novel Privacy-preserving ZK CP-ABE Data Sharing System

4.1 Proposed Solution

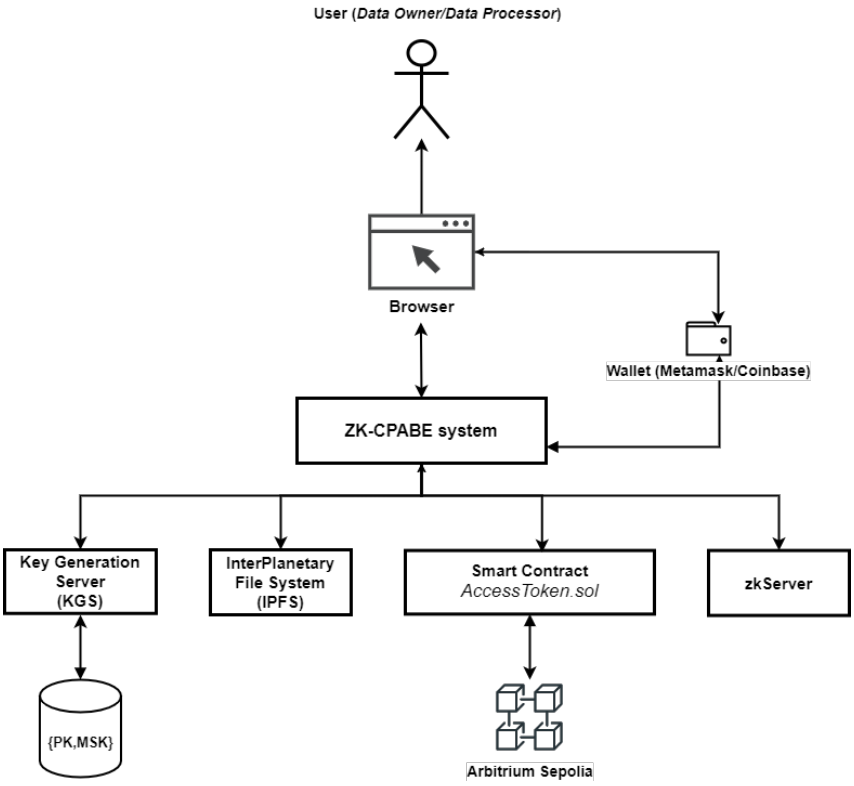

Figure 4.1: A high-level architecture diagram of the proposed privacy-preserving data sharing system based on Attributed-based Encryption and Zero Knowledge Proof.

Protecting sensitive personal data in today's digital world is essential, particularly as privacy concerns grow across domains like healthcare, finance, and government [92]. This project proposes a secure, privacy-preserving solution for accessing and managing personal data using a **Zero-Knowledge Ciphertext-Policy Attribute-Based Encryption (ZK CP-ABE)** system. The solution combines a decentralised **blockchain technology**(as detailed in Table 2.1), **ZKP**, and **ABE** to ensure that only authorised users can access specific data, while preserving the privacy of user attributes enabling verifiable, tamper-proof access control.

The solution consists of three key phases:

- **1. Initialisation:** The protocol starts with the **Key Generation Server (KGS)** performing the `SetUp()` operation, which generates a **Public Key (PK)** and a **Master Secret Key (MSK)**. The **Data Owner (DO)** receives the PK, enabling them to encrypt their data securely. This stage establishes the necessary cryptographic foundations for secure data sharing.

  Algorithm 1: Initialization

  Input: None

  Output: PK,MSK

  KeyGenerationServer.setup();

  return (PK,MSK);

- **2. Encryption and Token Creation:**

  The DO uses the PK to encrypt the data and defines an **access policy** that specifies the attributes required for access. The attributes are divided into two sets: **A_pass** (attributes that satisfy the access policy and allow decryption) and **A_fail** (attributes that fail to meet the policy and prevent decryption). The encrypted data, along with the access policy, is uploaded to a decentralised storage platform i.e., **IPFS**. The DO interacts with a **SC** on the blockchain which acts as an immutable ledger

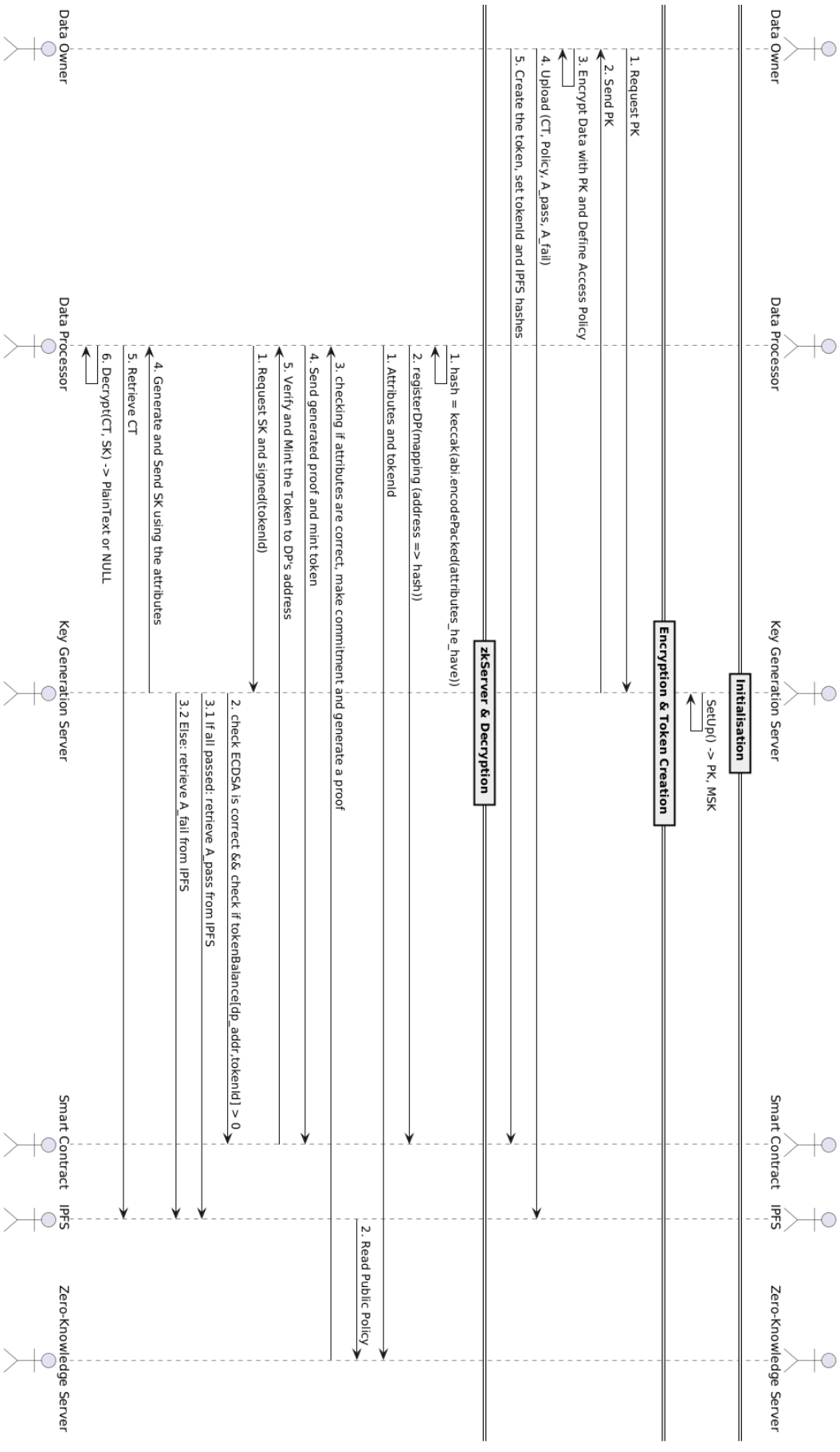

Figure 4.2: Proposed Flow and Phases of the ZK-CPABE system.

for access control management. The smart contract ensures that all access policies and permissions remain verifiable, tamper-proof, and decentralised. This creates a **token**, which contains a unique `tokenId` and the **IPFS CID** for the encrypted data.

Algorithm 2: Encryption & Token Creation

Input: $DataOwner, access\_policy$

Output: $CT, Policy, A_{pass}, A_{fail}, tokenId, IPFS\_hashes$

DataOwner requests PK;

KeyGenerationServer sends PK;

DataOwner encrypts data using PK and defines access policy;

DataOwner uploads $(CT, Policy, A_{pass}, A_{fail})$;

SmartContract creates token with $tokenId$ and stores IPFS hashes;

return $CT, Policy, A_{pass}, A_{fail}, tokenId, IPFS\_hashes$;

- **3. zkServer and Decryption:** To access the encrypted data, a **Data Processor (DP)** sends their attributes along with the `tokenId` to a **Zero-Knowledge Server (zkServer)**. The ZKS retrieves the public access policy from the Smart Contract (SC) and validates the DP's attributes using **zkServer** without revealing the actual attributes.If the DP's attributes meet the access policy, the SC mints an access token, allowing the DP to request the **SK** from the Key Generation Server (KGS). The KGS verifies the DP's `tokenId` and digital signature using the **ECDSA**, confirming their authenticity. If verified, the KGS retrieves the necessary **A__pass** or **A__fail** attributes from IPFS and issues the SK to the Data Processor (DP). The DP then uses the SK to retrieve and decrypt the ciphertext from IPFS. If the DP's attributes satisfy the access policy, the decryption succeeds, revealing the original plaintext. Otherwise, the decryption fails, returning a `NULL` value.

Algorithm 3: Zero-Knowledge Proof and Token Minting

Input: $DataProcessor, attributes$

Output: $proof, minted\_token$

$hash \leftarrow keccak(abi.encodePacked(attributes))$;

SmartContract.registerDP($DataProcessor.address, hash$);

DataProcessor retrieves attributes and $tokenId$;

ZeroKnowledgeServer verifies attributes;

if *attributes are valid* **then**

    ZeroKnowledgeServer generates proof;

    SmartContract mints token to $DataProcessor.address$;

return $proof, minted\_token$;

Algorithm 4: Token Verification and Secret Key Request

Input: $DataProcessor, tokenId$

Output: SK or $NULL$

DataProcessor requests SK and signed($tokenId$);

if *ECDSA signature is valid* ***and*** *token balance* > 0 **then**

    if *all checks pass* **then**

        DataProcessor retrieves A_{pass} from IPFS;

    else

        DataProcessor retrieves A_{fail} from IPFS;

KeyGenerationServer generates and sends SK based on attributes;

DataProcessor retrieves CT;

$PlainText \leftarrow DataProcessor.decrypt(CT, SK)$;

return $PlainText$ or $NULL$;

This system provides a decentralised, privacy-preserving, and secure method for data sharing where user attributes remain confidential and only authorised parties can access sensitive information.

4.2 Technical Stack

4.2.1 Client and Key Generation Server (KGS)

The Client and KGS are implemented in the same codebase using modern web technologies. The backend handles cryptographic operations and decentralised data management, while the frontend facilitates user interactions. The following sections outline the technology stack and key components of the system.

4.2.2 CP-ABE Backend (WASM/Rust)

The backend of the system was developed using **Rust**, a high-performance systems programming language which is best known for its memory safety and concurrency capabilities. Rust was chosen for its ability to generate WebAssembly WASM modules which allows cryptographic operations to run efficiently in a browser environment. The integration of Rust and WASM ensures that the system can handle complex cryptographic tasks securely and with high performance.

- **Language:** Rust is used as the core language for cryptographic operations due to its safety features, performance, and ability to compile to WebAssembly, enabling secure browser-based execution of cryptographic algorithms.
- **Primary Libraries:** RABE, A Rust library for Attribute-Based Encryption (ABE) was used for the implementation of the Bethencourt-Sahai-Waters (BSW) scheme used in this project [22]. RABE allows the system to implement policy-based encryption, where access is controlled by user attributes [2]. In CP-ABE, the access control policy is embedded within the ciphertext, and user attributes are encoded in

the private keys. Decryption is possible only if the user's attributes satisfy the access structure defined in the ciphertext. The `rabe` library implements the Bethencourt-Sahai-Waters (BSW) scheme [22], enabling policy-based encryption and decryption in decentralised environments.

The BSW CP-ABE scheme [22] is based on bilinear pairing and the Decisional Bilinear Diffie-Hellman (DBDH) assumption [75].

Setup: The trusted authority (TA) generates a public key and master key as follows [75]:

- Let G_1 and G_2 be two multiplicative cyclic groups of prime order p, and let $e : G_1 \times G_1 \to G_2$ be a bilinear map.
- The authority selects random generators $g \in G_1$ and random elements $\alpha, \beta \in \mathbb{Z}_p$.
- The public key PK and master key MK are defined as:

$$PK = \left(g, g^{\beta}, e(g,g)^{\alpha}\right), \quad MK = \alpha$$

Encryption: . To encrypt a message $M \in G_2$ under an access policy A, the data owner [75]:

- Selects a random value $s \in \mathbb{Z}_p$ and computes:

$$CT = \left(A, C = M \cdot e(g,g)^{\alpha s}, C_1 = g^s, \{C_x = g^{q_x(0)}, C'_x = T_x^{q_x(0)} \text{ for each } x \in A\}\right)$$

  where $q_x(0)$ is a polynomial for each attribute x, and T_x corresponds to the public key element for x.

Key Generation: The trusted authority uses the master key MK to generate a private key SK for a user with attributes S. For each attribute $x \in S$ [75]:

$$SK_x = \left(D_x = g^{\alpha + r_x}, D'_x = g^{r_x}\right)$$

where $r_x \in \mathbb{Z}_p$ is randomly chosen.

Decryption: A user with attributes S can decrypt the ciphertext if their attributes satisfy the access policy A. The decryption reconstructs $e(g,g)^{\alpha s}$ as follows[75]:

$$M = \frac{C}{e(C_1, D_x)/e(C'_x, D'_x)} = M$$

This works because the bilinear map cancels out the exponents and reconstructs $e(g,g)^{\alpha s}$, allowing the user to recover the message M.

The `rabe` library is ideal for this project as it supports the implementation of CP-ABE in decentralised environments [2].

A library that facilitates communication between Rust and TypeScript by generating WASM bindings [153]. It enables cryptographic functions written in Rust to be exposed to and called from TypeScript [127], allowing secure encryption and decryption operations to be performed directly in the browser. Implementing CP-ABE algorithms within a browser or WebAssembly runtime environment was a critical aspect for the proposed. To achieve this, a custom `rabe-wasm` wrapper was built, which imported the existing `rabe` library as a GitHub submodule [2]. The relevant algorithms were wrapped in functions capable of interacting with web-based objects, such as JSON. These wrapped algorithms were then compiled into WebAssembly using a Rust-to-WASM compiler, creating a custom CP-ABE library for the prototype. The compiler used for this process was `wasm-pack` [154].

```
1 #[wasm_bindgen]
2 pub fn encrypt(pk_json: &str, policy: &str, plaintext: &[u8
  ]) -> Result<String, JsValue> {
3     let pk: CpAbePublicKey =
4         serde_json::from_str(pk_json).map_err(|e| JsValue::
            from_str(&e.to_string()))?;
5     let ct = bsw::encrypt(&pk, policy, PolicyLanguage::
        HumanPolicy, plaintext)
6         .map_err(|e| JsValue::from_str(&e.to_string()))?;
7     let ct_json = serde_json::to_string(&ct).map_err(|e|
        JsValue::from_str(&e.to_string()))?;
```

```
    Ok(ct_json)
}
```

4.2.3 Frontend (React/Next.js)

The frontend of the application is built using **React** [125] and **Next.js** providing a modern, scalable, and responsive user interface [183]. React's component-based architecture is combined with Next.js's ability to enable server-side rendering and file-based routing, making the development process more streamlined and efficient. Additionally, **TypeScript** adds static type checking, improving code quality and reducing bugs[127].

- **Framework:** React is the core framework used to build the user interface [125]. Next.js extends React by offering server-side rendering, static site generation, and routing based on the file structure [183].
- **File-Based Routing:** Next.js uses a file-based routing system i.e., directory structure in the `app` directory defines the URL paths. For example, the file `src/app/mint/page.tsx` corresponds to the route `/mint` on the website, allowing for simple and efficient routing without the need for a separate router configuration [183].
- **Language:** TypeScript is used for the frontend code, providing static type checking and ensuring robust and error-free code [127].
- **Primary Libraries:**
  - `Pinata SDK`: Used to manage uploads to IPFS through the Pinata gateway. The system integrates with Pinata for easier file management and decentralised storage [146].
  - `supabase`: Supabase serves as the backend-as-a-service (BaaS) platform, providing a PostgreSQL database and an authentication system. It handles user data, access controls, and stores references to IPFS CIDs [170].

- `ethers.js`: A JavaScript library that allows the frontend to interact with the Ethereum blockchain, used for interacting with smart contracts, managing wallet connections, and executing transactions such as minting tokens and validating user attributes.
- `wagmi`: Wagmi is a library used to handle wallet connections, blockchain interactions, and account management [45]. In this project, Wagmi is configured to use the `Arbitrum Sepolia` network and supports the `CoinbaseWallet` for user connections. Wagmi simplifies the interaction with the Ethereum blockchain, using Alchemy as the transport layer to manage API calls and blockchain queries [45].
- `MetaMask`: MetaMask is integrated into the frontend for wallet authentication and blockchain interactions [174]. The system interacts with the MetaMask wallet through the `window.ethereum` object, which allows users to sign messages and perform transactions on the blockchain.
- `@mui/material`: MUI is used for UI components like buttons, text fields, and notifications to ensure a modern and responsive user interface [175].

4.2.4 Blockchain (Smart Contracts)

The blockchain component is implemented using **Solidity** [177], with the key smart contract being the *AccessToken contract*.This contract manages access control to encrypted data and provides cryptographic verification using the RISC Zero verifier [207]. The contract is responsible for managing the creation of tokens that represent access to encrypted data stored on IPFS [146].

- **Blockchain Network:** Ethereum (or any Ethereum-compatible network) is used as the platform for deploying the smart contracts [33]. It provides the decentralised infrastructure needed to ensure secure and auditable access control.

- **Smart Contract Language:** Solidity is used to implement the smart contracts. Solidity is designed specifically for the EVM [34], allowing the implementation of tokens, access control mechanisms, and policy management [177].
- **Primary Libraries:**
  - `@openzeppelin/contracts`: The contract uses OpenZeppelin libraries, which provide reusable and secure smart contract components [137]. This includes implementations for ERC1155 tokens (for semi-fungible tokens) and Ownable contracts to ensure best practices in access control and security [137].

AccessToken.sol Contract The AccessToken contract handles the creation, minting, and access control for tokens representing access to encrypted data. It integrates with the RISC Zero Verifier to validate ZKPs, ensuring that only authorised users can access the encrypted data.

```
contract AccessToken is ERC1155 {
    IRiscZeroVerifier public immutable verifier;
    bytes32 public constant imageId = ImageID.CHECK_POLICY_ID;

    mapping (uint256 tokenId => string cid) public tokenIpfsHash;
    mapping (uint256 tokenId => address owner) public tokenOwner;

    event TokenCreated(uint256 tokenId, address owner);

    constructor(IRiscZeroVerifier _verifier) ERC1155("") {
        verifier = _verifier;
    }

    function createToken(string memory cid) public {
        uint256 tokenId = uint256(keccak256(abi.encodePacked(msg.sender,
            block.timestamp)));
        tokenOwner[tokenId] = msg.sender;
        tokenIpfsHash[tokenId] = cid;
        emit TokenCreated(tokenId, msg.sender);
    }
}
```

Listing 4.1: AccessToken Solidity Contract Snippet

- **Token Creation:** Data owners can create tokens representing access to specific data stored on IPFS. Each token is associated with an IPFS CID, which links to the encrypted data. The function *createToken* generates a new token ID based on the data owner's address and an incrementing counter. The token is linked to the CID and assigned to the data owner.
- **Access Control:**The contract implements attribute-based access control. DPs must register their attributes using the *registerDP*function, which stores a hashed version of their attributes. A DP can request access by providing a proof generated and their respective attributes. The contract verifies the proof using the RISC Zero Verifier. Upon validity of proof, an access token is minted for the DP.
- **RISC Zero Verifier Integration:**The contract uses the RISC Zero Verifier to validate that the ZKP submitted by a data processor matches the access policy. The verifier ensures that only pre-defined guest programs, represented by a unique image ID can generate valid proofs. The verifier checks that the token ID is included in the ZKP and that the data processor's attributes match the policy defined in the encryption process.
- **Token Minting:**Once the ZKP is verified, the contract mints an access token for the data processor, allowing them to retrieve the encrypted data from IPFS. The function *mintAccessTokenForDP* handles this process.
- **IPFS Integration:**The contract stores the CID of each token, allowing users to retrieve the encrypted data from IPFS. The function *getCid* allows retrieval of the CID for a given token ID.
- **ECDSA Signature Validation:**The contract includes ECDSA-based signature validation to ensure that only the rightful owner of a token can perform specific actions, such as checking their balance or retrieving token information.

Usage of ERC1155 tokens allows for flexible, semi-fungible tokens, where each token represents access to a unique piece of encrypted data on IPFS.

4.2.4.1 Decentralised Storage (IPFS/Pinata)

IPFS (InterPlanetary File System) is the decentralised storage solution used in this system. IPFS enables data to be stored in a distributed manner across multiple nodes, ensuring availability, resilience, and security of the data. The system utilises **Pinata** [146] as a gateway to simplify interaction with IPFS.

- **Platform:** IPFS provides a decentralised, peer-to-peer file storage network. When files are uploaded to IPFS, they are given a unique **Content Identifier (CID)** that can be used to retrieve the file from any IPFS node.
- **Interaction:** Pinata is used as the gateway for managing uploads to IPFS. Data including the encrypted ciphertext and associated policies, are uploaded through Pinata's API, which returns a CID. This CID is stored within the smart contract to manage access to the encrypted data. By using Pinata, the system benefits from both IPFS's decentralised storage and Pinata's ease of file management.
- **Library:** The `PinataSDK` is used in the frontend to interact with the Pinata API. The SDK handles authentication via a JWT and provides methods for uploading data in JSON format to IPFS.

4.2.4.2 Database and Authentication(Supabase)

Supabase serves as the database and authentication layer of the system, managing user data and access control metadata. It provides a structured way to store user credentials, encrypted data references and mappings between users and their access rights [170].

- **Platform:** Supabase is an open-source BaaS platform helps in providing real-time databases, authentication services, and API integrations. The platform leverages a PostgreSQL database to manage structured data and offers built-in authentication mechanisms.

- **Usage:**
  - **Database:** Supabase provides a PostgreSQL database that stores public key and master secret key generated from KGS.
  - **Authentication:** Supabase's built-in authentication system is used to manage user access and identity verification. It supports multiple authentication methods, including OAuth and email/password, ensuring secure access to the application and its features.

4.2.5 Development Tools

A variety of development tools are used to streamline the building, testing, and deployment of the system's frontend and backend components.Create React App was used to bootstrap the frontend application [125]. Webpack configurations were customised to support the integration of WASM modules compiled from Rust. Wasm-Pack is used to compile Rust code into WASM and generate the corresponding JavaScript or TypeScript bindings [154]. Wasm-Pack ensures that the Rust-based cryptographic functions can be seamlessly integrated into the React frontend. The development environment was further supported by Visual Studio C++ as the integrated development environment (IDE) for developing the proposed system.

4.2.6 ZKP server(zkServer) Technical Stack

The **zkServer** [204] backend is built using a zkVM [207] that enables cryptographic proof generation in a decentralised manner. The backend leverages advanced cryptography, blockchain integration and decentralised storage to perform CP-ABE operations, generate ZKPs, and enforce access control through smart contracts.

4.2.6.1 Language and Framework

- **Rust:** The primary programming language used to build the zkServer. Rust's strong memory safety and concurrency support ensure that cryptographic operations, especially ZKP generation, run securely and efficiently. Rust also compiles to WASM, allowing for high-performance execution in browser-based environments if needed.
- **WebAssembly (WASM):** Rust code is compiled into WASM to execute in environments that require efficient cryptographic proof generation and also ensuring cross-environment compatibility and performance.
- **Docker:** The backend environment is containerised using Docker for zkServer can be deployed consistently across different environments. The Docker setup includes system dependencies like OpenSSL and the **Foundry** framework for smart contract development.

4.2.7 zkServer

4.2.7.1 RISC Zero

Developed by RISC Zero, Inc., RISC Zero is an advanced zero-knowledge proof technology that enhances privacy and security in software applications. Its unique zkVM allows developers to generate proofs verifying the correct execution of code without exposing underlying data. Compatible with languages like Rust and C++, RISC Zero enables the development of privacy-focused applications without requiring deep cryptographic expertise, broadening accessibility and scalability. This technology is particularly beneficial for decentralised systems, blockchain, and privacy-preserving data-sharing applications, where security and verifiable computation are critical. The following are key tech points and terminologies associated with RISC Zero technology:

- **RISC Zero ZKVM:** The **RISC Zero ZKVM** [207] provides a platform for generating and verifying zero-knowledge proofs. The RISC Zero ZKVM executes a piece of guest code and generates a proof (receipt) that consists of a **seal** [206]and a **journal** [206]. The **seal** confirms that the code was executed correctly and securely, providing a form of "digital signature" that certifies authenticity without revealing underlying data. The **journal** records non-sensitive information about the code execution, allowing verifiers to understand the code's outcome without accessing any confidential inputs. The receipt proves that the execution of the guest code was correct, without revealing any sensitive data used during execution.
- **ImageId:** The 'ImageId' is a unique identifier generated during the build phase of zkServer. It uniquely represents the guest code that runs inside the RISC Zero ZKVM. Only proofs generated by zkServers using the correct guest code (represented by the ImageId) are valid. This ensures that no one can generate fake proofs using incorrect guest code [205].
  - Represents Specific Guest Code: Each ImageId corresponds uniquely to the guest code within the RISC Zero ZKVM. This ensures that any generated proof is directly tied to a specific piece of code, enhancing code verification security.
  - Prevents Unauthorised Proof Generation:Only proofs generated by zkServers that match the designated ImageId are considered valid. This prevents fake or unauthorised proofs by ensuring that the code associated with the ImageId has not been altered or tampered with during execution [205].
  - Enables Reproducibility and Consistency:ImageId enables the consistent validation of code proofs across different instances, ensuring that any verifier can reliably confirm the validity of a proof when the ImageId aligns with the expected guest code.

4.2.7.2 Proof Generation Workflow

The zkServer interacts with various components to generate zero-knowledge proofs:

1. **Attribute Retrieval:** The DP submits their attributes and the `token_id`. The `token_id` is generated using the `keccak256` hash of the data owner's address and a counter to ensure uniqueness.
2. **Policy Check:** The zkServer retrieves the access policy from **IPFS** using the CID associated with the encrypted data. The policy is then evaluated against the submitted attributes to check if they satisfy the required conditions.
3. **Proof Generation:** If the attributes satisfy the policy, zkServer generates a proof using RISC Zero ZKVM. The proof contains two parts:
   - **Seal**: A cryptographic commitment to the guest program's execution.
   - **Journal**: A log of the program's execution, containing details such as the `token_id`.

   The proof (receipt) is then sent to the smart contract for verification.

4.2.7.3 Bonsai Integration

- **Bonsai** is a service that provides a bridge between zero-knowledge proof (ZKP) systems and decentralised networks like IPFS and Ethereum. Bonsai is used to manage the flow of data between the zkServer, IPFS for storage, and Ethereum for smart contract interactions [176]. It plays a key role in securely generating, validating, and managing zero-knowledge proofs within the decentralised infrastructure.
- In the context of the zkServer, Bonsai ensures that the zero-knowledge proofs generated by the server, leveraging the RISC Zero proving system, are securely processed and validated. It helps streamline the process by:
  - Fetching access policies stored on IPFS and verifying whether the data processor's attributes satisfy the conditions specified by the policy.

  - Handling proof validation using RISC Zero's remote proving capabilities and securely storing the proofs, ensuring that only valid proofs are accepted by the smart contract.
  - Providing a reliable mechanism to integrate ZKP with the blockchain and decentralised storage in a scalable and efficient manner, facilitating seamless interaction between the zkServer, IPFS, and Ethereum.
- Bonsai's role in the zkServer is to orchestrate the interaction between the cryptographic proof generation (leveraging RISC Zero's remote proving system) and the decentralised storage i.e., IPFS and smart contracts, thereby simplifying the entire process of policy verification and proof management in a decentralised, privacy-preserving system [176].

4.2.7.4 Foundry for Smart Contract Integration

- **Foundry** is used for developing and testing the smart contracts associated with the zkServer. The Foundry framework allows developers to build, test, and deploy Solidity contracts seamlessly, and integrates with Rust for ZKP verification [204].
- Foundry's integration in the Docker environment includes installing the Foundry toolchain (`forge` and `cast`) and setting up the necessary environment to manage Ethereum smart contracts.
- **Anvil**, a key component of Foundry, is utilised in the zkServer project for local blockchain development and contract deployment. Anvil provides a local Ethereum node that simulates blockchain environments, allowing for rapid testing and interaction with smart contracts [173].

Chapter 5

Discussion

5.1 System Implementation and Analysis

In this section, we delve into the technical implementation of the proposed privacy-preserving data-sharing system.

5.1.1 KGS and Encryption

The Client and KGS [22] were implemented on the same codebase and can be referred to as a single unified web application. This subsection will outline the technologies used to implement the web application and provide an overview of the core features.

In the first step of this system, the PK and MSK for the data owner are generated using the RABE BSW CP-ABE scheme [22]. CP-ABE is a form of attribute-based encryption where the data owner defines a policy that dictates which users can decrypt the data [82, 22]. The policy is embedded directly into the ciphertext, and users are assigned keys based on their attributes. The BSW variant of CP-ABE is well-known for its flexible policy

structure, allowing the use of expressive Boolean logic to control access [22]. To store the keys securely, Supabase is utilised as a secure storage solution for the generated keys [170]. It is an open-source alternative to Firebase, providing real-time databases, authentication, and storage services [170]. The client interacts with Supabase using environment variables such as:

```
NEXT_PUBLIC_SUPABASE_URL=https://yourlsupbaseurl.supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=your_supabase_key
```

These credentials enable the system to store and retrieve the public and master secret keys for the data owner which inturn ensures seamless encryption and decryption processes. Once the PK and MSK are generated, they are then stored in Supabase [170]. The PK is publicly retrievable, while the MSK is kept confidential and is not shared with users.

The PK and MSK are generated using RABE's `setup()` function. The PK is distributed to anyone who needs to encrypt data under a defined policy, while the MSK is used by the data owner to generate secret keys for users. This key generation process includes the following steps:

- **Setup**: The data owner runs the `setup()` function to create the PK and MSK.
- **Storage**: Both keys are stored in Supabase, where the PK can be retrieved for encryption, while the MSK is kept private.

The CP-ABE encryption process takes a plaintext message and a policy as inputs. The policy is evaluated against the attributes of potential decryption users, and only users with attributes satisfying the policy will be able to decrypt the resulting ciphertext.

The encryption process works as follows:

1. **Input**: The data owner inputs a plaintext message and defines a policy.

2. **Encryption**: The RABE encryption function takes the plaintext and the policy, generating a ciphertext.The policy is embedded in the ciphertext, ensuring that only authorised users can decrypt it.
3. **Output**: The resulting ciphertext is uploaded to IPFS using Pinata, a gateway service for IPFS that facilitates easier file management and retrieval in the decentralised storage system. The system returns a content identifier (CID), which uniquely identifies the file on IPFS.

In the CP-ABE BSW scheme [22], the policy for encryption defines the attributes required for decryption. These policies are written using Boolean logic with operators such as `AND`, `OR`, and `NOT`. A typical policy might look like this [2]:

```
("Role:Doctor" AND "Department:Oncology") OR
("Role:Doctor" AND "Department:Cardiology") OR
("Role:Doctor" AND "Department:XXX")
```

This policy allows decryption by users who meet any one of the following criteria: 1. A role of ”Doctor” in the ”Oncology” department, 2. A role of ”Doctor” in the ”Cardiology” department, or 3. A role of ”Doctor” in a department identified as ”XXX”.

In the BSW CP-ABE scheme, the policy used for encryption defines the set of attributes required for decryption. These policies are expressed in Boolean logic using operators such as `AND`, `OR`, and `NOT`.

- **Conjunctive (AND) Policy**: A conjunctive policy requires **all** specified attributes to be present for decryption.

  Example:

  ```
  "Role:Doctor" AND "Department:Oncology"
  ```

  Only users who are doctors in the oncology department can decrypt the data.

- **Disjunctive (OR) Policy**: A disjunctive policy allows decryption if **at least one** of the specified attributes is satisfied.

  Example:

  ```
  "Department:Oncology" OR "Department:Cardiology"
  ```

  Users from either the oncology or cardiology departments can decrypt the data.

- **Mixed Policy (AND/OR Combination)**: Mixed policies use both `AND` and `OR` operators to form more complex conditions for decryption.

  Example:

  ```
  "Role:Doctor" AND ("Department:Oncology" OR "Department:Cardiology")
  ```

  This allows any doctor from either the oncology or cardiology department to decrypt the data.

- **Negative (NOT) Policy**: A negative policy excludes certain attributes, denying access to users with specific characteristics.

  Example:

  ```
  NOT "Department:XXX"
  ```

  Users associated with the ”XXX” department are restricted from decrypting the data.

- **Threshold Policy**: A threshold policy allows decryption if **at least a minimum number** of attributes from the policy are satisfied. It is often represented as a (k,n) policy, where k is the minimum number of attributes required, and n is the total number of attributes.

  Example:

  ```
  (2, 3) threshold:
  "Role:Doctor",
  "Department:Oncology",
  "Clearance:Level3"
  ```

  A user can decrypt the data if they meet **any 2 out of the 3** conditions. For example, a doctor in the oncology department, or someone in the oncology department with level 3 clearance.

The encryption time is measured to analyze how policy complexity affects the system's performance. As policies become more complex (with more attributes or Boolean operators), encryption time increases linearly as seen in the performance conducted, which is an important metric for optimizing system efficiency.

After the ciphertext is generated, it is uploaded to IPFS for decentralised storage [18]. IPFS is a peer-to-peer file storage protocol designed to make data distribution more efficient. Files uploaded to IPFS are given a unique CID, which allows them to be retrieved from any node in the network, ensuring accessibility and immutability [18].

Pinata is used as a gateway service to handle the uploading and pinning of files to IPFS [146]. Pinning ensures that the uploaded file is persistently stored and remains accessible across the IPFS network. Once uploaded, the CID is returned, which represents the location of the ciphertext on the IPFS network. The CID returned by Pinata uniquely identifies the location of the encrypted file on IPFS.

After the ciphertext is uploaded to IPFS, a token representing access to the encrypted data is created using a Solidity smart contract [177] following the ERC-1155 standard [137]. The token creation process involves the following steps:

- **Token ID Generation**: A unique token ID is created by combining the data owner's address and an incrementing counter. This ensures each token is unique to the owner.
- **Token and CID Association**: The token is linked to the CID generated during the IPFS upload, connecting the token to the encrypted data stored on IPFS.
- **Token Minting**: The `createToken()` function in the smart contract mints the token and assigns it to the data owner. Below is the relevant code snippet:

```
function createToken(string memory cid) public {
    _tokenIdCounter++;
    uint256 tokenId = uint256(keccak256(abi.encodePacked(msg.sender,
        _tokenIdCounter)));
    require(tokenOwner[tokenId] == address(0), "This Token Id Has Been
        Created.");
    tokenOwner[tokenId] = msg.sender;
    _ownerTokens[msg.sender].push(tokenId);
    setIpfsHash(tokenId, cid);
    emit TokenCreated(tokenId, msg.sender);
}
```

This process ensures that only the data owner, who holds the token, can access the encrypted data on IPFS. The step involves generating public and master keys using the RABE BSW CP-ABE scheme [22], securely storing these keys in Supabase, encrypting data based on a specified policy, and uploading the encrypted data to IPFS using Pinata [146]. Access control is managed by minting tokens using a Solidity smart contract [177] , AccessToken.sol, where the data owners' tokens are associated with the IPFS CID.

Next , The DP must first register their attributes with the SC before they can access any encrypted data. The steps involved are:

1. **Hashing Attributes**: The DP generates a cryptographic hash of their attributes using the `keccak` hashing algorithm [162]. This ensures that the DP's attributes remain private while allowing the SC to later verify them without revealing the actual attribute values.

```
    mapping (address dpAddr => bytes32 attributesHash) internal
        dpAttrHash;
```

2. **Registering with the Smart Contract**: The hashed attributes are then mapped to the DP's address using the SC's `registerDP()` function. This ensures that the SC can verify the DP's attributes at a later stage.

```
function registerDP(address dpAddr, bytes32 attributesHash) public
    {
    dpAttrHash[dpAddr] = attributesHash;
    emit DPRegistered(dpAddr, attributesHash);
}
```

This registration allows the DP to mint access tokens based on their attributes in subsequent steps.

5.1.2 Minting the Access Token Using RISC Zero ZKVM

The minting of the access token involves several interactions between the DP, the zkServer, IPFS, and the SC. The zkServer in this system is based on the `risc0-foundry-template` [204], utilizing the **RISC Zero ZKVM** [207]. The zkServer ensures privacy-preserving verification of the DP's attributes without revealing them to external parties. Here's the process:

1. **Sending Attributes and Token ID to zkServer**: The DP sends their attributes and Token ID to the zkServer. The Token ID was generated by the SC during the token creation process. The zkServer ensures the correctness of these attributes using the `ImageID`, a unique identifier generated during the building phase of the zkServer. This `ImageId` guarantees that proofs generated by other zkServer's with incorrect code will not pass verification [205].

```
pub fn generate_proof(
    policy_str: &str,
    dp_attr_str: &str,
    token_id: U256,
) -> Result<(Vec<u8>, U256)> {
    let token_id_bytes = token_id.abi_encode();
```

```
        let env = ExecutorEnv::builder()
            .write(&policy_str)?
            .write(&dp_attr_str)?
            .write_slice(&token_id_bytes)
            .build()?;
        // Generate the proof using zkServer
        let prover = default_prover();
        let receipt = prover.prove_with_ctx(env, &
           VerifierContext::default(), CHECK_POLICY_ELF, &
           ProverOpts::groth16())?.receipt;
        let seal = groth16::encode(receipt.inner.groth16()
           ?.seal.clone())?;
        Ok((seal, recovered_token_id))
    }
```

2. **Retrieving Public Policy from IPFS**: The zkServer reads the public policy from IPFS, which outlines the attributes required to access the encrypted data. The zkServer checks if the DP's attributes satisfy the policy. If the attributes match, a proof is generated and returned to the DP.

```
    pub async fn read_policy_from_ipfs(cid: String) ->
       Result<String> {
        let ipfs_url = format!("{}{}", env::var("
           IPFS_GATEWAY").unwrap(), cid);
        let res = reqwest::get(ipfs_url).await.unwrap().
           text().await.unwrap();
        let policy: serde_json::Value = serde_json::
           from_str(&res).unwrap();
        let policy_str = policy["policy"].as_str().unwrap()
           ;
        Ok(policy_str.to_string())
    }
```

3. **Generating Zero-Knowledge Proof**: If the attributes match the policy retrieved from IPFS, the zkServer generates a zero-knowledge proof, which contains both a `seal` and a `journal`.The seal is a cryptographic commitment that ensures the computation (matching the attributes with the policy) was performed correctly, without revealing any sensitive attribute information [206]. It guarantees the integrity of the computation and confirms that the zkServer processed the data as expected, while keeping the actual attribute values private.

   The journal, on the other hand, contains the public information derived from the computation that can be safely shared with the DP. It includes non-sensitive results of the verification, such as metadata or confirmation that the policy was satisfied, enabling the DP to mint the access token without needing to see or handle the private attributes [206].

   This proof which contains both the seal and journal, is then sent to the DP. The proof ensures that the DP can mint the access token while keeping sensitive attribute information hidden, maintaining privacy through zero-knowledge proofs.

4. **Submitting Proof to Smart Contract**: The DP submits the zero-knowledge proof and the Token ID to the SC. The SC verifies the proof by comparing the hashed attributes stored during the registration process with the provided proof.

```
1    function mintAccessTokenForDP(
2        bytes calldata seal,
3        uint256 tokenId,
4        bytes32 attributesHash
5    ) public {
6        require(dpAttrHash[msg.sender] != bytes32(0), "DP has not been
             registered");
7        require(dpAttrHash[msg.sender] == attributesHash, "Invalid
            attributes hash");
8        verifier.verify(seal, imageId, sha256(abi.encodePacked(tokenId
            )));
9        _mint(msg.sender, tokenId, 1, "");
10       emit AccessTokenMinted(msg.sender, tokenId);}
```

5. **Minting the Token**: Once the proof is successfully verified, the SC mints the access token and assigns it to the DP's address. The token represents DP's ability to access the encrypted data stored on IPFS.

```
function createToken(string memory cid) public {
    _tokenIdCounter++;
    uint256 tokenId = uint256(keccak256(abi.encodePacked(msg.
        sender, _tokenIdCounter)));
    require(tokenOwner[tokenId] == address(0), "Token ID already
        exists");
    tokenOwner[tokenId] = msg.sender;
    _ownerTokens[msg.sender].push(tokenId);
    setIpfsHash(tokenId, cid);
    emit TokenCreated(tokenId, msg.sender);
}
```

5.1.3 Decryption

The decryption process ensures that only authorised DP, who satisfy the attribute-based policy, can decrypt the ciphertext. The smart contract , AccessControl.sol plays a crucial role in controlling access by verifying the DP's attributes and managing the access token. The decryption process follows these steps:

1. **Requesting the Secret Key**: The DP must first request their secret key from the smart contract. This secret key is necessary for decrypting the ciphertext. The contract ensures that only DPs with the correct attributes (verified via a zero-knowledge proof) can access the secret key. The contract verifies the DP's attributes, stored earlier during the registration process.

   The registration of the DP's attributes is handled by the following part of the smart contract:

```
    mapping (address dpAddr => bytes32 attributesHash) internal
        dpAttrHash;

    function registerDP(bytes32 attributesHash) public {
        dpAttrHash[msg.sender] = attributesHash;
        emit DPRegistered(msg.sender, attributesHash);
    }
```

The request for the secret key is sent to the smart contract, and the contract checks if the DP has registered valid attributes.

```
    const handleSkRequest = async () => {
        if (!address || !tokenId) {
            setError('Please connect your wallet and enter a token ID'
                )
            return
        }

        try {
            const Processor = new ethers.Processors.Web3Processor(
                window.ethereum)
            const signer = await Processor.getSigner()

            const message = ethers.utils.solidityPack(['address', '
                uint256'], [address, tokenId])
            const messageHash = ethers.utils.keccak256(message)
            const signature = await signer.signMessage(ethers.utils.
                arrayify(messageHash))

            const response = await sendRequest({
                address,
                tokenId,
                signature,
                messageHash,
            })

            onSecretKeyRetrieved(response.secretKey)
```

```
        } catch (error: any) {
            setError(error.message)
        }
    }
```

2. **Fetching the Ciphertext from IPFS**: After the secret key is retrieved, the DP fetches the encrypted data, ciphertext from IPFS. The contract maps each token to its corresponding CID, which is used to retrieve the encrypted data from IPFS. This is handled by the following function in the contract:

```
    mapping (uint256 tokenId => string cid) public tokenIpfsHash;

    function getCid(uint256 tokenId) public view returns (string
        memory cid) {
        return tokenIpfsHash[tokenId];
    }
```

The DP uses this CID to retrieve the encrypted data from IPFS.

```
    const fetchCiphertext = async () => {
        const contractAddress = process.env.
            NEXT_PUBLIC_ARB_SEP_ACTK_ADDRESS
        const Processor = createPublicClient({
            chain: arbitrumSepolia,
            transport: http(`https://arb-sepolia.g.alchemy.com/v2/${
                process.env.NEXT_PUBLIC_ALCHEMY_ID}`)
        })

        const cid = await Processor.readContract({
            address: contractAddress,
            abi: AccessToken.abi,
            functionName: 'getCid',
            args: [BigInt(tokenId)]
        })

        const ipfsGateway = process.env.NEXT_PUBLIC_PINATA_GATEWAY
        const ipfsResponse = await fetch(`${ipfsGateway}${cid}`)
        const ipfsData = await ipfsResponse.json()
        setCiphertext(ipfsData.ciphertext)
```

```
    }
```

3. **Decrypting the Data**: Once the DP has both the secret key and the ciphertext, they can decrypt the data. The decryption occurs locally, ensuring privacy.
   The contract itself does not perform decryption but ensures that only authorised DPs can access the secret key and the CID. The decryption is handled by the DP's system using the secret key and ciphertext.

```
    const handleDecrypt = async () => {
        if (!secretKey || !ciphertext) {
            setError('Both secret key and ciphertext are required')
            return
        }

        try {
            const result = decrypt(secretKey, ciphertext)
            const decodedPlaintext = new TextDecoder().decode(result)
            setDecryptedText(decodedPlaintext)
            onDecryption(decodedPlaintext)
        } catch (err) {
            setError('Decryption failed')
        }
    }
```

The systematic flow from key generation to decryption ensures that only authorised DP's with the correct attributes can access sensitive data securely whilst protecting their attributes upon revelation. After fulfilling the access policy requirements and successfully verifying their attributes through a zero-knowledge proof, the DP receives the necessary secret key. DP can retrieve the ciphertext from IPFS and decrypt it using secret key and transform it to plaintext. The ciphertext is successfully deciphered, granting access only to those who satisfy the designated policy,thereby achieving the goal of privacy-preserving data sharing system.

5.2 Demonstration and Analysis

This section presents a comprehensive evaluation of the computational efficiency i.e., Time and Space Complexity, scalability and performance of the proposed ZK CP-ABE data sharing system.

5.2.1 Time Complexity

Time complexity measures the amount of time an algorithm requires to complete as a function of the input size [46]. It represents the algorithm's growth rate, allowing predictions of how execution time will increase as the input grows. In computational theory, this is usually denoted with Big-O notation (e.g., $O(1), O(n), O(n^2)$), which indicates an upper bound on time required for processing an input of a particular size. For example, an algorithm with $O(n)$ time complexity grows linearly with the size of the input. The time complexity for each operation in *lib.rs* which uses CP-ABE scheme using the RABE library are:

- **Setup:** The setup operation generates a public key and a master secret key using the `bsw::setup()` function. This operation does not depend on the size of the plaintext or the access policy. Therefore, the setup has a constant time complexity of $O(1)$.
- **Encryption:**The encryption function takes three inputs: the public key, the access policy, and the plaintext. In BSW CP-ABE, the time complexity of encryption depends on the number of attributes m in the access policy. Since each attribute in the policy requires processing, the time complexity of encryption is $O(m)$.
- **Key Generation:** The key generation function derives a secret key based on the user's attributes.The operation processes each attribute in the policy, hence the time complexity is $O(m)$ where m is the number of attributes in the policy.

- **Decryption:** In the decryption process, the user's secret key must satisfy the ciphertext's access policy. The time complexity of decryption in BSW CP-ABE is $O(m')$, where m' represents the minimum number of attributes required to satisfy the access policy. Only the attributes necessary to fulfill the policy need to be processed, making the complexity $O(m')$.

5.2.2 Space Complexity

Space complexity quantifies the amount of memory an algorithm requires based on the size of its input [46]. Represented in Big-O notation (e.g., $O(1), O(n), O(n^2)$), space complexity provides an upper bound on the memory usage of an algorithm for a given input size. This accounts for both the fixed memory allocations required for the algorithm to run and any additional memory consumed as input data size increases. Evaluation of Space Complexity is essential in resource-constrained environments because it helps in indicating how efficiently an algorithm uses memory and whether its memory requirements grow proportionally with input size. An algorithm with $O(1)$ space complexity, for instance, uses a constant amount of memory regardless of input size, while an algorithm with $O(n)$ space complexity consumes memory linearly as input size grows. The space complexity for each operation in *lib.rs* which uses CP-ABE scheme using the RABE library are:

- **Setup:** The space complexity of the setup function is $O(1)$ because it only stores the public key and master secret key. Both of these are fixed in size and do not depend on the input data.
- **Encryption:** During encryption, both the access policy and the plaintext are stored. The space complexity is $O(m)$, where m is the number of attributes.
- **Key Generation:** The space complexity for key generation is $O(m)$ because the secret key depends on the number of attributes in the policy.
- **Decryption:** The space complexity for decryption is also $O(m)$, as it involves storing the user's secret key and the ciphertext policy.

5.2.3 Encryption

The **encryption** process in CP-ABE encrypts a plaintext message based on an access policy defined by the data owner. This policy is embedded into the ciphertext, ensuring that only users with attributes satisfying the policy can decrypt the message. During encryption, the algorithm processes each attribute in the access policy to generate the corresponding ciphertext components, thereby embedding the access control mechanism into the encrypted data.

The encryption process iterates over all m attributes in the access policy. For each attribute, pairing-based cryptographic operations are performed to embed the policy into the ciphertext. As a result, the **time complexity** of encryption is linear with respect to the number of attributes. The time complexity can be expressed as:

$$O(m)$$

where m is the number of attributes in the access policy. As the number of attributes increases, the time taken for encryption increases proportionally. Each additional attribute adds complexity to the ciphertext, making the encryption process more time-consuming as the policy becomes more intricate.

To evaluate the encryption performance, tests were conducted using the RABE BSW CP-ABE scheme. The objective was to measure the encryption time for different policies with varying numbers of attributes. The tests dynamically generated policies with increasing numbers of attributes and recorded the average encryption time over multiple iterations.

The Rust-based implementation of the CP-ABE encryption scheme followed these key steps:

- Dynamically generated complex policies with a varying number of attributes, ranging from 2 to 1022.
- Encryption was performed using the `bsw::encrypt()` function, and the time taken for each encryption operation was recorded.
- The average encryption time for each policy size was calculated over 10 iterations to provide reliable performance metrics.

The policy generation code was designed to simulate real-world scenarios with complex nested conditions involving multiple roles, departments, and access levels. Below is an example of the policy generation function:

```
fn generate_complex_policy(num_attributes: usize) -> String {
    let mut base_policy = String::from("\"Certification:
       Advanced and (");
    let mut conditions = Vec::new();

    for i in 1..=num_attributes {
        let role = format!("(Role:Role{} and Department:
           Department{})", i, i);
        conditions.push(role);
    }

    conditions.push(String::from("(Role:Administrator and (
       Department:HR or Department:Finance))"));
    conditions.push(String::from("(AccessLevel:Full or
       AccessLevel:Limited)"));
    conditions.push(String::from("(Role:Intern and (Department:
       IT or Department:Research))"));
    conditions.push(String::from("(Mentor:Assigned or Mentor:
       Unassigned)"));
```

```
    conditions.push(String::from("(Role:Contractor and (Department:Security or Department:Maintenance) and (SecurityClearance:High or SecurityClearance:Medium))"));

    base_policy.push_str(&conditions.join(" or "));
    base_policy.push(')');
    base_policy.push('"');
    base_policy
}
```

The encryption process was run for policies containing different numbers of attributes, with the number of attributes increasing from 2 to 1022. Each encryption run was timed using the `Instant::now()` function in Rust [105], and the average time was computed for each attribute set over 10 iterations. The following code snippet was used to generate policies, encrypt data, and measure encryption time:

```
let attributes_to_test: Vec<usize> = (2..=1022).step_by(10).collect();
let mut summary = Vec::new();

for num_attributes in &attributes_to_test {
    let mut total_duration = 0.0;
    for i in 0..iterations {
        let policy = generate_complex_policy(*num_attributes);
        let start = Instant::now();
        let _ct = bsw::encrypt(&pk, &policy, rabe::utils::policy::pest::PolicyLanguage::HumanPolicy, plaintext)
            .expect("Encryption failed");
        let duration = start.elapsed().as_secs_f64();
```

```
12          total_duration += duration;
13      }
14
15      let average_time = total_duration / iterations as f64;
16      summary.push((*num_attributes, average_time));
17      println!("Attributes: {}, Average time: {:.2} seconds",
            num_attributes, average_time);
18  }
```

The test results for policies containing 2 to 1022 attributes show a clear trend where encryption time increases as the number of attributes grows. Below is a table summarizing the average encryption times for selected policy sizes:

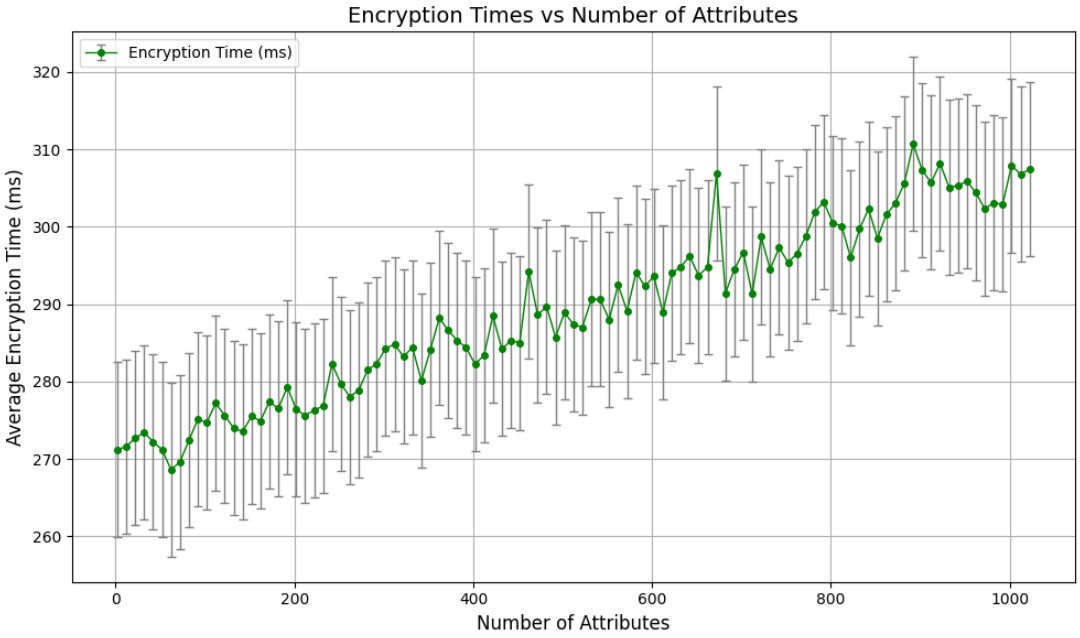

Figure 5.1: Average Encryption Time vs Number of Attributes

The graph (Figure 5.1) highlights a gradual rise in encryption time as the number of attributes increases. This is expected due to the linear time complexity of the encryption algorithm. The test results provide the following insights:

Number of Attributes	Average Encryption Time (milliseconds)
2	271.20
12	271.60
62	268.60
252	279.70
502	287.00
752	303.20
1022	307.40

Table 5.1: Encryption Time Performance for Varying Policy Sizes

- **Scalability**: The system exhibits efficient scalability with policies containing up to 500 attributes. For smaller attribute sets, the encryption time increases at a consistent, gradual rate. This confirms the linear relationship between the number of attributes and the time complexity $O(m)$, where m represents the number of attributes. This implies that for each additional attribute, the increase in computational work required remains manageable, and the overall system performs well without any significant bottlenecks.

  However, the encryption time increases sharply as the number of attributes grows beyond 500. This deviation can be attributed to the cumulative effect of pairing-based cryptographic operations. While the time complexity is still linear, larger policies introduce a greater number of pairing and exponentiation operations, which increases the computational burden. Although the system remains scalable, it becomes more resource-intensive as policy size expands.
- **Performance Trend**: For policies with a smaller number of attributes (ranging from 2 to 500), the encryption times follow a predictable, near-linear trend. This steady behavior aligns with the theoretical time complexity $O(m)$, as each additional attribute adds a fixed amount of computational effort to the encryption process. In this range, the algorithm efficiently handles the pairing operations required for each attribute, maintaining relatively stable performance across different policy sizes.

  Beyond 500 attributes, however, the performance trend changes more noticeably. The encryption time increases at a higher rate, reflecting the growing computational overhead. As policies become more complex with a larger number of attributes, the system needs to perform a greater number of cryptographic operations (e.g.,

bilinear pairings and exponentiations) for each attribute. Although these operations are still processed in linear time, the increased volume of data and the interaction between the components (e.g., attributes and access structures) introduce additional complexity, leading to a higher cumulative encryption time.

This trend continues as the number of attributes approaches 1000 and beyond, where the encryption process becomes significantly more time-consuming. This behavior highlights the linear nature of the algorithm but also indicates the practical limitations of handling policies with excessively large numbers of attributes. The larger the policy, the greater the overhead associated with managing and embedding each attribute into the ciphertext, causing the encryption time to rise correspondingly.

5.2.4 Key Generation

During key generation, a **secret key** is derived based on the user's attributes. The access policy, which is embedded in the ciphertext, defines the conditions under which decryption is possible. The key generation process evaluates the user's attributes against the policy. For each attribute in the policy, the algorithm performs cryptographic operations to derive the corresponding elements of the secret key.

- **Attributes:**

  Let m represent the number of attributes specified in the policy. The key generation algorithm must process each of these m attributes individually to generate the corresponding secret key. For example, if the policy includes conditions such as `Role:Doctor AND Department:Oncology` , the algorithm checks both the `Role` and `Department` attributes. Each attribute in the user's set of attributes is compared to those in the policy, and appropriate cryptographic operations (such as pairing-based computations) are performed to generate the key. The **time complexity** of this process is linear with respect to the number of attributes in the policy. Thus, the

time complexity for key generation can be expressed as [22]:

$$O(m)$$

where m is the number of attributes in the access policy. As the number of attributes increases, the time required to generate the secret key increases linearly, since each attribute requires cryptographic operations to check its satisfaction of the policy.

5.2.5 Decryption

The **decryption** process in CP-ABE uses the user's secret key to decrypt the ciphertext. The ciphertext contains an embedded access policy, and decryption is only successful if the user's attributes satisfy this policy[22]. During decryption, the algorithm checks whether the user's secret key (derived from their attributes) fulfills the policy requirements embedded in the ciphertext.

- **Attributes:**

  In contrast to iterating over all m attributes in the policy, decryption only requires processing the subset of attributes m' that satisfy the access policy. The decryption algorithm performs pairing-based operations for each attribute necessary to meet the policy requirements, matching them against corresponding components in the secret key.

  The **time complexity** of decryption, therefore, depends on m', the minimum number of attributes required to satisfy the policy, not the total number of attributes in the policy. Consequently, the time complexity for decryption can be expressed as [22]:

  $$O(m')$$

where m' is the minimum number of attributes needed to satisfy the access policy. As the number of required attributes increases, the time taken for decryption increases linearly. Each necessary attribute must be checked and verified, making the process more time-consuming as the policy requirements become more complex.

The proposed system integrates CP-ABE with blockchain technology and ZKP to create a novel privacy-preserving data sharing system. Performance evaluation measures execution times and computational complexity relative to the number of attributes by examining distinct components of the system i.e., ABE , ZKP and Smart Contract execution. As detailed in the subsections below, the results such as encryption and decryption times and the breakdown of proof generation and verification times which gives an insight about the performance. Direct comparisons with other approaches are avoided because there's no common baseline for a fair evaluation as the solution combines a decentralised blockchain technology , ZKP, and ABE to ensure that only authorised users can access specific data, while preserving the privacy of user attributes.

5.2.6 Performance Testing of Proof Generation and Verification

To evaluate the efficiency and scalability of the zkServer's zero-knowledge proof (ZKP) generation and verification processes, comprehensive performance testing was conducted. The tests focused on measuring the time required for both operations as the number of attributes in the policy varied. Specifically, the attribute counts ranged from **2** to **1000**, enabling insights into how increasing complexity impacts system performance. This section outlines the testing methodology, results, and implications for scalability.

The performance tests were executed as follows:

- **Attributes Generation:** The function `generate_attributes(num_attributes)` was used to generate a list of attributes in JSON format, where each attribute was represented as a key-value pair, such as:

```
{ "Attr1": "Value1",
 "Attr2": "Value2",
        ... }
```

  The number of attributes generated was incrementally increased for each test case, from **2** to **1000**, providing a clear spectrum of scalability.
- **Policy Definition:** A unique policy was defined for each attribute set, simulating varying complexities in access control. Each policy corresponded to a different level of interaction with the attributes, directly influencing the proof generation and verification times.
- **ZKP Generation and Verification:** The `prover::generate_proof()` function was invoked for each policy to generate the ZKP. Proof generation was subject to a **10-minute timeout** to prevent excessive runtime for complex cases. Once generated, the proof was verified, and both operations were timed. Each test case was repeated **ten times** to ensure reliability, and the average time was recorded.
- **Token ID:** A static token ID (`U256::from(1234)`) was used to simulate production-like conditions, ensuring that the token's ID was handled uniformly across all tests.
- **Performance Measurement:** The `Instant::now()` function from Rust [105] was utilised to measure the duration of both proof generation and verification. The total time for each test case was accumulated over **ten iterations** and the average time for each successful operation was computed.

The tests covered a good range of attribute sizes, starting from **2 attributes and extending up to 1000 attributes**. Performance data was analysed for proof generation times, measured in **milliseconds**, and proof verification times, measured in **nanoseconds**.

The performance of both the proof generation and verification processes is analysed as follows:

Metric	Proof Generation Time (ms)	Proof Verification Time (ns)
Minimum Time	29.29	41.00
Maximum Time	42.90	42.00
Average Time	38.76	41.87
Standard Deviation	2.87	0.34

Table 5.2: Summary of Proof Generation and Verification Performance with Standard Deviation

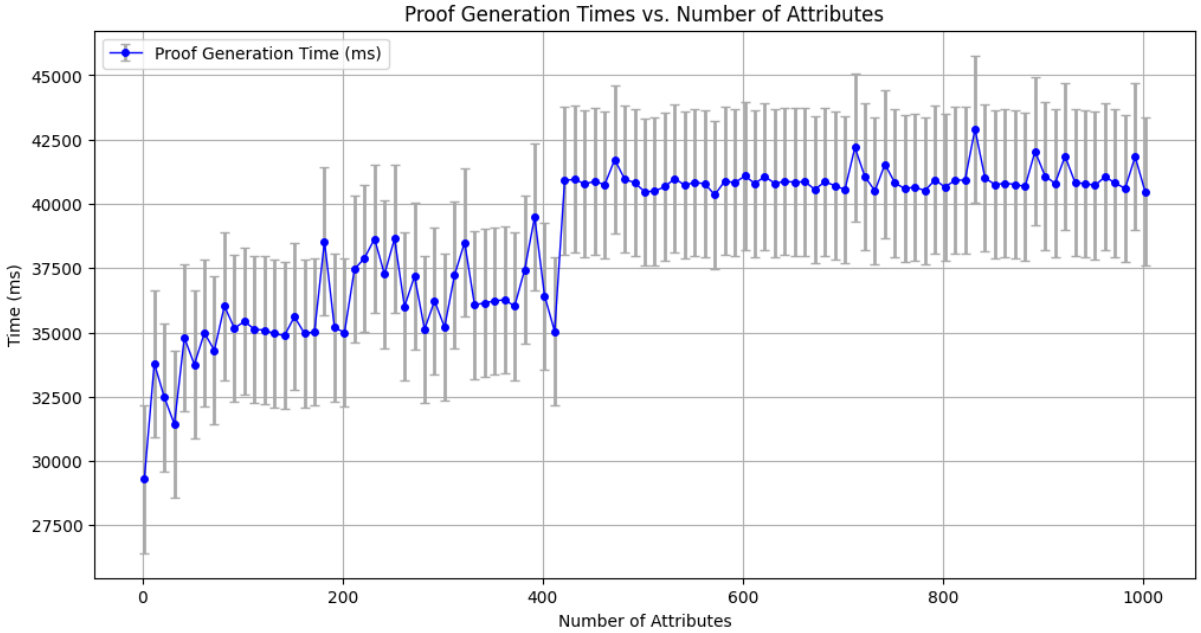

Figure 5.2: Proof Generation Time for Varying Numbers of Attributes

- **Scalability:** The results from the graphs show that proof generation times increase linearly as the number of attributes grows. For smaller attribute sets (below 200 attributes), proof generation times range between approximately 30,000 to 35,000 milliseconds. As the number of attributes approaches 1000, the generation time stabilises around 40,000 milliseconds. This consistent linear growth indicates that the system scales predictably with larger attribute sets. Even for high attribute counts, the performance does not degrade beyond the expected trend, which suggests the system is well-suited to handle increasing complexity in real-world applications. In contrast, proof verification times, measured in nanoseconds, remain stable across all tested attribute counts. Whether the attribute size is 2 or 1000, the verification time stays between 41 and 42 nanoseconds. The minimal fluctuation in verification

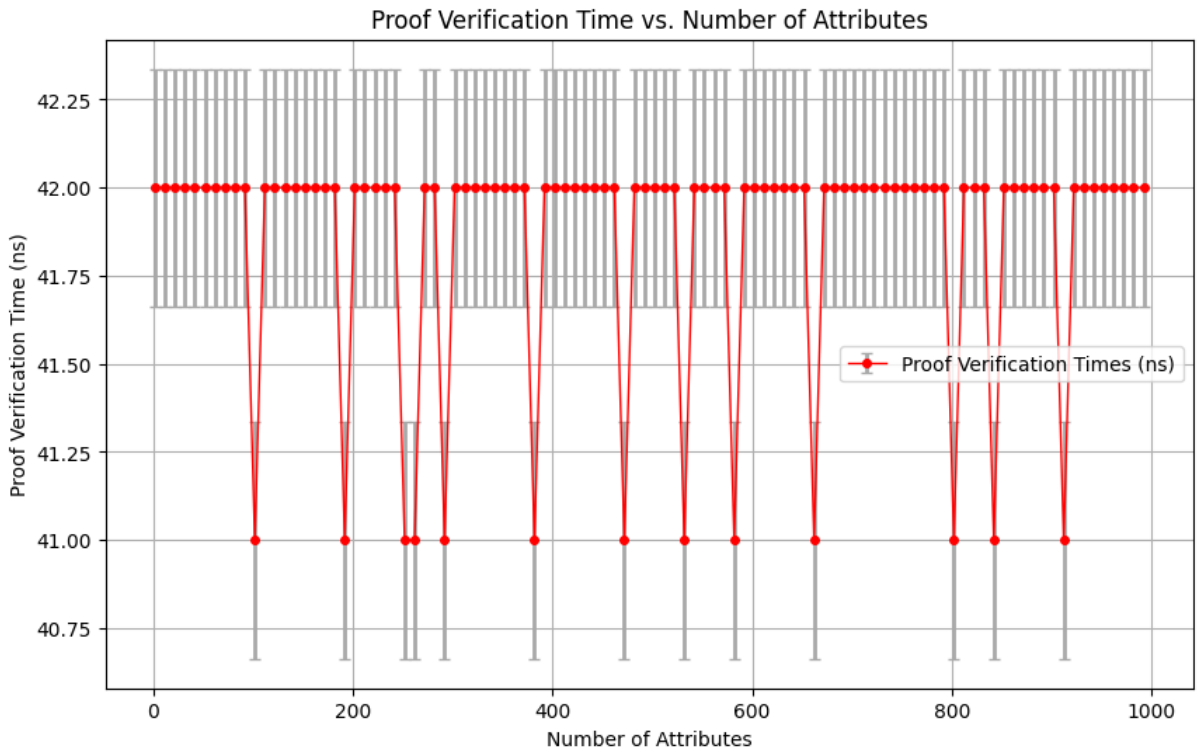

Figure 5.3: Proof Verification Time for Varying Numbers of Attributes

time highlights the efficiency of the verification process, making it computationally lightweight. Regardless of attribute count, this consistency in verification times shows that the system can efficiently handle complex policies without introducing additional computational overhead during the verification phase.

- **Performance:** The system's performance remains highly efficient, even as the complexity of the policies grows. While proof generation times increase with more attributes, the linear progression shows that the system can handle moderate to large numbers of attributes without significant degradation. For larger attribute sets (up to 1000), generation times remain within a reasonable and predictable range. The results showcase system's predictability and performance consistency. Proof verification times are extremely fast and constant. The verification process remains quick across all tests irrespective of the attribute count, with times consistently around 41–42 nanoseconds. This efficiency is critical for scenarios requiring frequent proof

verification, as the negligible overhead ensures the system can verify proofs at high throughput. Overall, the system demonstrates that it can scale effectively, with proof generation times growing predictable and linearly and proof verification times remaining constant.

5.2.7 Performance Testing of Proof Generation with Dynamic Policies and Data Attributes

Performance testing was conducted for generating ZKPs using zkServer with dynamically generated policies and data attributes. The aim of this test was to evaluate how different combinations of attribute set sizes and policy complexities impact the time required to generate a proof. The performance testing focused on attribute set sizes ranging from 10 to 100 and policy sizes varying from 10 to 100 attributes.

The testing process was conducted as follows:

- **Attribute Generation:** Attributes were dynamically generated using the `generate_attributes(num_attributes)` function, which produced a JSON object containing key-value pairs for each attribute. For example:

  ```
  { "Attr1": "Value1",
   "Attr2": "Value2",
          ... }
  ```

  Attribute set sizes were incrementally increased from 10 to 100, with tests executed for each step size of 10 attributes.
- **Policy Definition:** For each test, a dynamic policy was created using the `generate_policy(num_attributes)` function, which combined attribute conditions with an OR operator and an AND operator. Each policy's complexity grew with the number of attributes involved, providing a varying level of interaction between the attributes. The general structure of the policies was as follows:

```
("Attr1:Value1" OR "Attr2:Value2") AND
("Attr3:Value3" OR "Attr4:Value4") AND ...
("AttrN-1:ValueN-1" OR "AttrN:ValueN")
```

As the number of attributes increased, the policies became more complex. Thereby reflecting a policy with growing number of conditions combined with the OR operator.

- **ZKP Generation:** The `prover::generate_proof()` function was invoked for each combination of policy and attributes. To prevent excessive runtime, a **10-minute timeout** was enforced for proof generation. This ensured that no single test case dominated the testing time. Each test was repeated to ensure reliability, and the average time for successful proof generation was recorded.
- **Performance Measurement:** The time taken to generate a proof was measured using the `Instant::now()` function from Rust [105]. For each successful proof generation, the total time was recorded in **milliseconds** and averaged over **five iterations** for accuracy. These times were logged for later analysis.

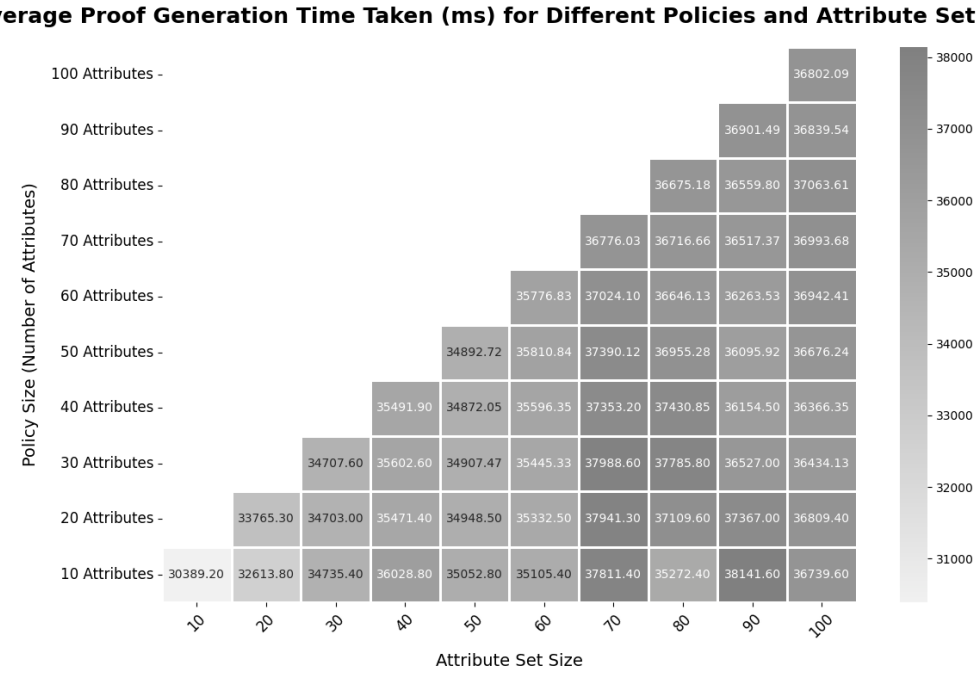

Figure 5.4: Proof Generation Time for Varying Numbers of Attributes within Policy and Attribute set sizes.

- **Scalability:** As shown in the heatmap (Figure 5.4) and table, proof generation times generally increase as the number of attributes and policy size grows. For small attribute sets (10–50 attributes), the time ranges between **30,000** and **35,000 milliseconds**. However, with larger sets (above 50 attributes), proof generation times stabilise around **36,000–38,000 milliseconds**. This linear progression indicates that the system can handle moderate to large attribute sets without significant degradation in performance.
- **Performance:** The overall performance remains consistent across varying policy sizes, showing predictable scaling as the number of attributes increases. The system can handle increasing complexities with predictable proof generation times, making it suitable for applications involving dynamic policy definitions and large attribute sets.

5.2.8 Gas Impact

Gas efficiency is a critical factor in evaluating the feasibility of blockchain-based smart contracts. The gas costs associated with different functions of the *AccessToken* smart contract were analysed based on executed transactions on the Arbitrum Sepolia Testnet.

The table 5.3 highlights the computational costs associated with different functions in the *AccessToken* smart contract. The `registerDP` function consumes 95,157 gas, placing it in the low gas category, indicating efficiency in registering a Data Processor. The `createToken` function requires 149,054 gas, categorising it as medium gas usage due to operations such as hashing and storage updates. However, the `mintAccessTokenForDP` function incurs a significantly higher gas cost of 309,601 gas, primarily due to cryptographic verification of Zero-Knowledge Proofs (ZKP) and token minting.Optimisation categories, including storage efficiency, proof verification enhancements, and mapping structure refinements, are currently being worked on to further improve gas efficiency.

Function	Gas Used	Function Purpose
`createToken`	149,054	The **Data Owner (DO)** calls this function to create an access token. This function generates a unique token ID using the hash of the caller's address and an incrementing counter. It assigns the ownership of the token to the Data Owner, records the token in the owner's balance, and stores an IPFS hash representing the encrypted data.
`registerDP`	95,157	This function registers a **Data Processor (DP)** by storing their **hashed attributes**. The attributes act as an identity verification mechanism to determine eligibility for accessing encrypted data. The function simply updates a mapping that associates the DP's address with their attribute hash.
`mintAccessTokenForDP`	309,601	The **Data Processor (DP)** calls this function to request access to a token. It verifies that the DP has been registered and that their attributes match the pre-registered hash. It then verifies a **Zero-Knowledge Proof (ZKP)** provided by the DP, ensuring that they meet the required policy conditions without exposing the attributes themselves. Once verification is successful, the function mints an ERC-1155 token to the DP's address and grants them access.

Table 5.3: Gas Usage Analysis.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This research presents a novel privacy-preserving data-sharing framework that integrates Ciphertext-Policy Attribute-Based Encryption (CP-ABE) with Zero-Knowledge Proofs (ZKP) within a decentralised blockchain environment.

The primary challenge addressed in this research is the need to protect user attributes, which data processors typically have to reveal when accessing information. Conventional Ciphertext-Policy Attribute-Based Encryption (CP-ABE) systems enforce selective access control but often require attributes to be disclosed during verification, compromising user privacy. This exposure is particularly concerning in sensitive fields such as healthcare and finance. This research introduces a Zero-Knowledge Ciphertext-Policy Attribute-Based Encryption (ZK CP-ABE) data-sharing framework to address these limitations where we utilise two privacy preservation techniques, Proof (ZKP) and CP-ABE. This framework enables access verification without revealing user attributes, empowering data processors to verify access rights while maintaining the confidentiality of underlying attributes.

To solve the challenge of attribute disclosure, this research presents a novel Zero-Knowledge Ciphertext-Policy Attribute-Based Encryption (ZK CP-ABE) framework. Integrating Zero-Knowledge Proofs (ZKP) with Ciphertext-Policy Attribute-Based Encryption (CP-ABE) enables data processors to satisfy access policy without exposing sensitive attributes. The ZKP component allows privacy-preserving verification, ensuring users can prove their authorization to access specific data without revealing the actual attributes involved. The InterPlanetary File System (IPFS) for decentralised storage is used in the proposed data sharing system which reduces reliance on centralised systems and enhancing data security, scalability, and resilience. The IPFS integration ensures that data remains accessible and verifiable through decentralised networks, with only data hashes stored on-chain.

The encryption was thoroughly tested using policies with varying attribute counts to assess scalability and performance. Results showcased a linear relationship between encryption time and policy size, reflecting efficient handling of moderate to large attribute sets. Policies containing up to 500 attributes saw a consistent, gradual increase in encryption time, while policies exceeding 500 attributes demonstrated a sharper rise due to the cumulative effect of pairing-based cryptographic operations.

Performance testing of the proposed privacy preserving data sharing system reveals notable efficiency in proof generation and verification times,critical for ensuring usability of the system in real-world applications. The tests encompassed a range of policy complexities and attribute set sizes to simulate dynamic, real-world conditions where policies and access requirements vary frequently. The tests demonstrated scalability in proof generation times, with predictable increases as attribute counts grew. For smaller sets (10–50 attributes), generation times ranged from 30,000 to 35,000 milliseconds, while larger sets (above 50 attributes) remained stable at around 36,000–38,000 milliseconds. Verification times maintained a steady 41–42 nanoseconds across varying attribute sizes within policies. This consistency in proof verification ensures the system's suitability for high-throughput environments, affirming its robustness and adaptability for real-world, privacy-sensitive applications.

6.2 Future Work

The proposed framework offers several significant advantages in the context of decentralised, privacy-sensitive environments. One of the key strengths is its robust privacy protection through the combination of Ciphertext-Policy Attribute-Based Encryption (CP-ABE) and Zero-Knowledge Proofs (ZKP). CP-ABE ensures fine-grained access control, allowing only authorised users to decrypt data based on specific attributes, which is vital in securing sensitive information. ZKP enhances this by allowing access validation without exposing sensitive attributes, maintaining confidentiality in scenarios where revealing user identity or attributes could be detrimental.

Future research could focus on improving the computational overhead associated with Zero-Knowledge Proofs and CP-ABE, especially in environments with larger attribute sets and more complex policies. Exploring efficient cryptographic techniques, i.e., combining multiple cryptographic mechanisms like homomorphic encryption with ZKP, could improve performance in resource-constrained environments. Introducing a multi-authority CP-ABE scheme could enhance scalability and flexibility. Multi-authority models would distribute the responsibility of attribute management across several entities, allowing for more decentralised and robust attribute verification in complex networks.

Exploring ongoing advancements in legal standards for decentralised systems, such as emerging standards for digital identity and decentralised storage under the GDPR and HIPAA, could also support wider application of the proposed framework. Addressing regulatory and legal constraints, the framework may offer a secure, privacy-compliant solution adaptable to sectors requiring stringent data confidentiality.

Bibliography

[1] John M Abowd. 'The US Census Bureau adopts differential privacy'. In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2018, pp. 2867–2867.

[2] Fraunhofer AISEC. *RABE - Rust Attribute-Based Encryption Library*. `https://github.com/Fraunhofer-AISEC/rabe`. 2024.

[3] Wilson Abel Alberto Torres et al. 'Post-quantum one-time linkable ring signature and application to ring confidential transactions in blockchain (lattice RingCT v1. 0)'. In: *Information Security and Privacy: 23rd Australasian Conference, ACISP 2018, Wollongong, NSW, Australia, July 11-13, 2018, Proceedings 23*. Springer. 2018, pp. 558–576.

[4] Steffen Albrecht et al. 'Blockchain Technology for Industrial Applications: A Systematic Literature Review'. In: *2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*. IEEE, 2018, pp. 1–9. DOI: `10.1109/ICE.2018.8436274`.

[5] Afnan Alsadhan, Areej Alhogail and Hessah Alsalamah. 'Blockchain-Based Privacy Preservation for the Internet of Medical Things: A Literature Review'. In: *Electronics* 13.19 (2024). ISSN: 2079-9292. DOI: `10.3390/electronics13193832`. URL: `https://www.mdpi.com/2079-9292/13/19/3832`.

[6] Marcin Andrychowicz et al. 'Secure multiparty computations on bitcoin'. In: *Communications of the ACM* 59.4 (2016), pp. 76–84.

[7] Andreas M Antonopoulos and Gavin Wood. *Mastering ethereum: building smart contracts and dapps*. O'reilly Media, 2018.

[8] Alexander Asplund and Peter F Hartvigsen. *Reclaiming Data Ownership: Differential Privacy in a Decentralized Setting*. 2015. URL: `https://api.semanticscholar.org/CorpusID:55245471`.

[9] James Aspnes, Hagit Attiya and Keren Censor. 'Combining shared-coin algorithms'. In: *Journal of Parallel and Distributed Computing* 70.3 (2010), pp. 317–322.

[10] Nuttapong Attrapadung, Benoît Libert and Elie De Panafieu. 'Expressive key-policy attribute-based encryption with constant-size ciphertexts'. In: *Public Key Cryptography–PKC 2011: 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings 14*. Springer. 2011, pp. 90–108.

[11] Muhammad Ajmal Azad et al. 'Verify and trust: A multidimensional survey of zero-trust security in the age of IoT'. In: *Internet of Things* 27 (2024), p. 101227.

[12] Soumya Banerjee et al. 'Private blockchain-envisioned multi-authority CP-ABE-based user access control scheme in IIoT'. In: *Computer Communications* 169 (2021), pp. 99–113.

[13] Sourav Banerjee et al. 'Study and survey on blockchain privacy and security issues'. In: *Cross-industry use of Blockchain Technology and Opportunities for the Future*. IGI Global, 2020, pp. 80–102.

[14] Nishtha Baria, Dharmil Parmar and Vidhi Panchal. 'Blockchain User, Network and System-Level Attacks and Mitigation'. In: *The Auditor's Guide to Blockchain Technology*. CRC Press, 2022, pp. 223–243.

[15] Eli Ben-Sasson et al. 'Scalable, transparent, and post-quantum secure computational integrity'. In: *Cryptology ePrint Archive* (2018).

[16] Eli Ben-Sasson et al. 'Zerocash: Decentralized Anonymous Payments from Bitcoin'. In: *2014 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2014, pp. 459–474. DOI: `10.1109/SP.2014.36`.

[17] Ali Benabdallah et al. 'Analysis of Blockchain Solutions for E-Voting: A Systematic Literature Review'. In: *IEEE Access* (2022).

[18] Juan Benet. 'IPFS - Content Addressed, Versioned, P2P File System'. In: *arXiv preprint arXiv:1407.3561* (2014).

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

[20] Fabrice Benhamouda, Shai Halevi and Tzipora Halevi. 'Supporting private data on hyperledger fabric with secure multiparty computation'. In: *IBM Journal of Research and Development* 63.2/3 (2019), pp. 3–1.

[21] Elisa Bertino, Ahish Kundu and Zehra Sura. 'Data transparency with blockchain and AI ethics'. In: *Journal of Data and Information Quality (JDIQ)* 11.4 (2019), pp. 1–8.

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

[23] Karthikeyan Bhargavan et al. 'Formal verification of smart contracts: Short paper'. In: *Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security*. ACM. 2016, pp. 91–96.

[24] Alex Biryukov and Sergei Tikhomirov. 'Security and privacy of mobile wallet users in Bitcoin, Dash, Monero, and Zcash'. In: *Pervasive and Mobile Computing* 59 (2019), p. 101030.

[25] Nir Bitansky et al. 'From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again'. In: *Proceedings of the 3rd innovations in theoretical computer science conference*. 2012, pp. 326–349.

[26] Nir Bitansky et al. 'Recursive composition and bootstrapping for SNARKs and proof-carrying data'. In: *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. 2013, pp. 111–120.

[27] Manuel Blum et al. 'Noninteractive zero-knowledge'. In: *SIAM Journal on Computing* 20.6 (1991), pp. 1084–1118.

[28] Rakesh Bobba, Himanshu Khurana and Manoj Prabhakaran. 'Attribute-sets: A practically motivated enhancement to attribute-based encryption'. In: *Computer Security–ESORICS 2009: 14th European Symposium on Research in Computer Security, Saint-Malo, France, September 21-23, 2009. Proceedings 14*. Springer. 2009, pp. 587–604.

[29] Alexander Bogdanov et al. 'Solving the Problems of Byzantine Generals Using Blockchain Technology'. In: *Proceedings of the 9th International Conference" Distributed Computing and Grid Technologies in Science and Education"(GRID'2021), Dubna, Russia*. 2021, pp. 573–578.

[30] Joseph Bonneau et al. 'Mixcoin: Anonymity for bitcoin with accountable mixes'. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2014, pp. 486–504.

[31] Elette Boyle et al. 'Practical fully secure three-party computation via sublinear distributed zero-knowledge proofs'. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019, pp. 869–886.

[32] Benedikt Bünz et al. 'Zether: Towards privacy in a smart contract world'. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2020, pp. 423–443.

[33] Vitalik Buterin. *A Next-Generation Smart Contract and Decentralized Application Platform*. 2014.

[34] Vitalik Buterin. 'Ethereum: platform review'. In: *Opportunities and Challenges for Private and Consortium Blockchains* 45 (2016).

[35] Dongliang Cai et al. 'Attribute-Based Encryption With Payable Outsourced Decryption Using Blockchain and Responsive Zero Knowledge Proof'. In: *arXiv preprint arXiv:2411.03844* (2024).

[36] Fran Casino, Thomas K. Dasaklis and Constantinos Patsakis. 'A Systematic Literature Review of Blockchain-Based Applications: Current Status, Classification and Open Issues'. In: *Telematics and Informatics* 36 (2019), pp. 55–81. DOI: 10.1016/j.tele.2018.11.006.

[37] Wenqiang Chai et al. 'Blockchain-based Privacy-Preserving Electronic Voting Protocol'. In: *International Journal of Network Security* 24.2 (2022), pp. 230–237.

[38] Melissa Chase. 'Multi-authority attribute based encryption'. In: *Theory of Cryptography: 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007. Proceedings 4*. Springer. 2007, pp. 515–534.

[39] Melissa Chase and Sherman SM Chow. 'Improving privacy and security in multi-authority attribute-based encryption'. In: *Proceedings of the 16th ACM conference on Computer and communications security*. 2009, pp. 121–130.

[40] David L Chaum. 'Untraceable electronic mail, return addresses, and digital pseudonyms'. In: *Communications of the ACM* 24.2 (1981), pp. 84–90.

[41] Shekha Chenthara et al. 'Healthchain: A novel framework on privacy preservation of electronic health records using blockchain technology'. In: *Plos one* 15.12 (2020), e0243043.

[42] Sofie Christensen. 'A Comparative Study of Privacy-Preserving Cryptocurrencies: Monero and ZCash'. In: *School of Computer Science University of Birmingham* (2018).

[43] Konstantinos Christidis and Michael Devetsikiotis. 'Blockchains and smart contracts for the internet of things'. In: *Ieee Access* 4 (2016), pp. 2292–2303.

[44] Mauro Conti et al. 'A survey on security and privacy issues of bitcoin'. In: *IEEE Communications Surveys & Tutorials* 20.4 (2018), pp. 3416–3452.

[45] wagmi Contributors. *wagmi Documentation*. 2023. URL: `https://wagmi.sh/`.

[46] Thomas H Cormen et al. *Introduction to algorithms*. MIT press, 2022.

[47] Nicolas T Courtois and Rebekah Mercer. 'Stealth address and key management techniques in blockchain systems'. In: *ICISSP 2017-Proceedings of the 3rd International Conference on Information Systems Security and Privacy*. 2017, pp. 559–566.

[48] Ronald Cramer, Ivan Damgård and Ueli Maurer. 'General secure multi-party computation from any linear secret-sharing scheme'. In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2000, pp. 316–334.

[49] Yueyue Dai et al. 'Blockchain empowered access control for digital twin system with attribute-based encryption'. In: *Future Generation Computer Systems* (2024).

[50] Chris Dannen. *Introducing Ethereum and solidity*. Vol. 1. Springer, 2017.

[51] Stefano De Angelis et al. 'PBFT vs proof-of-authority: Applying the CAP theorem to permissioned blockchain'. In: *CEUR workshop proceedings*. Vol. 2058. CEUR-WS. 2018, pp. 1–11.

[52] Dominic Deuber and Dominique Schröder. 'CoinJoin in the Wild: An Empirical Analysis in Dash'. In: *Computer Security–ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part II 26*. Springer. 2021, pp. 461–480.

[53] Marten van Dijk et al. 'Fully homomorphic encryption over the integers'. In: *Annual international conference on the theory and applications of cryptographic techniques*. Springer. 2010, pp. 24–43.

[54] Trinh Viet Doan et al. 'Toward decentralized cloud storage with IPFS: opportunities, challenges, and future considerations'. In: *IEEE Internet Computing* 26.6 (2022), pp. 7–15.

[55] Ali Dorri et al. 'Blockchain: A distributed solution to automotive security and privacy'. In: *IEEE Communications Magazine* 55.12 (2017), pp. 119–125.

[56] Wenliang Du and Mikhail J Atallah. 'Secure multi-party computation problems and their applications: a review and open problems'. In: *Proceedings of the 2001 workshop on New security paradigms*. 2001, pp. 13–22.

[57] Evan Duffield and Daniel Diaz. *Dash: A privacycentric cryptocurrency*. 2015.

[58] Cynthia Dwork. 'Differential privacy'. In: *International colloquium on automata, languages, and programming*. Springer. 2006, pp. 1–12.

[59] Cynthia Dwork. 'Differential privacy: A survey of results'. In: *International conference on theory and applications of models of computation*. Springer. 2008, pp. 1–19.

[60] Cynthia Dwork and Aaron Roth. 'The Algorithmic Foundations of Differential Privacy'. In: *Found. Trends Theor. Comput. Sci.* 9.3–4 (2014), pp. 211–407. ISSN: 1551-305X. DOI: 10.1561/0400000042. URL: https://doi.org/10.1561/0400000042.

[61] Cynthia Dwork, Aaron Roth et al. 'The algorithmic foundations of differential privacy'. In: *Foundations and Trends® in Theoretical Computer Science* 9.3–4 (2014), pp. 211–407.

[62] Keita Emura et al. 'A Ciphertext-Policy Attribute-Based Encryption Scheme with Constant Ciphertext Length'. In: *Information Security Practice and Experience (ISPEC 2009)*. Vol. 5451. Lecture Notes in Computer Science. Springer, 2009, pp. 13–23. DOI: 10.1007/978-3-642-00843-6_2.

[63] Parisa Esmaeilzadeh. 'The Role of Blockchain Technology in the Healthcare Sector: A Systematic Review'. In: *Healthcare* 7.4 (2019), p. 56. DOI: 10.3390/healthcare7040122.

[64] Anna Essén and Anders Ekholm. 'Centralization vs. decentralization on the blockchain in a health information exchange context'. In: Oct. 2019, pp. 58–82. ISBN: 9780429319297. DOI: 10.4324/9780429319297-4.

[65] Ittay Eyal and Emin Gün Sirer. 'Majority is Not Enough: Bitcoin Mining is Vulnerable'. In: *Association for Computing Machinery* 61.7 (2018), pp. 95–102. ISSN: 0001-0782. DOI: 10.1145/3212998. URL: https://doi.org/10.1145/3212998.

[66] Weidong Fang et al. 'Digital signature scheme for information non-repudiation in blockchain: a state of the art review'. In: *EURASIP Journal on Wireless Communications and Networking* 2020.1 (2020), pp. 1–15.

[67] Qi Feng et al. 'A survey on privacy protection in blockchain system'. In: *Journal of Network and Computer Applications* 126 (2019), pp. 45–58.

[68] Uriel Fiege, Amos Fiat and Adi Shamir. 'Zero knowledge proofs of identity'. In: *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. 1987, pp. 210–217.

[69] R. French-Baidoo, D. Asamoah and S. O. Oppong. 'Achieving confidentiality in electronic health records using cloud systems'. In: *International Journal of Computer Network and Information Security* 10 (1 2018), pp. 18–25. DOI: 10.5815/ijcnis.2018.01.03.

[70] H. E. Gafif and T. Ahmed. 'Efficient ciphertext-policy attribute-based encryption constructions with outsourced encryption and decryption'. In: *Security and Communication Networks* 2021 (2021), pp. 1–17. DOI: 10.1155/2021/8834616.

[71] Keke Gai et al. 'Differential privacy-based blockchain for industrial internet-of-things'. In: *IEEE Transactions on Industrial Informatics* 16.6 (2019), pp. 4156–4165.

[72] Chandana Gamage et al. 'An identity-based ring signature scheme with enhanced privacy'. In: *2006 Securecomm and Workshops*. IEEE. 2006, pp. 1–5.

[73] Simson L Garfinkel, John M Abowd and Sarah Powazek. 'Issues encountered deploying differential privacy'. In: *Proceedings of the 2018 Workshop on Privacy in the Electronic Society*. 2018, pp. 133–137.

[74] Christina Garman, Ian Miers and Matthew Green. *RFC: Decentralized Anonymous Credentials*. Tech. rep. Johns Hopkins University Department of Computer Science, 2013.

[75] Aijun Ge et al. 'Threshold ciphertext policy attribute-based encryption with constant size ciphertexts'. In: *Information Security and Privacy: 17th Australasian Conference, ACISP 2012, Wollongong, NSW, Australia, July 9-11, 2012. Proceedings 17*. Springer. 2012, pp. 336–349.

[76] Craig Gentry. *A fully homomorphic encryption scheme*. Stanford university, 2009.

[77] Oded Goldreich, Silvio Micali and Avi Wigderson. 'Proofs That Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems'. In: *Journal of the ACM* 38.3 (1991), pp. 690–728. DOI: 10.1145/116825.116852.

[78] Oded Goldreich and Yair Oren. 'Definitions and properties of zero-knowledge proof systems'. In: *Journal of Cryptology* 7.1 (1994), pp. 1–32.

[79] Shafi Goldwasser, Silvio Micali and Chales Rackoff. 'The knowledge complexity of interactive proof-systems'. In: *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*. 2019, pp. 203–225.

[80] Shafi Goldwasser, Silvio Micali and Charles Rackoff. 'The knowledge complexity of interactive proof-systems'. In: *Proceedings of the seventeenth annual ACM symposium on Theory of computing* (1985), pp. 291–304.

[81] Yuan Gong, Yang Yang and Zhen Li. 'Efficient and Privacy-Preserving Data Sharing in Vehicular Cloud Computing'. In: *IEEE Transactions on Vehicular Technology* 68.5 (2019), pp. 4430–4440. DOI: 10.1109/TVT.2019.2902490.

[82] Vipul Goyal et al. 'Attribute-based encryption for fine-grained access control of encrypted data'. In: *Proceedings of the 13th ACM conference on Computer and communications security*. 2006, pp. 89–98.

[83] Zhangshuang Guan et al. 'BlockMaze: An efficient privacy-preserving account-model blockchain based on zk-SNARKs'. In: *IEEE Transactions on Dependable and Secure Computing* 19.3 (2020), pp. 1446–1463.

[84] Harry Halpin and Marta Piekarska. 'Introduction to Security and Privacy on the Blockchain'. In: *2017 IEEE European Symposium on Security and Privacy Workshops*. IEEE. 2017, pp. 1–3.

[85] Jinguang Han et al. 'Privacy-preserving decentralized key-policy attribute-based encryption'. In: *IEEE transactions on parallel and distributed systems* 23.11 (2012), pp. 2150–2162.

[86] Muneeb Ul Hassan, Mubashir Husain Rehmani and Jinjun Chen. 'Differential privacy in blockchain technology: A futuristic approach'. In: *Journal of Parallel and Distributed Computing* 145 (2020), pp. 50–74.

[87] Ryan Henry, Amir Herzberg and Aniket Kate. 'Blockchain access privacy: Challenges and directions'. In: *IEEE Security & Privacy* 16.4 (2018), pp. 38–45.

[88] Daira Hopwood et al. 'Zcash protocol specification'. In: *GitHub: San Francisco, CA, USA* (2016), p. 1.

[89] Haojun Huang et al. *Blockchains for Network Security: Principles, technologies and applications*. Institution of Engineering and Technology, 2020.

[90] Junbeom Hur. 'Improving security and efficiency in attribute-based data sharing'. In: *IEEE transactions on knowledge and data engineering* 25.10 (2011), pp. 2271–2282.

[91] Jung Hur and Dong Kun Noh. 'Attribute-Based Access Control with Efficient Revocation in Data Outsourcing Systems'. In: *IEEE Transactions on Parallel and Distributed Systems* 22.7 (2011), pp. 1214–1221. DOI: 10.1109/TPDS.2010.186.

[92] Deloitte Insights. *2021 Global Blockchain Survey: A New Age of Digital Assets*. Deloitte, 2021.

[93] IT Governance. *How much does GDPR compliance cost in 2020?* 2020.

[94] Shivani Jamwal et al. 'A survey on ethereum pseudonymity: Techniques, challenges, and future directions'. In: *Journal of Network and Computer Applications* (2024), p. 104019.

[95] Laraib Javed et al. 'ShareChain: Blockchain-enabled model for sharing patient data using federated learning and differential privacy'. In: *Expert Systems* 40.5 (2023), e13131.

[96] Jayapriya Jayabalan and N Jeyanthi. 'Scalable blockchain model using off-chain IPFS storage for healthcare data security and privacy'. In: *Journal of Parallel and distributed computing* 164 (2022), pp. 152–167.

[97] Bin Jia et al. 'Blockchain-enabled federated learning data protection aggregation scheme with differential privacy and homomorphic encryption in IIoT'. In: *IEEE Transactions on Industrial Informatics* 18.6 (2021), pp. 4049–4058.

[98] Yu Jiang, Xiaolong Xu and Fu Xiao. 'Attribute-based encryption with blockchain protection scheme for electronic health records'. In: *IEEE Transactions on Network and Service Management* 19.4 (2022), pp. 3884–3895.

[99] Don Johnson, Alfred Menezes and Scott Vanstone. 'The elliptic curve digital signature algorithm (ECDSA)'. In: *International journal of information security* 1.1 (2001), pp. 36–63.

[100] Manpreet Kaur et al. 'Ipfs: An off-chain storage solution for blockchain'. In: *Proceedings of International Conference on Recent Innovations in Computing: ICRIC 2022, Volume 1*. Springer. 2023, pp. 513–525.

[101] Krishnaram Kenthapadi, Ilya Mironov and Abhradeep Guha Thakurta. 'Privacy-preserving data mining in industry'. In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 2019, pp. 840–841.

[102] Fawad Khan et al. 'Granular data access control with a patient-centric policy update for healthcare'. In: *Sensors* 21.10 (2021), p. 3556.

[103] Shafaq Naheed Khan et al. 'Blockchain smart contracts: Applications, challenges, and future trends'. In: *Peer-to-peer Networking and Applications* 14 (2021), pp. 2901–2925.

[104] Sunny King and Scott Nadal. *PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake*. 2012.

[105] Steve Klabnik and Carol Nichols. *The Rust Programming Language*. Available online: `https://doc.rust-lang.org/book/`. No Starch Press, 2023. URL: `https://doc.rust-lang.org/book/`.

[106] Anatoly Konkin and Sergey Zapechnikov. 'Systematization of knowledge: privacy methods and zero knowledge proofs in corporate blockchains'. In: *Journal of Computer Virology and Hacking Techniques* 20.2 (2024), pp. 219–224.

[107] Ahmed Kosba et al. 'Hawk: The blockchain model of cryptography and privacy-preserving smart contracts'. In: *2016 IEEE symposium on security and privacy (SP)*. IEEE. 2016, pp. 839–858.

[108] Randhir Kumar and Rakesh Tripathi. 'Implementation of distributed file storage and access framework using IPFS and blockchain'. In: *2019 Fifth International Conference on Image Information Processing (ICIIP)*. IEEE. 2019, pp. 246–251.

[109] Merve Can Kus and Albert Levi. 'Investigation and Application of Differential Privacy in Bitcoin'. In: *IEEE Access* 10 (2022), pp. 25534–25554. DOI: 10.1109/ACCESS.2022.3151784.

[110] Satpal Singh Kushwaha et al. 'Systematic review of security vulnerabilities in ethereum blockchain smart contract'. In: *IEEE Access* 10 (2022), pp. 6605–6621.

[111] Andrei Lapets et al. 'Role-based ecosystem for the design, development, and deployment of secure multi-party data analytics applications'. In: *2019 IEEE Cybersecurity Development (SecDev)*. IEEE. 2019, pp. 129–140.

[112] Ryan Lavin et al. 'A Survey on the Applications of Zero-Knowledge Proofs'. In: *arXiv preprint arXiv:2408.00243* (2024).

[113] Hongwei Li et al. 'Privacy-Preserving Data Sharing Protocol for Blockchain-Based Mobile Social Networks'. In: *IEEE Access* 7 (2019), pp. 8785–8797. DOI: 10.1109/ACCESS.2018.2889715.

[114] Qian Li et al. 'Attribute-Based Encryption with Privacy Protection and Accountability for Cloud IoT'. In: *IEEE Access* 6 (2018), pp. 27324–27335. DOI: 10.1109/ACCESS.2018.2832218.

[115] Xiaofang Li et al. 'A blockchain privacy protection scheme based on ring signature'. In: *IEEE Access* 8 (2020), pp. 76765–76772.

[116] Laura Lotti. 'Contemporary art, capitalization and the blockchain: On the autonomy and automation of art's value'. In: *Finance and Society* 2.2 (2016), pp. 96–110.

[117] Hai Lu et al. 'Policy-driven Data Sharing over Attribute-Based Encryption supporting Dual Membership'. In: *Journal of Systems and Software* 188 (2022), p. 111271. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2022.111271. URL: https://www.sciencedirect.com/science/article/pii/S0164121222000346.

[118] Loi Luu et al. 'Making smart contracts smarter'. In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016, pp. 254–269.

[119] Felix Konstantin Maurer, Till Neudecker and Martin Florian. 'Anonymous CoinJoin transactions with arbitrary values'. In: *2017 ieee trustcom/bigdatase/icess*. IEEE. 2017, pp. 522–529.

[120] Frank McSherry and Kunal Talwar. 'Mechanism design via differential privacy'. In: *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*. IEEE. 2007, pp. 94–103.

[121] Sarah Meiklejohn et al. 'A fistful of bitcoins: characterizing payments among men with no names'. In: *Proceedings of the 2013 conference on Internet measurement conference*. 2013, pp. 127–140.

[122] Maria Luisa Merani, Daniele Croce and Ilenia Tinnirello. 'Rings for privacy: an architecture for large scale privacy-preserving data mining'. In: *IEEE Transactions on Parallel and Distributed Systems* 32.6 (2021), pp. 1340–1352.

[123] Rebekah Mercer. *Privacy on the Blockchain: Unique Ring Signatures*. 2016. arXiv: 1612.01188 [cs.CR].

[124] Johnnatan Messias et al. 'Dissecting Bitcoin and Ethereum Transactions: On the Lack of Transaction Contention and Prioritization Transparency in Blockchains'. In: *arXiv preprint arXiv:2302.06962* (2023).

[125] Meta. *React Documentation*. https://reactjs.org/docs/getting-started.html. 2023.

[126] Microsoft. *Advancing Privacy with Zero-Knowledge Proof Credentials*. Accessed: 2024-10-01. 2020. URL: https://techcommunity.microsoft.com/t5/security-compliance-and-identity/advancing-privacy-with-zero-knowledge-proof-credentials/ba-p/1441554.

[127] Microsoft. *TypeScript: JavaScript With Syntax For Types*. https://www.typescriptlang.org/. 2023.

[128] Moralis. *Breaking Down ETH 2.0 - zk-SNARKS and zk-Rollups*. https://academy.moralis.io/down-eth-2-0-zk-snarks-and-zk-rollups. Accessed: 2023-12-01.

[129] Sascha Müller, Stefan Katzenbeisser and Claudia Eckert. 'Distributed attribute-based encryption'. In: *Information Security and Cryptology–ICISC 2008: 11th International Conference, Seoul, Korea, December 3-5, 2008, Revised Selected Papers 11*. Springer. 2009, pp. 20–36.

[130] Michael Naehrig, Kristin Lauter and Vinod Vaikuntanathan. 'Can homomorphic encryption be practical?' In: *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*. 2011, pp. 113–124.

[131] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008.

[132] T. Naruse, M. Mohri and Y. Shiraishi. 'Provably secure attribute-based encryption with attribute revocation and grant function using proxy re-encryption and attribute key for updating'. In: *Human-Centric Computing and Information Sciences* 5 (1 2015). DOI: 10.1186/s13673-015-0027-0.

[133] Cong T. Nguyen et al. 'Proof-of-Stake Consensus Mechanisms for Future Blockchain Networks: Fundamentals, Applications and Opportunities'. In: *IEEE Access* 7 (2019), pp. 85727–85745. DOI: 10.1109/ACCESS.2019.2925010.

[134] William Nikolakis, Lijo John and Harish Krishnan. 'How blockchain can shape sustainable global value chains: An evidence, verifiability, and enforceability (EVE) framework'. In: *Sustainability* 10.11 (2018), p. 3926.

[135] Shen Noether, Adam Mackenzie et al. 'Ring confidential transactions'. In: *Ledger* 1 (2016), pp. 1–18.

[136] Ilhaam A Omar et al. 'Ensuring protocol compliance and data transparency in clinical trials using Blockchain smart contracts'. In: *BMC Medical Research Methodology* 20 (2020), pp. 1–17.

[137] OpenZeppelin. *OpenZeppelin Documentation: ERC1155*. 2024. URL: https://docs.openzeppelin.com/contracts/3.x/erc1155.

[138] OpenZeppelin. *OpenZeppelin Documentation: ERC20*. 2024. URL: https://docs.openzeppelin.com/contracts/3.x/erc20.

[139] OpenZeppelin. *OpenZeppelin Documentation: ERC721*. 2024. URL: https://docs.openzeppelin.com/contracts/3.x/erc721.

[140] Aafaf Ouaddah, Anas Abou Elkalam and Abdellah Ait Ouahman. 'Towards a Novel Privacy-Preserving Access Control Model Based on Blockchain Technology in IoT'. In: *Europe and MENA Cooperation Advances in Information and Communication Technologies*. Ed. by Álvaro Rocha, Mohammed Serrhini and Carlos Felgueiras. Cham: Springer International Publishing, 2017, pp. 523–533.

[141] Andreea-Elena Panait and Ruxandra F Olimid. 'On using zk-SNARKs and zk-STARKs in blockchain-based identity management'. In: *Innovative Security Solutions for Information Technology and Communications: 13th International Conference, SecITC 2020, Bucharest, Romania, November 19–20, 2020, Revised Selected Papers 13*. Springer. 2021, pp. 130–145.

[142] Young-Hoon Park, Yejin Kim and Junho Shim. 'Blockchain-Based Privacy-Preserving System for Genomic Data Management Using Local Differential Privacy'. In: *Electronics* 10.23 (2021). ISSN: 2079-9292. DOI: 10.3390/electronics10233019. URL: https://www.mdpi.com/2079-9292/10/23/3019.

[143] Bryan Parno et al. 'Pinocchio: Nearly Practical Verifiable Computation'. In: *2013 IEEE Symposium on Security and Privacy*. IEEE, 2013, pp. 238–252. DOI: 10.1109/SP.2013.47.

[144] Juha Partala, Tri Hong Nguyen and Susanna Pirttikangas. 'Non-interactive zero-knowledge for blockchain: A survey'. In: *IEEE Access* 8 (2020), pp. 227945–227961.

[145] Alexey Pertsev, Roman Semenov and Roman Storm. 'Tornado Cash Privacy Solution Version 1.4'. In: *Tornado cash privacy solution version* 1 (2019).

[146] Pinata. *Pinata Documentation: Quickstart Guide*. 2024. URL: https://docs.pinata.cloud/quickstart.

[147] Alexandre Miranda Pinto. 'An introduction to the use of zk-SNARKs in blockchains'. In: *Mathematical Research for Blockchain Economy*. Springer, 2020, pp. 233–249.

[148] Eugenia Politou et al. 'Blockchain mutability: Challenges and proposed solutions'. In: *IEEE Transactions on Emerging Topics in Computing* 9.4 (2019), pp. 1972–1986.

[149] Yogachandran Rahulamathavan et al. 'Privacy-preserving blockchain based IoT ecosystem using attribute-based encryption'. In: *2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*. IEEE. 2017, pp. 1–6.

[150] Anuj Ramachandran and Murat Kantarcioglu. 'Using Blockchain and Smart Contracts for Secure Data Provenance Management'. In: *arXiv preprint arXiv:1709.10000* (2017). URL: https://arxiv.org/abs/1709.10000.

[151] Ronald L Rivest, Adi Shamir and Yael Tauman. 'How to leak a secret: Theory and applications of ring signatures'. In: *Theoretical Computer Science: Essays in Memory of Shimon Even* (2006), pp. 164–186.

[152] O. Ruan and Y. Hu. 'Cp-abe access control scheme supporting data permission management in iot'. In: *Sixth International Conference on Computer Information Science and Application Technology (CISAT 2023)* (2023). DOI: 10.1117/12.3003847.

[153] Rust and WebAssembly Working Group. *wasm-bindgen*. https://rustwasm.github.io/wasm-bindgen/. 2023.

[154] Rust and WebAssembly Working Group. *wasm-pack: Your favorite Rust Wasm workflow tool!* https://rustwasm.github.io/wasm-pack/. 2023.

[155] Haitz Sáez de Ocáriz Borde. 'An Overview of Trees in Blockchain Technology: Merkle Trees and Merkle Patricia Tries'. In: *Department of Engineering, University of Cambridge* (Feb. 2022).

[156] Amit Sahai and Brent Waters. 'Fuzzy identity-based encryption'. In: *Annual international conference on the theory and applications of cryptographic techniques*. Springer. 2005, pp. 457–473.

[157] Amit Sahai, Brent Waters and Steve Lu. 'Attribute-based encryption'. In: *Identity-Based Cryptography*. IOS Press, 2009, pp. 156–168.

[158] Abylay Satybaldy and Mariusz Nowostawski. 'Review of techniques for privacy-preserving blockchain systems'. In: *Proceedings of the 2nd ACM International Symposium on Blockchain and Secure Critical Infrastructure*. 2020, pp. 1–9.

[159] Amy Schmitz and Colin Rule. 'Online dispute resolution for smart contracts'. In: *J. Disp. Resol.* (2019), p. 103.

[160] Yutao Shi et al. 'A semi-homomorphic privacy computing solution based on SM2 and blockchain'. In: *International Conference on Cryptography, Network Security, and Communication Technology (CNSCT 2023)*. Vol. 12641. SPIE. 2023, pp. 29–38.

[161] Ardeshir Shojaeinasab, Amir Pasha Motamed and Behnam Bahrak. 'Mixing detection on bitcoin transactions using statistical patterns'. In: *IET Blockchain* 3.3 (2023), pp. 136–148.

[162] Shlok Shrivastava. *Keccak256 in Solidity*. 2024. URL: `https://dev.to/shlok2740/keccak256-in-solidity-433m`.

[163] Kalpana Singh, Nicolas Heulot and Elyes Ben Hamida. 'Towards anonymous, unlinkable, and confidential transactions in blockchain'. In: *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE. 2018, pp. 1642–1649.

[164] Hiralal Bhaskar Solunke, Pawan Bhaladhare and Amol Potgantwar. 'Privacy Preservation Using Blockchain and Machine Learning: A Review'. In: *Ensuring Security and End-to-End Visibility Through Blockchain and Digital Twins* (2024), pp. 299–325.

[165] V Suma. 'Security and privacy mechanism using blockchain'. In: *Journal of Ubiquitous Computing and Communication Technologies (UCCT)* 1.01 (2019), pp. 45–54.

[166] Shi-Feng Sun et al. 'Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero'. In: *European Symposium on Research in Computer Security*. Springer. 2017, pp. 456–474.

[167] Xiaoqiang Sun et al. 'A survey on zero-knowledge proof in blockchain'. In: *IEEE network* 35.4 (2021), pp. 198–205.

[168] Yuqing Sun et al. 'Privacy-Preserving Multi-Authority Attribute-Based Encryption with Revocation for Big Data Sharing'. In: *IEEE Transactions on Computers* 67.9 (2018), pp. 2444–2457. DOI: `10.1109/TC.2018.2817568`.

[169] Ali Sunyaev and Christian Zirpins. 'Peer-to-Peer Data Networks and the Inter-Planetary File System'. In: *Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies*. Springer, 2024, pp. 273–316.

[170] Supabase. *Supabase Documentation*. 2024. URL: https://supabase.com/docs.

[171] Nick Szabo. 'Formalizing and securing relationships on public networks'. In: *First monday* (1997).

[172] Editorial Team. *Republic protocol (REN) review: Complete Beginners Guide to ren*. 2024. URL: https://www.coinbureau.com/review/republic-protocol-ren/.

[173] Foundry Team. *Anvil Documentation: Foundry Reference*. 2023. URL: https://book.getfoundry.sh/reference/anvil/.

[174] MetaMask Team. *MetaMask Documentation*. 2023. URL: https://metamask.io/.

[175] MUI Team. *MUI: Material-UI for React Documentation*. 2023. URL: https://mui.com/.

[176] RISC Zero Team. *RISC Zero Documentation: Remote Proving*. 2023. URL: https://dev.risczero.com/api/generating-proofs/remote-proving.

[177] Solidity Team. *Solidity Documentation: Introduction to Smart Contracts*. Version 0.8.27. 2023. URL: https://docs.soliditylang.org/en/v0.8.27/introduction-to-smart-contracts.html.

[178] Feng Tian. 'An Agri-Food Supply Chain Traceability System for China Based on RFID & Blockchain Technology'. In: *2016 13th International Conference on Service Systems and Service Management (ICSSSM)*. IEEE, 2016, pp. 1–6. DOI: 10.1109/ICSSSM.2016.7538424.

[179] Devharsh Trivedi. 'Towards Efficient Security Analytics'. PhD thesis. Stevens Institute of Technology, 2024.

[180] Diego Valdeolmillos et al. 'Blockchain technology: a review of the current challenges of cryptocurrency'. In: *Blockchain and Applications: International Congress*. Springer. 2020, pp. 153–160.

[181] Nicolas Van Saberhagen. *CryptoNote v2.0*. 2013. URL: https://cryptonote.org/whitepaper.pdf.

[182] Helen Mary Varghese, Dhwani Apurva Nagoree, N Jayapandian et al. 'Cryptocurrency Security and Privacy Issues: A Research Perspective'. In: *2021 6th International Conference on Communication and Electronics Systems (ICCES)*. IEEE. 2021, pp. 902–907.

[183] Vercel. *Next.js Documentation*. `https://nextjs.org/docs`. 2023.

[184] Hoang Tam Vo, Ashish Kundu and Mukesh K Mohania. 'Research Directions in Blockchain Data Management and Analytics.' In: *EDBT*. 2018, pp. 445–448.

[185] Bo Wang et al. 'PPFLHE: A privacy-preserving federated learning scheme with homomorphic encryption for healthcare data'. In: *Applied Soft Computing* 146 (2023), p. 110677.

[186] Guojun Wang, Qin Liu and Jie Wu. 'Hierarchical attribute-based encryption for fine-grained access control in cloud storage services'. In: *Proceedings of the 17th ACM conference on Computer and communications security*. 2010, pp. 735–737.

[187] Haisong Wang, Yang Song and Qing Zhang. 'Medical Data Sharing Model Based on Blockchain'. In: *2018 IEEE International Symposium on Computer, Consumer and Control (IS3C)*. IEEE, 2018, pp. 388–392. DOI: `10.1109/IS3C.2018.00105`.

[188] Shulan Wang et al. 'A fast CP-ABE system for cyber-physical security and privacy in mobile healthcare network'. In: *IEEE Transactions on Industry Applications* 56.4 (2020), pp. 4467–4477.

[189] Wenting Wang et al. 'A Survey on Consensus Mechanisms and Mining Strategy Management in Blockchain Networks'. In: *IEEE Access* 7 (2019), pp. 22328–22370. DOI: `10.1109/ACCESS.2019.2896108`.

[190] Shawn Wilkinson, Jim Lowry and Tome Boshevski. 'Metadisk a blockchain-based decentralized file storage application'. In: *Storj Labs Inc., Technical Report, hal* 1.11 (2014).

[191] Maximilian Wohrer and Uwe Zdun. 'Smart contracts: security patterns in the ethereum ecosystem and solidity'. In: *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*. IEEE. 2018, pp. 2–8.

[192] Axin Wu et al. 'Efficient and privacy-preserving traceable attribute-based encryption in blockchain'. In: *Annals of Telecommunications* 74 (2019), pp. 401–411.

[193] Huixin Wu and Feng Wang. 'A survey of noninteractive zero knowledge proof system and its applications'. In: *The scientific world journal* 2014.1 (2014), p. 560484.

[194] Jiajing Wu et al. 'Detecting mixing services via mining bitcoin transaction network with hybrid motifs'. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 52.4 (2021), pp. 2237–2249.

[195] Lei Wu et al. 'Towards understanding and demystifying bitcoin mixing services'. In: *Proceedings of the Web Conference 2021*. 2021, pp. 33–44.

[196] Jianan Xu et al. 'BlendCAC: A Blockchain-Enabled Decentralized Capability-Based Access Control Mechanism for IoT'. In: *IEEE Internet of Things Journal* 6.2 (2019), pp. 2119–2130. DOI: 10.1109/JIOT.2018.2874058.

[197] Zhigang Xu et al. 'BMTAC: a decentralized, auditable, time-limited, multi-authority attribute access control scheme in blockchain environment'. In: *2022 IEEE Smartworld, Ubiquitous Intelligence & Computing, Scalable Computing & Communications, Digital Twin, Privacy Computing, Metaverse, Autonomous & Trusted Vehicles (SmartWorld/UIC/ScalCom/DigitalTwin/PriComp/Meta)*. IEEE. 2022, pp. 1997–2002.

[198] Liang Xue et al. 'Efficient attribute-based encryption with attribute revocation for assured data deletion'. In: *Information Sciences* 479 (2019), pp. 640–650.

[199] Anatoly Yakovenko. 'Solana: A new architecture for a high performance blockchain v0. 8.13'. In: *Whitepaper* (2018).

[200] Yuping Yan et al. *HE-DKSAP: Privacy-Preserving Stealth Address Protocol via Additively Homomorphic Encryption*. 2023. arXiv: 2312.10698 [cs.CR].

[201] Kan Yang and Xiaohua Jia. 'Attributed-based access control for multi-authority systems in cloud storage'. In: *2012 IEEE 32nd International Conference on Distributed Computing Systems*. IEEE. 2012, pp. 536–545.

[202] Andrew C Yao. 'Protocols for secure computations'. In: *23rd annual symposium on foundations of computer science (sfcs 1982)*. IEEE. 1982, pp. 160–164.

[203] Congcong Ye et al. 'Analysis of security in blockchain: Case study in 51%-attack detecting'. In: *2018 5th International conference on dependable systems and their applications (DSA)*. IEEE. 2018, pp. 15–24.

[204] RISC Zero. *RISC Zero Foundry Template*. https://github.com/risc0/risc0-foundry-template. 2024.

[205] RISC Zero. *RISC Zero Terminology: Image ID*. Accessed: 2024-10-01. 2024. URL: https://dev.risczero.com/terminology#image-id.

[206] RISC Zero. *RISC Zero Terminology: Seal*. Accessed: 2024-10-01. 2024. URL: https://dev.risczero.com/terminology#seal.

[207] RISC Zero. *RISC Zero zkVM API Documentation*. Accessed: 2024-10-01. 2024. URL: https://dev.risczero.com/api/zkvm.

[208] Jianhong Zhang, Wenle Bai and Zhengtao Jiang. 'On the Security of a Practical Constant-Size Ring Signature Scheme.' In: *Int. J. Netw. Secur.* 22.3 (2020), pp. 392–396.

[209] Rui Zhang, Rui Xue and Ling Liu. 'Security and privacy on blockchain'. In: *ACM Computing Surveys (CSUR)* 52.3 (2019), pp. 1–34.

[210] Yachao Zhang et al. 'Efficient Attribute-Based Access Control with Attribute Revocation for Outsourced Data Sharing in Cloud Computing'. In: *Proceedings of the 2013 International Conference on Cloud Computing and Big Data (CLOUDCOM-ASIA '13)*. IEEE, 2013, pp. 437–442. DOI: 10.1109/CLOUDCOM-ASIA.2013.48.

[211] Yinghui Zhang et al. 'Attribute-based encryption for cloud computing access control: A survey'. In: *ACM Computing Surveys (CSUR)* 53.4 (2020), pp. 1–41.

[212] Chuan Zhao et al. 'Secure multi-party computation: theory, practice and applications'. In: *Information Sciences* 476 (2019), pp. 357–372.

[213] Jinyi Zhao et al. 'ZK-CPABE: blockchain-based CP-ABE scheme via zero-knowledge proof'. In: *Seventh International Conference on Advanced Electronic Materials, Computers, and Software Engineering (AEMCSE 2024)*. Vol. 13229. SPIE. 2024, pp. 867–872.

[214] Zibin Zheng et al. 'An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends'. In: *2017 IEEE International Congress on Big Data (BigData Congress)*. IEEE, 2017, pp. 557–564. DOI: 10.1109/BigDataCongress.2017.85.

[215] Hanrui Zhong et al. 'Secure multi-party computation on blockchain: An overview'. In: *Parallel Architectures, Algorithms and Programming: 10th International Symposium, PAAP 2019, Guangzhou, China, December 12–14, 2019, Revised Selected Papers 10*. Springer. 2020, pp. 452–460.

[216] Jianying Zhou and Kwok-Yan Lam. 'Securing digital signatures for non-repudiation'. In: *Computer Communications* 22.8 (1999), pp. 710–716.

[217] Lu Zhou et al. 'Leveraging zero knowledge proofs for blockchain-based identity sharing: A survey of advancements, challenges and opportunities'. In: *Journal of Information Security and Applications* 80 (2024), p. 103678.

[218] Guy Zyskind, Oz Nathan and Alex Pentland. 'Decentralizing privacy: Using blockchain to protect personal data'. In: *2015 IEEE Security and Privacy Workshops* (2015), pp. 180–184.

[219] Guy Zyskind, Oz Nathan and Alex Pentland. 'Enigma: Decentralized computation platform with guaranteed privacy'. In: *arXiv preprint arXiv:1506.03471* (2015).