Privacy-Preserving and Regulation-Enabled Mechanisms for Blockchain-based Financial Services

by

Liang Xue

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2022

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Kui Ren Professor School of Computer Science and Technology Zhejiang University
Supervisor(s):	Xuemin (Sherman) Shen Professor Department of Electrical and Computer Engineering University of Waterloo
	Xiaodong Lin Professor School of Computer Science University of Guelph
Internal Member:	Sagar Naik Professor Department of Electrical and Computer Engineering University of Waterloo
Internal Member:	Vijay Ganesh Associate Professor Department of Electrical and Computer Engineering University of Waterloo
Internal-External Member:	Jun Liu Associate Professor Department of Applied Mathematics University of Waterloo

Author's Declaration

Abstract

With the success of cryptocurrencies such as Bitcoin, blockchain technology has attracted extensive attention from both academia and industry. As a distributed ledger technology, blockchain provides decentralization and immutability, and can build trust among multiple parties. Owning to these unique characteristics, blockchain has become an innovative approach to secure and reliable record-keeping and transaction execution, and has the potential to revolutionize the financial industry and drive economic change on a global scale. For example, it can streamline banking and lending services, enable decentralized trading, and facilitate cross-border payment transactions. Although blockchain is expected to create a new paradigm for the financial industry, transactions stored on the blockchain are shared among the nodes in the blockchain network, which may contain sensitive information of users, such as the identities of senders and receivers, and the contents of transactions. Thus, privacy preservation should be achieved when applying blockchain to different financial services. Many privacy-preserving mechanisms have been proposed to guarantee identity privacy and data confidentiality for blockchain-based transactions. However, the strong degree of privacy may create new regulatory concerns. First, in privacy-preserving mortgage lending, there exists double-mortgage fraud, by which a borrower can use the same asset as collateral to obtain multiple loans from different financial institutions. Second, in decentralized data trading, data buyers may refuse to pay funds to data sellers after obtaining data, and data sellers may send fake data to data buyers. Verifying data availability and retrievability without viewing data before payment for fair trading is a challenging issue. Moreover, the identity privacy of data sellers should be preserved during the trading. Third, in privacy-preserving blockchain-based payment systems, the identities of the payer, payee, and transferred amount are protected. Nevertheless, the anonymity of transactions can be exploited for illegal activities, such as money laundering. Thus, considering the strict regulatory requirements of the financial industry, such as limiting the amount of cryptocurrency transferred over a period of time, privacy preservation and regulation should be balanced in blockchain-based financial services.

In this thesis, we focus on three major blockchain-based financial services to concentrate on how to solve the dilemma between privacy protection and strict regulatory requirements at various phases in the fund flow, which are lending, trading, and payment. Firstly, the thesis investigates the borrower privacy and double-mortgage regulation issues in mortgage lending, and proposes a blockchain-based privacy-preserving and accountable mortgage data management scheme. In the scheme, the mortgage data of borrowers can be shared on the blockchain to detect the double-mortgage fraud without revealing the identity of borrowers. But financial institutions can still uncover the identity of a dishonest borrower if he/she pledges the same asset for multiple mortgages, which is achieved by integrating cryptographic tools such as verifiable secret sharing, zero-knowledge proof, and ElGamal encryption. A mortgage request contains a share of identity information of the borrower and the ownership certificate of an asset. By utilizing ElGamal encryption and verifiable secret sharing, the borrower can prove that its identity information is indeed included in the mortgage request and can be used to reconstruct its identity when double-mortgage behavior is detected. Secondly, the thesis investigates the identity privacy and trading-misbehavior regulation in blockchain-based data trading. Blockchain can build trust between data buyers and data sellers. To resolve the fairness issue of demonstrating data availability and retrievability without leaking data while preserving identity privacy of data sellers, we propose a blockchain-based fair data trading protocol with privacy preservation, where a data buyer can declare data requirements and acceptable issuers of data, and a data seller can conduct privacy-preserving and fine-grained data selling. We first define the fairness and privacy demands for both parties. By incorporating anonymous attribute-based credentials, structure-preserving signatures, and zero-knowledge proofs, data can be traded in part while data authenticity is guaranteed and data issuers are hidden. A smart contract is utilized to realize atomic transactions. Security proof is provided to demonstrate that the scheme can achieve privacy preservation and fairness for the participants. Thirdly, the thesis investigates the transaction privacy and anti-money laundering regulation issues in distributed anonymous payment (DAP) systems. To solve the conflict between privacy and regulation, we propose a novel DAP scheme that supports regulatory compliance and enforcement. We first introduce regulators into the system, who define regulatory policies, including limiting the total amount of cryptocurrency one can transfer and the frequency of transactions one can conduct in a time period. The policies are enforced through commitments and non-interactive zero-knowledge proofs for compostable statements. By this, users can prove that transactions are valid and comply with regulations. We use both Zero-knowledge Succinct Non-Interactive Arguments of Knowledge (Zk-SNARKs) and sigma protocols to generate the zero-knowledge proofs for regulation compliance. A tracing mechanism is designed in the scheme to allow regulators to recover the real identities of users when suspicious transactions are detected.

In summary, this thesis proposes effective privacy-preserving and regulation-enabled solutions for blockchain-based lending, data trading, and anonymous payment. The results from the thesis should shed light for future study on blockchain-based systems where privacy preservation and regulation are required.

Acknowledgements

First, I would like to express my heartfelt gratitude to my supervisor, Professor Xuemin (Sherman) Shen, not only for his patient guidance and invaluable advice but also for being a role model in doing research. He encourages us to think more and discuss with others so that we can learn from each other and improve together. His rigorous attitude towards academic research and willingness to help others have also influenced me and are precious treasure in my future career. I would also gratefully acknowledge my co-supervisor, Professor Xiaodong Lin, who provides me with considerable encouragement and generous assistance. He enlightens us to have an in-depth analysis of the research problem and capture the essence. His wide range of knowledge has broadened my view and inspired me with new ideas for my research.

I would like to thank Professor Kui Ren, Professor Sagar Naik, Professor Vijay Ganesh, and Professor Jun Liu for being committee members for my thesis examination. Their valuable questions and comments provide me with deep insights into my research topic and help improve the quality of the thesis.

I wish to express my sincere thanks to all BBCR group members who have helped and taught me immensely during the four years of my Ph.D. study. I have gained much knowledge from our weekly group meetings and spirited discussions. Moreover, with many memorable moments, they have made my life enjoyable and colorful. Special thanks go to Dr. Jianbing Ni, Dr. Ning Zhang, Dr. Nan Cheng, Dr. Qiang Ye, Dr. Wen Wu, Dr. Nan Chen, Dr. Junling Li, Dr. Weisen Shi, Dr. Huaqing Wu, Dr. Jiayin Chen, Dr. Haixia Peng, Dr. Dongxiao Liu, Dr. Cheng Huang, Dr. Kaige Qu, Dr. Mushu Li, Dr. Jiahui Hou, Conghao Zhou, Han Yin, Lingshuang Liu, Professor Yong Yu, Professor Haomiao Yang, Jinwen Liang, and Fuyuan Song, for their inspiring discussions and valuable insights on my research. I would like to thank all BBCR group members for their generous support, helpful advice, and all the good times we spent together.

Also, I would like to thank my course instructors and staff members of the Electrical and Computer Engineering Department, University of Waterloo. Their professional knowledge and enthusiastic support make me enjoy the Ph.D. study period.

Finally, my thanks would go to my beloved parents, my sister, and my brother, for their unconditional love and care over all these past years. Their wholehearted support also helps me fulfill my career goals.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

PoW	Proof of Work
PoS	Proof of Stake
BFT	Byzantine Fault Tolerance
PBFT	Practical Byzantine Fault Tolerance
PoET	Proof of Elapsed Time
TEE	Trusted Execution Environment
FinCEN	Financial Crimes Enforcement Network
MSB	Money Services Business
ECDSA	Elliptic Curve Digital Signature Algorithm
ZKP	Zero-Knowledge Proof
NIZKP	Non-Interactive Zero-Knowledge Proof
ERC	Ethereum Request for Comments
DeFi	Decentralized Finance
ZKCP	Zero-Knowledge Contingent Payment
CP-NIZK	Commit-and-Prove Non-Interactive Zero-knowledge argument of Knowledge
SGX	Software Guard Extensions
DAP	Decentralized Anonymous Payment
TTP	Trusted Third Party

DDH	Decisional Diffie-Hellman
EUF-CMA	Existential Unforgeable under the Chosen Message Attacks
IND-CPA	Indistinguishable under the Chosen Plaintext Attacks
IPFS	InterPlanetary File System
IoT	Internet of Things
EMR	Electronic Medical Record
CBDC	Central Bank Digital Currency
TA	Trust Authority
PRF	Pseudorandom Function
PoA	Proof of Assets
PoL	Proof of Liabilities
SMHT	Sequence-based Merkle Hash Tree
MHT	Merkle Hash Tree
GDPR	General Data Protection Regulation
Zk-SNARK	Zero-knowledge Succinct Non-interactive Arguments of Knowledge
KYC	Know-Your-Customer
AML	Anti-Money Laundering

Chapter 1

Introduction

Blockchain, as one of the enabling technologies for Bitcoin, has attracted considerable attention from both academia and industry. A blockchain can be seen as a public ledger of transactions or digital events that is maintained by the nodes in a peer-to-peer network. By integrating the technologies of distributed consensus mechanism, cryptographic hash, digital signature, and economic incentive mechanism, blockchain owns the features of decentralization, immutability, verifiability, and reliability. Because of the unique characteristics of blockchain, it has been exploited in many scenarios, including cryptocurrencies, healthcare records management, verifiable electronic voting, identity management systems, Internet of things, and supply chain management. Companies such as IBM, Amazon, eBay, Ali, and Samsung, are all exploring novel uses of blockchain for their own applications. According to an IBM survey [1], 91% of banks are investing in blockchain solutions, and 66% of institutions expect to be in production and running at scale with blockchain.

Blockchain can revolutionize a broad range of industries, and the most successful and popular domain of blockchain use is the financial industry. The reasons include that security is of utmost importance for the financial domain, and blockchain's features make it ideal for financial services. To be specific, blockchain facilitates safe, easy transactions and can build trust among trading partners. Hence, no central authority or intermediary is needed. Moreover, blockchain can achieve high levels of transparency, real-time settlement, and a drastic reduction in operational costs. These obvious benefits have made many financial services gain traction in the industry, such as banking [2] and lending [3], data trading [4], and decentralized anonymous payments. According to a study from Markets and Markets [5], the global blockchain market size is expected to grow from 4.9 billion in 2021 to 67.4 billion by 2026, at an impressive compound annual growth rate of 68.4%.

Although blockchain enables many attractive financial services, transactional privacy is one of the most challenging problems when deploying blockchain in different applications. Since the blockchain can be accessed by all the nodes in the network, the sensitive information such as the sender address and receiver address of a transaction and the link relationship among transactions can be obtained by everyone in the network. Thus, the transactions that contain private information should be confidential, and the participants should be anonymous. On the other hand, when the privacy is guaranteed by using a series of cryptographic techniques, the strong privacy creates new regulatory concerns. For example, the anonymity feature of transactions can facilitate the illegal activities, such as online extortion and money laundering, causing the financial markets largely unregulated. Thus, providing reasonable privacy protection while allowing regulation to prevent criminal exploitation is necessary for blockchain-based financial services. In this thesis, we will resolve the contradiction between privacy and regulation in blockchain-based financial services.

1.1 Blockchain and Blockchain-based Financial Services

1.1.1 Blockchain Technology

Blockchain technology was first proposed by Nakamoto [6] in 2008. A blockchain is a distributed ledger maintained by many nodes in a peer-to-peer network. Compared with the conventional centralized storage platform, blockchain can store data in a decentralized way by storing identical copies of data on multiple computers across a network. In a blockchain, all transactions are stored in a chain of blocks, and new blocks can be added linearly and chronologically. Once a block is generated and stored on the blockchain, everyone can verify the validity of the block. An example of blockchain is shown in Figure 1.1. Each block in the blockchain contains a cryptographic hash of the previous block, a timestamp, transaction data, etc. Since the hash of a block is included in its next block, and modifying the transaction of one block makes all the subsequent blocks invalid, it is hard to alter the contents in the blockchain without the collusion of the network majority. The features of blockchain allow many complex applications to be built on top of this disruptive innovation, such as record keeping, data sharing and processing, and identity management. Moreover, blockchain can facilitate interoperability and collaboration among organizations, and brings a potential paradigm shift in how the applications will be built, deployed, and serve their customers.

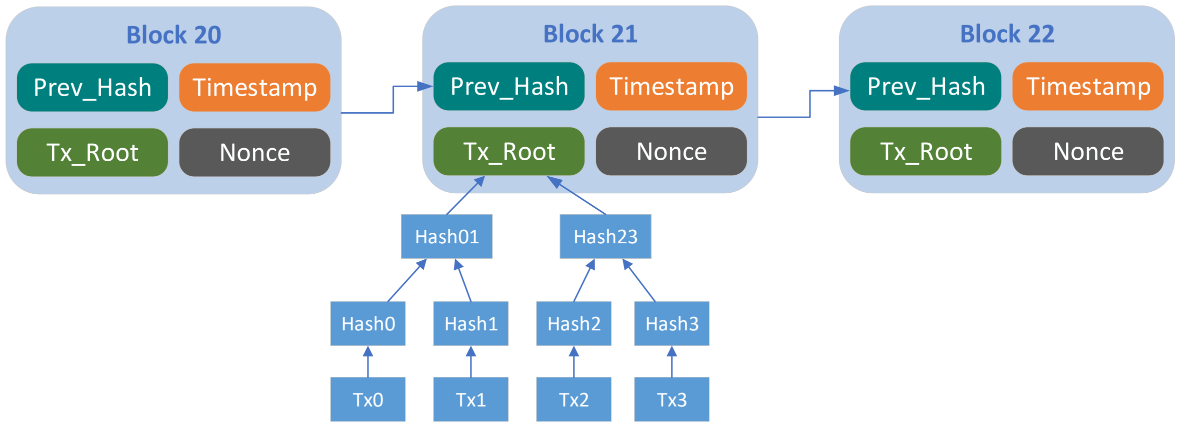


Figure 1.1: An example of blockchain

With the development of blockchain technology, how blockchains are constructed, accessed, and validated is becoming more diverse and flexible. Currently, blockchains can be classified into three categories: public blockchain, private blockchain, and consortium blockchain. For public blockchain, anyone can join the network and participate in the consensus procedure of determining which block is valid and can be added to the blockchain. For private blockchain, the write permissions are strictly controlled by a single organization, even though the access to the blockchain can be open to the public or restricted to a subset of nodes in the network. For consortium blockchain, instead of a single participant who rules the system, it is managed by a pre-selected set of organizations. These organizations can influence and control the consensus process. Consortium blockchain allows cooperation among the organizations and provides a certain degree of security because only approved nodes can join the network.

The consensus protocols employed by different types of blockchains are different. Proof of Work (PoW) [7] and Proof of Stake (PoS) [8] are often used in public blockchain. PoW consensus is characterized by two properties: 1) It is difficult and time-consuming for a node to create a proof that satisfies the requirements defined in the consensus protocol; 2) It is easy for others to verify that the generated block is acceptable. The consensus protocol of Bitcoin is PoW, which limits the rate of generating a valid block by the nodes in the network to roughly 10 minutes. Miners need to find a nonce, which is included in the block header, such that the hash of a block header begins with a number of zeros. The rate control of the block generation is implemented by adjusting the difficulty of the challenge to make sure that it takes about 10 minutes to solve the puzzle. PoS consensus requires the nodes to stake

Table 1.1: Comparison of consensus protocols

Consensus Protocol	Type of Blockchain	Threshold of Fault Tolerance	Efficiency	Power Consumption	Scalability
PoW	Public	50%	Low	Large	High
PoS	Public	50%	Medium	Small	High
PBFT	Consortium/private	33%	High	Negligible	Low
Raft	Consortium/private	50%	High	Negligible	Low
PoET	Consortium/private	TEE failure	High	Negligible	High

a certain number of their tokens so as to have a chance to be selected as the node that can create and validate new blocks. The probability of being selected is proportional to their stake. Byzantine Fault Tolerance (BFT)-based consensus protocols are often employed in consortium blockchain and private blockchain. For BFT-based consensus [9], the agreement can be reached if a majority of nodes are honest. Practical Byzantine Fault Tolerance (PBFT) consensus protocol [10] is the first high-performance consensus protocol and has been widely implemented and applied in Hyperledger Fabric [11]. Raft consensus protocol [12] works by selecting a leader among the nodes, and all the nodes trust the elected leader. Raft can tolerate up to 50% of crashed nodes, which is higher than BFT-based consensus protocols. The fault in Raft is typically counted as network down, node crash, and network delay. However, in the Byzantine system, nodes can be malicious and affect the decision-making process. Proof of Elapsed Time (PoET) consensus protocol [13] achieves fairness and low computing consumption by leveraging Trusted Execution Environment (TEE). We compare the different consensus protocols in terms of the type of blockchain a consensus protocol is suitable for, the threshold of fault tolerance, efficiency, power consumption, and scalability. The results are shown in Table 1.1.

Smart contracts are computer programs stored on a blockchain, and when predetermined conditions are met, they can be automatically executed so that all participants agree with the outcome. Since smart contracts can be written in Turing-complete programming languages, they can perform arbitrary computations and achieve different functionalities, such as voting [14], auction [15], and trading [16].

In a blockchain network, there are mainly two types of roles: clients and validators.

▷ *Clients*: They can send transactions to the blockchain network. Moreover, they can deploy smart contracts on the blockchain and invoke the deployed smart contracts.

▷ *Validators*: They are consensus nodes in the blockchain network and are responsible for packing the received transactions into blocks and finalizing blocks.

As a distributed ledger technique, blockchain has the features of decentralization, immutability, transparency, and programmability. A decentralized consensus mechanism enables distrusted entities to finish transactions without the involvement of a trusted third party. By utilizing cryptographic tools such as hash and signature, no one can tamper with a transaction after it is recorded on the blockchain. Since newly created blocks are linked to previous blocks, and the ledger is shared by all the nodes in the network, nodes can track the data on the blockchain with full transparency. Smart contracts enable the programmability of the blockchain and expand the application scenarios of the blockchain. For example, it can be applied in finance [17], Internet of things [18], supply chain [19], and so on.

1.1.2 Blockchain-based Financial Services

Since blockchain has the characteristics of decentralization and can build trust among different users, it is suitable for financial services, where multiple parties are involved, and trust is critical for participants. Moreover, without the third-party intermediaries, transactions can be confirmed rapidly and can be tracked in real-time, which improves the user experience and reduce risk in business operations. Furthermore, smart contracts can automate business logic and streamline asset and stakeholder management. There are mainly three blockchain-based financial services that have gained traction in both academia and industry, which are described as follows:

▷ *Lending*: The transparent and immutable ledger enables different parties to collaborate, manage data, and track records [20]. In traditional lending, it takes more than one month for information verification, credit scoring, loan processing, and distribution of funds. Moreover, since there is no information exchange between different financial institutions, the double-mortgage fraud exists, i.e., a dishonest borrower makes a repeated mortgage to acquire more money from different lenders, which causes the lenders to be exposed to huge financial risks. Blockchain can be a shared ledger that is maintained by different financial institutions, which can add mortgage information to the chain and prevent the double-mortgage fraud. Meanwhile, blockchain facilitates collateralization of assets as it enables real-time asset management, data tracking, and enforcement of regulatory controls.

▷ *Data Trading*: Fair data trading between data buyers and data sellers is hard to implement due to the mistrust between them. In traditional data markets, a trusted data trading platform is involved in facilitating transactions. However, it may try to steal and resell the data of data sellers and is vulnerable to the single point of failure attack. Blockchain can replace the third party and achieve fair data trading in a trust-less manner by exploiting its decentralization and programmability [21]. Smart contracts can be utilized to guarantee the atomicity of transactions. The process of blockchain-based data trading is as follows: A data buyer can publish the data requirements and deposit the data rewards on the blockchain. If a data seller has data that satisfy the data requirements, it can send the encrypted data off-chain to the data buyer and demonstrate the data availability to the data buyer. After that, the data seller uploads the corresponding key material to the blockchain. Once the key material is verified, the smart contract transfers the rewards to the data seller. Blockchain-based data trading enables users to trade without intermediaries and avoid the risks caused by centralized trading platforms.

▷ *Cryptocurrency*: As a typical application of blockchain, cryptocurrencies enable users to transfer funds in an efficient way without the involvement of a trusted third party [22]. Since anyone can access a public blockchain network, geographic location has less impact on transaction speed. With low cost and short delay, cryptocurrencies are promising in cross-border payments. By utilizing public key cryptography and the transparency of the public ledger, it can be guaranteed that payers can only transfer their cryptocurrency to a recipient and the cryptocurrency cannot be double spent. Thus, blockchain provides higher security and efficiency for payments between individuals.

1.2 Privacy and Regulatory Requirements in Blockchain-based Financial Services

Financial services can involve a large amount of personal information, such as one's identity, account, and asset information, which users are not reluctant to disclose to others. If the information is stored on a blockchain, it will cause privacy risks to users and impede the development of blockchain-based financial services. Moreover, since finance services are bound up with funds and assets, no supervision can disrupt a country's financial order. Currently, many countries have enacted financial regulations to prevent money laundering activities and inhibit related criminal offenses. In blockchain-based financial services, we should take both privacy and regulation into consideration. In the following, we will analyze the privacy and regulation requirements for the above-mentioned blockchain-based financial services.

1.2.1 Privacy Requirements

▷ *Lending*: In blockchain-based asset mortgage management, mortgage data are uploaded to the chain [23]. However, it is desired that the privacy information of a borrower, such as its identity and mortgaged asset, is not stored on the blockchain since there are many financial institutions in the blockchain network, and the business data of a financial institution should not be leaked to other financial institutions. Moreover, the identity information of a borrower can also be concealed from the lending institution, as long as the institution can recover the identity of the borrower if it double-mortgages its asset.

▷ *Data trading*: In blockchain-based data trading, data requirements and rewards are published on the blockchain. The identities of data sellers should be protected, and trading transactions of the same seller cannot be linked [24]. Otherwise, one can infer the identity of a data seller through the types of data it possesses. Furthermore, not only others cannot obtain the identities of data sellers from the blockchain, but also a data seller should be able to delete the personal information in the file without affecting the availability verification of the file. That is, the data buyer can verify the file is an authentic file of the data seller and that it satisfies the data requirements of the data buyer. To guarantee that the file is not leaked to the data buyer before the payment, the file should be encrypted by the data seller and sent to the data buyer in the ciphertext. Based on the received ciphertext, the data buyer should be able to verify the data availability.

▷ *Cryptocurrencies*: Cryptocurrencies allow a user to transfer funds to another user. The transparency and decentralization of blockchain make that all transactions on the blockchain can be accessed by each node in the distributed network. To preserve the privacy of the payer and the payee in a transaction, their identities, the transferred amounts, and account balances should be concealed from other users in the blockchain network. Moreover, the transactions of the same user should not be able to be linked. Otherwise, the flow of one's cryptocurrencies can be tracked. To guarantee that transactions can be completed and recorded on the blockchain, the consensus nodes should be able to verify the validity of transactions, i.e., the cryptocurrency belongs to the payer, the payer's current balance is greater than the amount of the transferred cryptocurrency, and the cryptocurrency is not being spent twice. Thus, privacy preservation should not impact the transaction verification.

1.2.2 Regulation Requirements

- ▷ *Lending*: A financial institution should authenticate the identity of a borrower and verify that the user indeed owns the asset to be pledged. Moreover, when there is double-mortgage fraud, a financial institution should be able to detect it and prevent the transaction. Also, the institution can recover the identity of the malicious borrower and add it to a blacklist.

- ▷ *Data trading*: If the data buyer or the data seller acts maliciously, the trading transaction should not succeed. The malicious behavior of a data buyer includes not paying for the data after receiving the data from a data seller and reselling the data to others. On the other hand, the malicious behavior of a data seller includes not sending the data to the data buyer after obtaining the data rewards. In blockchain-based data trading, the fairness of the trading should be guaranteed without a trusted third party. Moreover, data belong to the data owner, and a data buyer should not be able to resell the purchased data on the blockchain.

- ▷ *Cryptocurrencies*: Since users' identities are protected and cryptocurrencies are not regulated by any country, anonymous cryptocurrencies can be used for illegal activities, such as money laundering and terrorist financing. Many jurisdictions have released or are exploring regulations for cryptocurrencies [25,26]. In the US, Financial Crimes Enforcement Network (FinCEN) issued a guidance document that extends Money Services Business (MSB) of the US Bank Secrecy Act to cryptocurrencies, in which companies working with cryptocurrencies should comply with Anti-Money Laundering (AML) and Know-Your-Customer (KYC) regulations. Moreover, for the third-party service providers, such as exchanges, payment services providers, and wallet providers, licensing regimes [27] are introduced to protect customers and combat money laundering. Canada has also enacted the Proceeds of Crime and Terrorist Financing Act, and transactions over CA $10,000 should be reported. Thus, cryptocurrencies should enforce regulatory policies while preserving users' privacy.

1.3 Research Challenges and Objectives

The objective of this thesis is to develop a set of privacy-preserving and regulation-compliant schemes for blockchain-based financial services. Since financial services involve sensitive information of users while ensuring the functionality of services, we also consider the identity privacy and content confidentiality of transactions, which should not be leaked to other nodes in the blockchain network. Moreover, privacy preservation should not affect the verification of regulatory compliance. To balance the privacy and regulation requirements in different financial services, the following challenges should be addressed:

▷ *Lending*: In blockchain-based mortgage data management, financial institutions can share the mortgage asset information to prevent a borrower from using the same asset as collateral to obtain multiple loans from different financial institutions, which is also called double-mortgage fraud. Simply putting the records of mortgages on the blockchain and comparing them for double-mortgage detection does not solve the problem since financial institutions may not be willing to publish their data in a public manner. How to guarantee the identity privacy and asset privacy of borrowers in data sharing while a financial institution can verify the ownership of assets and detect the fraudulent behavior of borrowers is a challenging issue. Moreover, when the double-mortgage fraud is detected, the anonymity prevents the misbehaving borrower's true identity from being revealed. How to reveal the true identity of the malicious borrower when fraudulent behavior is detected while the privacy of honest borrowers is preserved is also a challenging issue.

▷ *Data Trading*: For blockchain-based fair data trading, the identities of data buyers and sellers should be protected as well as the trading data. However, to guarantee the fairness of data trading, data buyers need to verify the availability of data. That is, a data seller's data are authentic and satisfy the requirements of a data buyer. How to verify the data availability without viewing the plaintext data is a challenging issue [28]. Note that if the check is based on plaintext, the data buyer can refuse to pay for the data even if the data is indeed what it needs as it has already obtained the data. Moreover, when the data availability is verified based on the ciphertext of data, the data seller needs to upload the corresponding key material to a smart contract. If the key is correct, the reward is transferred to the data seller. Thus, another challenge is how to guarantee data retrievability, i.e., the key published on the blockchain can decrypt the ciphertext of data. Furthermore, to prevent the data buyer from obtaining the personal information included in the trading file, the data seller can delete the sensitive information before sending the ciphertext to the data buyer. However, removing part of the data in a file may affect the authenticity verification of the file. For example, a file is signed by an entity, and one can verify the authenticity of the file by verifying the signature. When the original data are modified, the signature cannot be verified as well as the data authenticity, which impacts the verification of data availability. Thus, the third challenge is how to enable the availability of data even if the data are not intact.

- ▷ *Cryptocurrency*: For cryptocurrency, a transfer transaction contains the payer, payee, and the transferred amount. In Bitcoin, the transferred Bitcoin in a transaction is shown in plaintext, and a pseudonym mechanism is employed to disguise the identities of users. Nonetheless, research shows that Bitcoin only provides pseudonymity, and one can de-anonymize users and trace transactions by utilizing graph analysis and address clustering [29,30]. To improve the privacy of cryptocurrencies, decentralized anonymous payment schemes are proposed, such as Zerocash and Monero, which enable users to pay each other without leaking the payment's origin, destination, and transferred amount, and transactions of a user cannot be linked. However, the strong degree of privacy prevents the regulation of cryptocurrencies. To balance the privacy and regulation for cryptocurrency, the following challenges need to be addressed: Firstly, when a regulator limits a user's total transaction amount and the number of transactions in a time period, the unlinkability of transactions causes the calculation of a user's total transaction amount a hard problem. Secondly, how to verify the validity and regulatory compliance of transactions for validators when the transactions are confidential and the identities of participants are anonymous is a challenging issue. Furthermore, when a transaction violates the regulatory policies, the identity of the identity should be recovered. Thus, the third challenge issue is to design an identity recovery mechanism that can recover the real identities of users only if they conduct illegal or suspicious transactions.

1.4 Research Contributions

To achieve the above objectives, we develop a suite of privacy-preserving schemes for blockchain-based financial services with regulatory compliance. Specifically, the main contributions are summarized as follows:

- ▷ *Accountable and privacy-preserving lending*: We construct a blockchain-based accountable mortgage data management scheme. It enables financial institutions to share the mortgage data of borrowers in an efficient and secure manner while ensuring the borrower's identity privacy and accountability. To be specific, borrower identity is concealed on the blockchain by the designed anonymous identity credential, and the extractability of a malicious borrower's identity is achieved by integrating ElGamal encryption, zero-knowledge proof, and verifiable secret sharing. Financial institutions can anonymously authenticate the honest borrowers during the process of mortgage. Only when the double-mortgage behavior is discovered, the greedy borrower's private key can be extracted, and the identity can be revealed by the financial institution. We demonstrate that the proposed scheme achieves anonymity, unforgeability, and accountability. Experiment results show the practicality of the proposed blockchain-based mortgage data management scheme.

- ▷ *Fair and privacy-preserving data trading*: We propose a blockchain-based fair and fine-grained data trading protocol with privacy preservation. We first define the fairness and privacy demands for both parties. In particular, a data buyer can declare attribute requirements for data, and it transfers funds only if it can obtain data that satisfy the requirements. A data seller can trade partial data, i.e., sensitive data are not sent to a data buyer. To achieve fair trading, by incorporating attribute-based credentials, encryption, and zero-knowledge proof, a data seller can demonstrate data availability by only disclosing the required attributes and proving the authenticity of data. For data privacy, we build a Merkle hash tree on the ciphertexts of data with a signature on its root node, which allows a data seller to send part of the data without affecting data availability verification. We formally prove that our scheme achieves privacy preservation and fairness in data trading.

- ▷ *Decentralized anonymous payment with regulatory compliance*: We propose a novel decentralized anonymous payment scheme that supports regulatory compliance and enforcement. We first introduce regulators into the system, who define regulatory policies for anonymous payment, and the policies are enforced through commitments and non-interactive zero-knowledge proofs for compostable statements. By this, users can prove that transactions are valid and comply with regulations. A tracing mechanism is embedded in the scheme to allow regulators to recover the real identities of users when suspicious transactions are detected. We provide a formal security model and proof to demonstrate that the proposed scheme can achieve desired security properties.

1.5 Thesis Outline

The organization of this thesis is as follows: Chapter 2 introduces the preliminaries utilized to design the proposed schemes and reviews the related work in privacy-preserving and blockchain-based schemes for financial services. Chapter 3 develops a blockchain-based mortgage data management scheme that can detect the double-mortgage fraud and preserve the privacy of borrowers. Chapter 4 designs a blockchain-based data trading scheme, which enables fine-grained data trading and achieves fairness of trading and privacy preservation for participants. Chapter 5 proposes a decentralized anonymous payment scheme with regulatory compliance and enforcement. Finally, Chapter 6 concludes the thesis and discusses the future research directions.

Chapter 2

Preliminary

In this chapter, we first introduce the underlying techniques exploited to design the proposed schemes. Then, we give a comprehensive survey of the literature on blockchain-based privacy-preserving financial services.

2.1 Basic Techniques

We present the preliminaries in this section, which include bilinear pairing, ElGamal encryption, structure-preserving signature, Elliptic Curve Digital Signature Algorithm (ECDSA), verifiable secret sharing, Zero-Knowledge Proof (ZKP), Zero-knowledge Succinct Non-interactive Arguments of Knowledge (Zk-SNARK), and smart contract.

2.1.1 Bilinear Pairing

G_1, G_2, and G_T are three multiplicative groups of order q. Let g and $\hat{g}$ be generators of group G_1 and G_2 respectively. $e : G_1 \times G_2 \to G_T$ is a computable bilinear map that has following properties [31]:

- $\triangleright$ Bilinear: Given $s, t \in \mathbb{Z}_p^*$, $e(g^s, \hat{g}^t) = e(g, \hat{g})^{st}$;
- $\triangleright$ Non-degenerate: $e(g, \hat{g}) \neq 1$;
- $\triangleright$ Efficient: $e(g, \hat{g})$ can be efficiently computed.

2.1.2 Negligible Function

A function $\epsilon : \mathbb{Z} \to R$ is called negligible [32] if for any $z \in \mathbb{Z}$, there exists a k such that $\epsilon(x) \leq \frac{1}{x^z}$ when $x \geq k$.

In the following, an adversary's success probability is negligible means that the probability is a negligible function of the security parameter.

2.1.3 ElGamal Encryption

ElGamal encryption is an asymmetric key encryption method [33]. There are three algorithms in ElGamal encryption, which are listed as follows:

- ▷ Key Generation: Let G be a cyclic group of order q with generator g. Alice chooses an integer x from $\{1, \ldots, q-1\}$, and computes $h = g^x$. The public key of Alice is (G, q, g, h) and the private key is x;
- ▷ Encryption: To encrypt a message $m \in G$ to Alice, Bob first chooses an integer γ randomly from $\{1, \ldots, q-1\}$. Then, Bob computes $C_1 = g^\gamma$ and $C_2 = h^\gamma \cdot m$. The ciphertext of m is (C_1, C_2);
- ▷ Decryption: Given (C_1, C_2) and the private key (x), Alice calculates $m = C_2/(C_1^x)$.

2.1.4 Structure-preserving Signature

In a structure-preserving signature scheme [34], messages, the verification key, and signatures are all group elements from G_1 and G_2. We recall a structure-preserving signature scheme proposed by Groth [35]. Note that the original scheme can support signing a matrix of elements. Here, we only require it to sign a single group element. Moreover, as [36], we consider two variants of the scheme, Groth_1 and Groth_2. Groth_1 signs an element in G_1 while the public key is in G_2. Groth_2 signs an element in G_2 while the public key is in G_1. Groth_1 is described as below, and Groth_2 can be attained by swapping the roles of G_1 and G_2.

- ▷ $\mathsf{Groth}_1.\mathsf{Setup}(\lambda)$: Given a security parameter (λ), the algorithm outputs public parameters (pp), which consist of $(G_1, G_2, G_T, p, e, g, \hat{g})$, where p is the order of G_1 and G_2. g and $\hat{g}$ are generators of G_1 and G_2, respectively. The algorithm also outputs a random number $(Y \in G_1)$;

▷ $\mathsf{Groth_1.KeyGen}(pp)$: This algorithm randomly chooses an element $u \in \mathbb{Z}_p^*$ as the private key sk and computes the corresponding public key as $pk = U = \hat{g}^u$;

▷ $\mathsf{Groth_1.Sign}(pp, sk, M)$: This algorithm signs a message $M \in G_1$ using sk. It randomly selects a number $r \in \mathbb{Z}_p^*$, and computes

$$\hat{R} = \hat{g}^r \quad S = (Y \cdot g^u)^{\frac{1}{r}} \quad T = (Y^u \cdot M)^{\frac{1}{r}}$$

The signature of M is $\sigma = (\hat{R}, S, T)$;

▷ $\mathsf{Groth_1.verify}(pp, pk, \sigma, M)$: Given message M and its signature σ, the algorithm checks the correctness of the signature. It outputs 1 if $e(S, \hat{R}) = e(Y, \hat{g}) \cdot e(g, \hat{U})$ and $e(T, \hat{R}) = e(Y, \hat{U}) \cdot e(M, \hat{g})$;

▷ $\mathsf{Groth_1.Rand}(pp, \sigma)$: This algorithm randomizes signature σ. It first picks $r' \in \mathbb{Z}_p^*$ and computes

$$\hat{R}' = \hat{R}^{r'} \quad S' = S^{\frac{1}{r'}} \quad T' = T^{\frac{1}{r'}}.$$

The randomized signature is $\sigma' = (\hat{R}', S', T')$.

2.1.5 ECDSA

ECDSA [37] belongs to the elliptic curve cryptosystems. The algorithms in ECDSA are described as below.

ECDSA.KGen(1^λ): Let G be an elliptic curve group chosen from a security parameter λ. G is equipped with the order q and a generator g. The algorithm chooses a collision-resistant hash function $H : \{0,1\}^* \to Z_q$. x is a random number in Z_q^*, which is set to be the secret key sk. The corresponding public key is $pk = g^x$.

ECDSA.Sign(sk, m): Given a message m that needs to be signed, a signer chooses a $k \in Z_q^*$ randomly, and computes $R = g^k$. Denote the x coordinate of R to be r. The signer calculates $s = k^{-1}(H(m) + rx) \bmod q$. This algorithm outputs $\sigma = (r, s)$ as the signature of m.

ECDSA.Verify(σ, m, pk): Given a signature σ on m, the verifier first parses the signature as $\sigma = (r, s)$, and computes $v = H(m)$ and $w = s^{-1} \bmod q$. Then, it calculates $u_1 = vw \bmod q$, $u_2 = rw \bmod q$ and $R = g^{u_1} \cdot pk^{u_2}$. If $R = r \bmod q$ holds, the signature is a valid signature and the verifier outputs 1. Otherwise, it outputs 0.

2.1.6 Verifiable Secret Sharing

The (k, n) threshold secret sharing scheme [38] can be utilized to distribute a secret s among n participants. The share each party owns is an evaluation of a polynomial $f(X) = \xi_{k-1}X^{k-1} + \cdots + \xi_1 X + s$.

For a party i, $i \in [n]$, the share can be calculated as $f(X_i)$. Given any k different shares, which provide k different points (x_i, y_i), $1 \leq i \leq k$, the secret s can be recovered as

$$s = \sum_{i=1}^{k} \rho_i y_i, \qquad \text{where} \qquad \rho_i = \prod_{j=1, j\neq i}^{k} \frac{-x_j}{x_i - x_j}.$$

Shamir's secret sharing can be publicly verifiable by utilizing the technique proposed by Feldman [39]. Verifiability means one can verify that a share is correctly generated. Let G be a multiplicative group which is equipped with the order q and a generator g. The basic idea of verifiability is to publish a sequence $(g^{\xi_{k-1}}, \cdots, g^{\xi_1}, g^{\xi_0})$, where $\xi_{k-1}, \cdots, \xi_1, \xi_0$ are coefficients of the polynomial $f(X)$ and $g^{\xi_0} = g^s$. Given a share y_i and its corresponding x_i, one can verify that the share is correct by checking whether the equation $g^{y_i} = \prod_{j=0}^{k-1} (g^{\xi_j})^{x_i^j}$ holds.

2.1.7 Zero-Knowledge Proof (ZKP)

ZKP allows a prover to convince a verifier that it knows a secret without leaking the secret itself [40]. For proving statements related to discrete logarithms and relations in prime-order groups, sigma-protocols are widely used because of the simplicity and versatility [41].

sigma-protocols: Let R be a binary relation and $(x, Y) \in R$. A prover that has x and wants to prove to the verifier that it knows x such that $(x, Y) \in R$ can interact with the verifier utilizing the following 3-move sigma-protocol, where P_1, P_2, and V are three algorithms.

1. With the input (x, Y), the prover randomly chooses a random number a and computes a number T by using $P_1(x, Y, a)$. Then, the prover sends T to the verifier while keeping a private;

2. The verifier randomly selects a challenge c from a challenge set C, and sends c to the prover;

3. The prover computes an element s using $P_2(x, Y, a, c)$ and returns s to the verifier. The verifier runs $V(Y, T, c, s)$, and if the output is true, the verifier accepts the proof.

Table 2.1: ZKPoK for a DDH tuple

Let $g_1, g_2, u_1, u_2 \in G$		
Prover		Verifier
$(u_1, u_2, k = log_{g_i} u_i)$		(u_1, u_2)
$r \leftarrow Z_q^*, r_i = g_i^r$	$\xrightarrow{r_1, r_2}$	
	$\xleftarrow{b}$	$b \overset{R}{\leftarrow} Z_q$
$v \leftarrow r + kb$	$\xrightarrow{v}$	accept iff $\forall i : g_i^v = r_i u_i^c$

In the sigma-protocol, the relationship R that a tuple of elements (g_1, g_2, u_1, u_2) forms a DDH tuple is presented as $(w, g_1, g_2, u_1, u_2) \in R \Leftrightarrow u_1 = g_1^w \wedge u_2 = g_2^w$, where (g_1, g_2, u_1, u_2) are in a cyclic group G. The proving process of a sigma-protocol for the DDH tuple is shown in Table 2.1.

By utilizing the Fiat-Shamir transformation [42], ZKP can be converted to an Non-Interactive Zero-Knowledge Proof (NIZKP) protocol [43]. The interaction between the two parties is removed by making use of a collision-free hash function to obtain the challenge. Compared with the interactive ZKP, an NIZKP protocol outputs a common reference string in the initial phase.

An NIZKP consists of three algorithms which are described as below.

NIZKP.Setup(1^k): Given a security parameter k, this algorithm generates a common reference string crs.

NIZKP.Proof(crs, x, w): This algorithm is run by the prover to generate a proof of the statement x. Given the crs and a witness w, the algorithm can output a proof π.

NIZKP.Verify(crs, x, π): This algorithm is run by the verifier to verify whether the statement x is true. Given the statement x, the proof π and crs, the algorithm can output a bit $b \in \{0, 1\}$.

An NIZKP protocol Π is zero-knowledge, if there exist an efficient simulator $S = (S_1, S_2)$ such that

$$\begin{aligned} &|\Pr[crs \leftarrow Setup_\Pi(1^k) : A(crs) = 1] - \\ &\Pr[(crs, \tau) \leftarrow S_1(1^k) : A(crs) = 1]| \leq \varepsilon_1, \end{aligned}$$

where ε_1 is a negligible number, and the probability that A wins the experiment Zero-Knowledge$^{\Pi}_{A,S}(k)$, which is shown in Fig.2.1 is also negligible, which means that

$$|\Pr[Zero - Knowledge^{\Pi}_{A,S}(k)] = 1 - \frac{1}{2}| \leq \varepsilon_2,$$

where ε_2 is a negligible probability.

```
Experiment $Zero - Knowledge^{\Pi}_{A,S}(k)$
    $b \leftarrow \{0,1\}$
    $(crs, \tau) \leftarrow S_1(1^k)$
    $b^* \leftarrow A^{P_b(\cdot,\cdot)}(crs)$
        Oracle $P_0$: Algorithm NIZKP.Proof on input $(x, \omega)$
            If $(x, w) \in R$, return $\pi \leftarrow$ NIZKP.Proof$_{\Pi}(crs, x, \omega)$
            Else, return $\perp$
        Oracle $P_1$: Simulator $S_2$ on input $(x, \omega)$
            If $(x, w) \in R$, return $\pi \leftarrow S_2(crs, \tau, x)$
            Else, return $\perp$
    If $b = b^*$, return 1
    Else, return 0
```

Figure 2.1: Zero-knowledge experiment

In the experiment Zero-Knowledge$^{\Pi}_{A,S}(k)$, $b \in \{0, 1\}$ is chosen by the challenger and given to adversary A. Simulator S_1 generates a common reference string (crs) and a trapdoor (τ). Adversary A can access Oracle P_b. In the experiment, P_0 represents the algorithm NIZKP.Proof that outputs a proof π with the inputs crs, statement x, and witness ω. P_1 represents simulator S_2, which can outputs a proof with the inputs crs, τ and x. Note that S_2 can output a proof without using witness ω [44]. After accessing Oracle P_b, A needs to output a guess of b, which is denoted as b^*. If $b = b^*$, the experiment returns 1.

2.1.8 Zk-SNARKs

Let C denote an arithmetic circuit. R_C represents an NP relation $R_C = \{(x, w)|C(x, w) = 0\}$. The language for R_C is $L_C = \{x| \exists\, \omega, s.t.\, C(x, \omega) = 0\}$. Zk-SNARKs are suitable for proving statements that are denoted as arithmetic circuits. The proofs have short sizes and can be verified within a few milliseconds. A zk-SNARK scheme [45, 46] for language L_C contains three algorithms (KeyGen, Prove, Verify):

KeyGen(λ, C): The inputs of the algorithm include a security parameter λ and a circuit C. The outputs are a proving key pk and a verification key vk.

Prove(pk, x, ω): With the inputs pk and an witness ω of x, the algorithm returns a proof π for the statement x.

Verify(vk, x, π): Given vk, x, and π, the algorithm outputs 1 if π is a valid proof for $x \in L_C$.

2.1.9 Smart Contract

A smart contract is a program stored on a blockchain. Smart contracts are used to automate the execution of a workflow and trigger the next action when specific conditions are met, so that all parties can obtain the outcome immediately without any intermediary's involvement [47]. Actions in a smart contract can be issuing a ticket, sending a notification, or releasing funds to an appropriate entity [48]. Since there is no third party involved and records of transactions are shared across participants, the execution of a smart contract is efficient, trusted, and transparent.

2.2 Related Work

We comprehensively review the literature on privacy and regulation in three blockchain-based financial services, which are most relative to this thesis, including privacy-preserving and accountable lending, privacy-preserving and fair data trading, and privacy-preserving and regulated decentralized anonymous payment.

2.2.1 Privacy-preserving and Accountable Lending

Mortgage fraud detection has already been investigated by social scientists and economists to reduce risks for financial institutions. The traditional approach for mortgage fraud prevention is to build a centralized computer system or the regulator who is responsible for collecting and analyzing abnormal situations. Ngai et al. [49] employed data mining techniques, for example, logistic models and neural networks, to detect financial fraud. Gestel et al. [50] analyzed the facilitating circumstances of the mortgage fraud and proposed two approaches to reduce the mortgage fraud, one is to increase the integrity of professionals, and the other focuses on the information exchange. The methods above deal with the fraud detection based on mortgage information of borrowers, but the privacy of borrowers is not protected during mortgage data sharing.

To mitigate the loan-associated financial risk, Reno et al. proposed a blockchain-based lending system [51], where borrowers can use Ethereum Request for Comments (ERC)-20 standard tokens as collateral to obtain a certain amount of cryptocurrency. To guarantee the financial benefit of lenders, all calculations are performed based on fiat money to avoid financial loss caused due to fluctuations in the exchange rate of the cryptocurrency. Smart contracts are utilized to create ERC-20 tokens and administrate the lending and borrowing processes. In their proposed system, tokens are used as collateral instead of real assets of borrowers, and privacy preservation is not taken into account. Praitheeshan et al. proposed a distributed lending model [20] based on Hyperledger Besu, which is an advanced Ethereum client that implements the privacy group feature and facilitates private smart contract executions. Only users in a private group can access their associated shared private states. The private group IDs are required to retrieve the private smart contract executions, and the IDs are protected using symmetric and asymmetric encryption algorithms such that only the users in the group can decrypt the group ID. Schar summarized the decentralized debt market in Decentralized Finance (DeFi) [52], which refers to an alternative financial infrastructure built on the Ethereum blockchain. In the DeFi ecosystem, users do not need to identify themselves, and they can borrow money or provide liquidity to the system for earning interest. To reduce the risks of lenders, there exist two distinct lending approaches: First, loans can be repaid atomically, which means if a borrower does not repay the funds on time, the lending transaction will be invalid, and the loan itself will be reverted. One limitation of this method is that the loans must be received, used, and repaid within the same blockchain; Second, loans of borrowers are secured with collateral, which is locked in a smart contract until the debt is repaid. There are many lending protocols in DeFi, such as Aave [53], Compound [54], and dYdX [55].

Considering that blockchain is a public ledger and data on the chain are shared among multiple parties. User identity privacy should be protected. To preserve users' privacy, Hardjono et al. [56] proposed a verifiable anonymous identities scheme ChainAchor for permissioned blockchains, where the anonymity of a user is achieved by utilizing the zero-knowledge proof protocol. Users need to prove their membership before registering their public transaction keys. After successfully completing the zero-knowledge proof protocol, a user generates a public transaction key, which can be used for transacting with others. Each user can own many unlinkable transaction public keys, and revealing the ownership of one key does not affect the security of other keys. Considering that in most anonymous credential systems, the distinct certification authority of each organization can leak the association of a user to its issuing organization, Camenisch et al. proposed the delegatable anonymous credential scheme [36] to hide that information. Specifically, a root authority delegates the generation of credentials to intermediate authorities, and when users use the credentials, only the root authority is revealed. Bogatov et al. extended the scheme of [36] by adding the functionalities of credential revocation and authorization auditing, and proposed a privacy-preserving authentication and authorization mechanism [57] for permissioned blockchain systems.

The mechanisms that can achieve anonymous authentication also include group signatures, ring signatures, and anonymous credentials. Group signatures [58] [59] allow a group member to sign a message on behalf of the group without leaking its real identity information. Only the group manager can trace the identity of the signer. Ballare proposed a dynamic group signature scheme [60], where the addition and removal of group members are allowed. Boneh et al. constructed a short group signature scheme [61], where the size of the signature is almost the same as the RSA signature scheme. Ring signatures also allow a member of a group to sign a message, but others do not know which member signed the message. Different from group signature schemes, there is no group manager in a ring signature [62] [63] scheme, and there is no need to initialize the group. Au et al. [64] proposed an identity-based ring signature in the standard model, where the signature size is linear in the cardinality of the ring. Anonymous credentials [65] [66] [67] can achieve stronger privacy preservation than traditional credentials. A user can own many pseudonyms in an anonymous credential system, and the pseudonyms of a user cannot be linked by others. Camenisch et al. [68] proposed an anonymous credential scheme that can support anonymity revocation and prevent the misuse of anonymity. Yang et al. [69] proposed a decentralized anonymous credential system with reputation. By utilizing the blockchain technique, no trusted credential issuer is needed in the system, and the blacklist can be shared among different service providers.

2.2.2 Privacy-preserving and Fair Data Trading

The problem of fair data trading has been studied for decades. Researchers have shown that without a trusted third party, the fairness of trading is unachievable [70]. However, with the emergence of blockchain-based cryptocurrencies, fair data trading can be achieved in a completely trustless manner, where blockchain replaces the role of the trusted party. As data availability and retrievability are challenging issues in the design of fair data trading protocol, in the following, we review the related works in terms of data availability and retrievability.

The first blockchain-based fair data trading protocol named Zero-Knowledge Contingent Payment (ZKCP) was proposed by Gregory Maxwell [71]. In the protocol, data availability can be verified by a public predicate function. For example, the availability of a movie can be verified by a hash on the movie file. The process of the protocol is as follows: A data seller first encrypts digital goods with a symmetric key. Then, it generates a zero-knowledge proof that proves the data encrypted satisfy a public predicate function, and a value h is the hash of the key. A data buyer verifies the proof and builds a hash lock smart contract which pays funds to one who opens the hash lock h by revealing the preimage of the hash. The data seller then sends the key to the smart contract, which transfers the funds to the data seller after checking the correctness of the key. Thus, in ZKCP, data retrievability is achieved by revealing the encryption key of the data. ZKCP protocol incurs setup issues since the setup of the zero-knowledge proof system [72], which is a Zk-SNARKs system, is done by the data buyer. Moreover, there is substantial proving overhead for the data seller, and it cannot process complicated validation of large-scale data. To address the issues in ZKCP, Li et al. proposed a fair-exchange protocol supporting practical data exchange that is called ZKPlus [73], in which a new Commit-and-Prove Non-Interactive Zero-knowledge argument of Knowledge (CP-NIZK) scheme is introduced to replace the Zk-SNARKs in ZKCP. ZKPlus has a public setup as commitment schemes and supports data-parallel computations, which enables it to handle complicated data validation.

For the verification of data availability, besides the predicate function, a few works use sampling techniques [28, 74, 75]. Delgado-Segura et al. presented a fair protocol for data trading based on Bitcoin transactions, where a sampling technique called cut and choose is used to prove the data availability [28]. The data buyer can randomly choose a subset of data that are called samples and obtain the plaintext of the sample data. By verifying the correctness of the sample data, the data buyer can validate the data availability. For data recoverability, a vulnerability of the ECDSA [76] is exploited to implicitly reveal the secret key that can decrypt all the ciphertexts. The vulnerability allows one to extract the secret key from two signatures of different messages, where the signatures are generated using the same random number. Moreover, Zhao et al. proposed a machine learning-based fair data trading scheme in the big data market [74]. In their scheme, a sampling technique and a distance metric learning method are used to prove the validity of data, and the double authentication preventing signature is used to recover the decryption key.

Another method to verify the data availability is an authentication-based method. Liu et al. proposed a blockchain-cloud transparent data marketing scheme where a distributed and trusted committee verifies the data availability and manages anonymous credentials for data sellers, which includes decentralized credential issuance and threshold openings [77]. In the scheme, data of data sellers are stored on an off-chain cloud server. With financial incentives and commitments to trading behavior stored on the blockchain, data sellers, data buyers, and the cloud server are motivated to behave correctly. Moreover, Galteland et al. proposed a blockchain-based privacy-preserving fair data trading protocol, where a data manager is introduced as a trusted authority [24]. Data signed by the manager are believed to be authentic. The manager encrypts data with the public key of the data owner and signs the data. The data buyer can verify the signature to attest to the data availability. The data owner then generates a re-encryption key, and it sends the key and a correctness proof of the key to a smart contract to prove data retrievability. With the re-encryption key, the data buyer can transform the original ciphertext into a ciphertext under its public key and decrypt it with its private key.

In our work, we also employ an authentication-based mechanism to prove the data availability. Instead of a public authority in the system, our scheme can support multiple issuers. Data signed by one of the issuers are acceptable and can be verified without leaking the identity of the issuer. Moreover, the proposed scheme can achieve fine-grained data selling by only exposing required data to data buyers. The comparison of related works is listed in TABLE 2.2.

Table 2.2: Comparison of related works

Scheme	Fairness	Privacy	Method for Data Availability	Method for Data Retrievability	Issuer Hiding	Fine-grained
[71]	✓	✗	Predicate function	Key revealing	NA	✗
[73]	✓	✗	Predicate function	Key revealing	NA	✗
[28]	✓	✗	Sampling technique	Key recovery	NA	✗
[74]	✓	✗	Sampling technique	Key recovery	NA	✗
[77]	✓	✓	Committee authentication	Key recovery	NA	✗
[24]	✓	✓	Signature authentication	Re-encryption key	✗	✗

Apart from raw data trading, some researchers proposed "data processing results" trading. Dai et al. presented a secure data trading ecosystem, where a data processing-as-a service model is introduced to defend against the dishonest data trading platform or data broker [78]. In the ecosystem, both the data broker and a buyer cannot access the raw data of a data seller. Instead, they can only obtain the data processing result that they require. The data processing is performed on the trusted nodes in the blockchain network that are equipped with the trusted execution environment, such as Intel's Software Guard Extensions (SGX), and only the analysis results are sent to the data buyer.

2.2.3 Privacy-preserving and Regulated Decentralized Anonymous Payment

Considering the transparency of blockchain [79], to achieve privacy-preserving cryptocurrency, Saberhagen proposed Cryptonote [80], which utilizes traceable ring signatures to hide the participants of transactions and one-time keys to prevent double-spending of a coin. CoinJoin [81] uses coin mixing services to protect the originators of transactions. However, a centralized CoinJoin server is required in the system. Monero protocol [82], which is designed based on CryptoNote, uses the technique of Confidential Transactions to hide the amounts of transactions. Considering that Bitcoin only preserves user privacy through pseudonyms, Miers et al. proposed Zerocoin [83], which uses an accumulator scheme and non-interactive ZKP to break the link between the Mint transactions and Spend transactions. Due to the fact that Zerocoin still reveals the destinations and amounts of transactions, Ben-Sasson et al. formulated Decentralized Anonymous Payment (DAP) schemes [84] and proposed Zerocash as a practical instantiation. By using Zk-SNARKs, commitment schemes [85], and collision-resistant hash function-based Merkle tree, Zereocash preserves the confidentiality of the origin, the destination, and transferred amount of a payment.

To solve the regulatory issues raised by anonymous transactions, many efforts have been made to balance the privacy protection and enforcement of regulatory policies. Garman et al. [86] added the policy-enforcement mechanisms to Zerocash, and the proposed approach allows selective user tracing and tained coins tracing. However, they are achieved by using Zk-SNARKs techniques, resulting in the poor performance of the system. Wu et al. proposed a regulated digital currency [87], where transactions are supervised by an auditor, who can monitor the flow of money and obtain the identities of users. In the scheme, coins in the system have fixed denominations, and transferred amounts are not concealed. Ma et al. proposed SkyEye [88], which allows a regulator to trace users' identities by adding identity proofs in each anonymous transaction. The identity proof is used to prove user legitimacy and achieve tracing. However, the regulator can recover the identities of all transaction participants regardless of whether a user violates the policies. Wust et al. proposed PRCash [89], which is designed based on Mimblewimble and can achieve private and regulated transactions in a permissioned blockchain setting. Lin et al. presented a decentralized condition anonymous payment system, [90], where for each transaction, a new anonymous address will be created by a transaction participant from its long-term address. In the scheme, users' long-term addresses are encrypted by the manager's public key, and the manager needs to obtain the long-term address of a user to determine whether the user is legitimate, which causes a large overhead for managers and the disclosure of user privacy.

Chatzigiannis et al. investigated the distributed payment schemes [91] that provide auditability functionalities for regulators. In the paper, the relevant work falls into two categories: organization-based auditability, where some third parties such as exchanges need to offer cryptographic solvency proofs [92], and user/transaction level-based auditability, where regulators can learn about the previous transactions of a specific user or the involved users of a transaction. The authors also summarized the related schemes in terms of security guarantees, efficiency, and general properties such as the account model and consensus protocol used. Chatzigiannis et al. also proposed a distributed payment scheme [22], which allows a third party to audit the transactions generated from an authorized set of entities. The provided pruning function can save the storage overhead of a ledger without affecting the audit functionalities. Li et al. proposed the concept of Traceable Monero [93], a novel method to balance the user anonymity and accountability on top of Monero. In their framework, the tracing authority is optimistic, which is only involved when investigations in malicious transactions are required. They presented a construction by cleverly taking advantage of the trick of verifiable encryption to identify the long-term account and the one-time account, respectively. Androulaki et al. proposed an auditable token system [94] for enterprise networks. The system is built based on a permissioned blockchain and uses conservative computational assumptions such as discrete-logarithm assumption.

2.3 Summary

In this chapter, we have briefly presented the preliminaries, which include bilinear pairing, ElGamal encryption, structure-preserving signature, ECDSA, verifiable secret sharing, ZKP, Zk-SNARKs, and smart contract. Moreover, we have given a literature review on the privacy preservation and regulation of blockchain-based financial services, including privacy-preserving and accountable lending, privacy-preserving and fair data trading, and privacy-preserving and regulated decentralized anonymous payment. In the following chapters, we will introduce the proposed schemes to solve the challenging issues that are not solved in existing work and balance privacy and regulation in blockchain-based financial services.

Chapter 3

Balancing Borrower Privacy and Double-Mortgage Regulation for Blockchain-based Lending

3.1 Motivations

Mortgage loan enables borrowers to pledge valuable assets to financial institutions to acquire a certain amount of loans. Compared with unsecured loan, borrowers obtain loans with a lower interest in a mortgage loan [95]. The borrowers can use the mortgage loan to make new investment and expand the scalability of their businesses. When a borrower defaults on the loan or otherwise fails to abide by the loan items, the lender takes possession of the mortgages for paying off the debt.

However, a mortgage loan allows a borrower to retain the ownership of the valuable asset. A borrower may use the same asset as collateral to obtain multiple loans from different financial institutions, which is denoted as the double-mortgage behavior. In this work, we focus on scenarios where mortgagors borrow money which is much greater than the value of the properties, which leaves lenders exposed to financial risks. A straightforward method to detect double-mortgage behavior is to establish a centralized database that records every mortgage case and identifies the double-mortgage misbehavior [96]. The solution may not work for efficient mortgage management because of the following two reasons: (1) There is lack of mutual trust among financial institutions to agree on the correctness and reliability of a centralized party [97]. Even if there is a trusted third party, a large number of interactions between the financial institutions and the third party will be involved in the mortgage management; (2) The mortgages may contain sensitive personal or enterprise-level information and direct data sharing among different parties is restricted by regulations, such as General Data Protection Regulation (GDPR) in Europe [98]. As a result, how to design a mechanism that quickly detects the double-mortgage behavior of borrowers while protecting honest borrowers' privacy is a challenging issue.

Blockchain technology is envisioned to promote universal trust among partners and increase operational efficiency for business process management [97]. As the enabling technologies that support Bitcoin, blockchain is actually a decentralized ledger that can be publicly validated. The ledger consists of an increasing number of blocks of transactions. The consensus mechanism of blockchain ensures the consistent view of the public ledger and prevents the adversaries from deleting or modifying the appended data on the chain. As a trusted distributed database, blockchain technology provides a way of recording currency transactions or any other digital information that is designed to be transparent, auditable, highly resistant to outages [99]. Currently, the blockchain applications span across diverse fields far beyond cryptocurrencies, which include supply chain, insurance, economics, Internet of things, etc. [100–102]. However, simply putting the records of mortgage on the blockchain and comparing them for double-mortgage detection does not solve the problem since loan companies may not be willing to publish their data in a public manner. Moreover, even if the double-mortgage behavior is discovered, the anonymity nature of the blockchain prevents the misbehaving borrower's true identity from being revealed.

To address the issue, we construct a Blockchain-based Accountable and Privacy-preserving Industrial Mortgage scheme (BAPIM), which not only protects the privacy of honest borrowers, but also helps financial companies or financial institutions to detect the double-mortgage behavior of borrowers. When the behavior is detected, the true identity of the borrower can be revealed publicly. BAPIM is built based on blockchain and double authentication preventing signature to achieve secure and reliable mortgage lending. The main contributions of BAPIM are two folds.

- A blockchain-based accountable mortgage lending scheme is designed to prevent a borrower from using the same asset as collateral to obtain multiple loans from different financial institutions. By taking advantages of the transparency and irreversibility of blockchain technology, BAPIM can help financial institutions efficiently identify the double-mortgage behavior of a borrower and reduce the financial risks.

- The extractability of a malicious borrower's identity is achieved by integrating ElGamal encryption, zero-knowledge proof and verifiable secret sharing. Financial institutions can anonymously authenticate the honest borrowers during the process of mortgage. Only when the double-mortgage behavior is discovered, the greedy borrower's private key can be extracted and the identity can be revealed by the financial institution. The security model of BAPIM is defined, and BAPIM is proven to be able to fulfill all the desirable security objectives under the security model.

3.2 Problem Statement

In this section, we first present the system model, and analyze the security threats and the goals that need to be achieved in the system. Then, we define the system components and security model of BAPIM.

3.2.1 System Model

In mortgage lending, borrowers usually mortgage their valuable assets to obtain loans from financial institutions. Borrowers can be individuals who are going to start an active business venture, or companies that are willing to make new investment or expand the business, but do not have sufficient money for investment. Thus, they turn to the financial institutions for financing and get loans. In a mortgage loan, the loan is secured by the asset in the borrower's name. If the borrower wants to get a loan from a financial institution, the institution first checks the ownership of the asset and evaluates the value. If the institution believes the value is higher than the amount of money that the borrower would like to get and the risk is low, the financial institution would provide the loan to the borrower. If the borrower fails to repay the loan in full, the institution can seize and sell the pledged asset to recover any outstanding balance.

As shown in Fig.5.1, there are three entities in our system:1) borrowers, 2) financial institutions, and 3) a Trusted Third Party (TTP).

1) *Borrowers.* Borrowers can mortgage their valuable assets to acquire loans from financial institutions.

2) *Financial Institutions.* A financial institution can provide loans to borrowers if the loan request of a borrower is verified. Financial institutions also upload the records of mortgage information to the public blockchain for double-mortgage behavior detection.

3) *TTP.* It is a trusted entity that is responsible for issuing anonymous identity credentials to borrowers when borrowers register the mortgage service. TTP is also in charge of certifying a borrower's legal ownership of an asset.

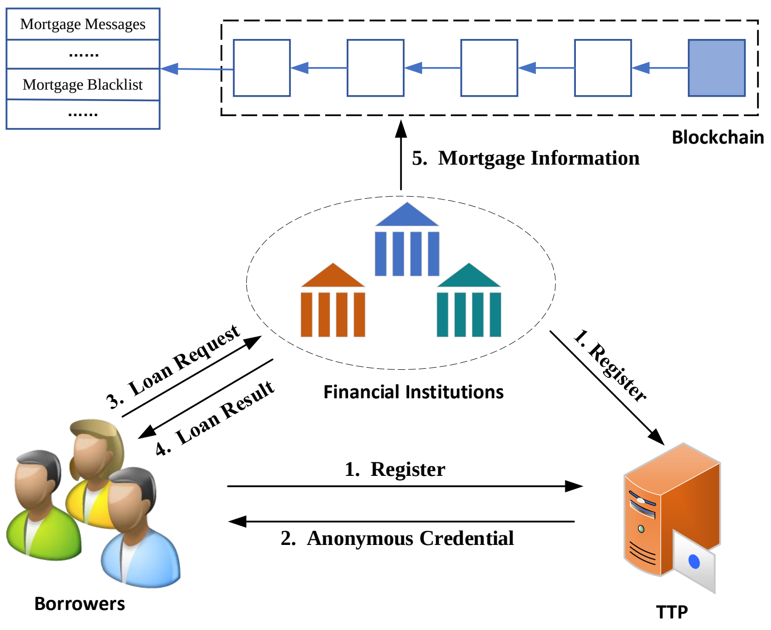


Figure 3.1: System model

At a high level, BAPIM works as follows. Borrowers and financial institutions first register themselves to TTP. For a borrower, after sending its identity and public key to the TTP, it obtains an anonymous identity credential from TTP. When a borrower plans to apply for a mortgage loan from a financial institution, the borrower generates a loan request and sends it to the financial institution. After verifying the validity of the application, the financial institution offers the mortgage loan to the borrower and uploads the mortgage message to the blockchain. If the financial institution discovers the behavior of double-mortgage, it can extract the identity of the malicious borrower and add the identity and the proof of the misbehavior to the mortgage blacklist maintained by the blockchain.

3.2.2 Attack Model

In our system, attackers can be dishonest borrowers, who apply for loans in the name of others or pledge the same asset to different financial institutions to get multiple loans. Moreover, the attackers can be honest-but-curious financial institutions, which honestly process the mortgage lending of borrowers but tries to obtain as many borrowers' identities as possible from the mortgage data on the blockchain. All the information exchanged between a financial institution and a borrower is transmitted in a secure channel. Attackers can access the mortgage data stored on the blockchain.

3.2.3 Design Goals

In mortgage loans, borrowers are unwilling to disclose their mortgaged assets to others. In addition, financial institutions need to verify the legality of loan requests. A dishonest borrower may pledge the same asset to different financial institutions to get more loans. Hence, double-mortgage behavior should be detected among the financial institutions while the identities and pledged assets of honest borrowers are not revealed. Therefore, for the purpose of achieving a privacy-preserving and accountable mortgage loan, three security goals should be achieved.

Anonymity: A borrower can apply for a mortgage loan without revealing its true identity, which means each financial institution does not know the identity of an honest borrower.

Unforgeability: An attacker cannot forge a legitimate loan request that can pass the verification, which means whether a borrower has the ownership of an asset and the signature of a loan request is valid can be verified by the financial institution.

Accountability: If the behavior of double-mortgage is discovered, the financial institution is aware of the behavior and can extract the borrower's identity without the assistance of TTP.

3.2.4 System Components

BAPIM consists of the following algorithms, namely, KGen, Register, ReqGen, ReqVerify, and Extract.

- **KGen**(λ): Given a security parameter λ, the algorithm outputs a secret key sk and a public key pk. For simplicity, we use $(SK_b, PK_b), (SK_c, PK_c)$ and (SK_T, PK_T) to denote the secret-public key pair for a borrower, a financial institution and TTP respectively.
- **Register**(ID, PK_b, σ_b, SK_T): Given the (ID, PK_b) of a borrower, the signature σ_b on (ID, PK_b) and the secret key SK_T of TTP, the algorithm outputs an anonymous identity certificate Crt_1 of the borrower.

- **ReqGen**(A, SK_b, PK_c): Given a mortgage asset A, SK_b, and PK_c, the algorithm outputs an $m = (H(A), PK_c)$ and a loan request Req, where H is a cryptographic hash function.
- **ReqVerify**(PK_b, PK_T, Req): Given the PK_b, PK_T and a loan request Req, the algorithm outputs a bit $b \in \{0, 1\}$.
- **Extract**$(m_1, m_2, \sigma_1, \sigma_2)$: Given two mortgage messages (m_1, m_2) with the same asset but different financial institutions and their signatures (σ_1, σ_2), the algorithm outputs a secret key SK_b and an identity ID.

3.2.5 Security Model

In mortgage lending, the security requirements include anonymity, unforgeability, and accountability. The game-based approach is employed to formally define the security model of BAPIM. The adversary can make queries to oracles that are defined in the games.

Definition 1 (GAME Anonymity): A challenger $\mathfrak{C}$ and an adversary $\mathfrak{A}$ are involved in the interactive game, which defines the property of anonymity. The details of the game are shown as below.

Initialization: Given a security parameter λ, $\mathfrak{C}$ runs **KGen** to generate two public-private key pair (pk_b, sk_b) and (pk_T, sk_T). $\mathfrak{C}$ keeps (sk_b, sk_T) itself and sends (pk_b, pk_T) to $\mathfrak{A}$.

Query: $\mathfrak{A}$ can make a polynomial number of queries to the Register oracle. When $\mathfrak{A}$ makes a registration query on an ID, the Register oracle encrypts the ID with pk_b and obtains the ciphertext $\overline{C}$, then it signs the ciphertext using sk_T to generate the signature σ. $\mathfrak{C}$ returns $(\overline{C}, \sigma)$ to $\mathfrak{A}$.

Challenge: $\mathfrak{A}$ chooses two IDs I_0, I_1 with the same length and sends them to $\mathfrak{C}$. $\mathfrak{C}$ chooses a random bit $b \in \{0, 1\}$ and encrypts the ID I_b with pk_b. Then, $\mathfrak{C}$ signs the ciphertext C^* and sends the signature σ^* and C^* to $\mathfrak{A}$.

Guess: For the I_b which is encrypted by $\mathfrak{C}$, $\mathfrak{A}$ outputs a guess $b' \in \{0, 1\}$.

$\mathfrak{A}$ wins the game if the guess of $\mathfrak{A}$ is correct. The advantage of $\mathfrak{A}$ in this game is defined as $\Pr[b' = b] - \frac{1}{2}$.

We say that BAPIM scheme achieves anonymity if the polynomial time adversary $\mathfrak{A}$ has at most a negligible advantage in the GAME Anonymity.

Definition 2 (GAME Unforgeability): The GAME unforgeability is an interactive game where $\mathfrak{C}$ and $\mathfrak{A}$ are involved. This game defines the property of unforgeability and details are shown as below.

Initialization: Given a security parameter λ, $\mathfrak{C}$ runs **KGen** to generate a private key sk_b and a public key pk_b. $\mathfrak{C}$ keeps sk_b private and sends pk_b to $\mathfrak{A}$. Moreover, $\mathfrak{C}$ initializes two empty sets $\mathcal{Q}$ and $\mathcal{R}$.

Query: $\mathfrak{A}$ can perform a polynomial number of signature queries to the Sign oracle. When $\mathfrak{A}$ makes a signature query on m, the Sign oracle first parses m as (a, c). If $a \in \mathcal{R}$, the oracle returns failure. Otherwise, it generates a signature σ on m, and returns it to $\mathfrak{A}$. Then, m is added to the set $\mathcal{Q}$ and a is added to the set $\mathcal{R}$.

Output: $\mathfrak{A}$ outputs a forged signature (m^*, σ^*).

$\mathfrak{A}$ wins the above game if σ^* is a valid signature of m^* and $m^* \notin \mathcal{Q}$.

We say that BAPIM scheme achieves unforgeability under chosen message attack if no polynomial adversary $\mathfrak{A}$ can win the GAME Unforgeability with a non-negligible probability.

Definition 3 (GAME Accountability): GAME Accountability is an interactive game between $\mathfrak{C}$ and $\mathfrak{A}$. This game defines the property of accountability and the details are shown as below.

Initialization: Given a security parameter λ, $\mathfrak{C}$ runs **KGen** to generate a private key sk_b and a public key pk_b. Then, $\mathfrak{C}$ sends pk_b and sk_b to $\mathfrak{A}$.

Output: Based on pk_b and sk_b, $\mathfrak{A}$ outputs two messages and their respective signatures $(m_1, \sigma_1, m_2, \sigma_2)$, where $m_i = (a_i, c_i)$, $i \in [2]$ and for the two messages, $a_1 = a_2$ and $c_1 \neq c_2$.

$\mathfrak{A}$ wins the above game if: (1) for $i \in [2]$, σ_i is a valid signature of m_i; (2) **Extract**$(pk_b, m_1, \sigma_1, m_2, \sigma_2)$ returns an $sk_b^{'}$, but $sk_b^{'} \neq sk_b$.

We say that BAPIM scheme achieves accountability if the polynomial time adversary $\mathfrak{A}$ can win the GAME Accountability with at most a negligible probability.

3.3 Proposed BAPIM

In this section, we first give the overview of BAPIM and then provide the details of BAPIM.

3.3.1 Overview of BAPIM

KGen: It is the key generation algorithm that is run by borrowers, financial institutions and TTP. To be specific, given a security parameter λ, a borrower can generate its secret-public key pair (SK_b, PK_b), and a financial institution can generate (SK_c, PK_c) as its secret-public key pair. TTP can also create its secret-public key pair (Sk_T, Pk_T).

Register: It is the register algorithm that is run by borrowers, financial institutions and TTP. A financial institution sends its name and public key to TTP such that borrowers can get the authenticated public key of a financial institution from TTP. A borrower registers with TTP by sending its identity ID, public key PK_b and the signature σ_b on (ID, PK_b) to TTP. If ID and PK_b are valid, TTP encrypts ID with the borrower's PK_b and the ciphertext is denoted by D. Then, TTP signs $T_1 = (PK_b, D)$, the resulting signature is denoted by σ_{T_1}. Finally, TTP sends the anonymous identity credential $Crt_1 = (T_1, \sigma_{T_1})$ to the borrower.

ReqGen: It is a loan request generation algorithm run by a borrower and TTP. When a borrower plan to apply for a mortgage loan from a financial institution, it first sends its PK_b, ID and the ownership certificate of an asset to TTP, who will generate a $T_2 = (PK_b, A)$, where A represents the asset of the borrower. TTP also generates a signature on T_2, which is denoted by σ_{T_2}. Then, TTP sends $Crt_2 = (T_2, \sigma_{T_2})$ to the borrower. The borrower creates a mortgage message $m = (H(A), PK_c)$, and generates its signature σ_m, where H is a hash function and PK_c is the public key of the financial institution. Finally, the borrower sends the loan request $Req = (m, \sigma_m, Crt_1, Crt_2)$ to the financial institution.

ReqVerify: It is a loan request verification algorithm run by a financial institution. After receiving a loan request, the financial institution first checks the validation of Crt_1 and Crt_2 by using the public key PK_T. Then, the financial institution checks whether Crt_1 is on the blacklist. If not, it verifies whether the signature σ_m of the message m is valid. If it is valid, the financial institution compares the mortgage message m with the previous mortgage messages on the blockchain that are uploaded by different financial institutions. If there is no active loan related to the asset A, the financial institution returns a success. Then, the financial institution generates a signature σ_M of $M = (m, \sigma_m, t_m)$, where t_m denotes the loan term, and uploads (M, σ_M, Crt_1) to the blockchain.

Extract: It is an identity extract algorithm that is run by a financial institution. Given the public key pk_b and two messages (m_1, m_2) with the same asset but different institutions and their corresponding signatures $(\sigma_{m_1}, \sigma_{m_2})$, the financial institution can extract the secret key of the borrower. Based on D in the Crt_1 and the secret key, the financial institution can obtain the true identity of the borrower.

3.3.2 Detailed BAPIM

The ECDSA signature, ZKP, and ElGamal encryption are employed to generate a loan request. To guarantee the extractability of the identity in case of misbehavior, when the borrower generates the signature of a mortgage message, the threshold secret sharing technique is utilized to generate a share of the secret key, and the share is embedded in the signature. To be specific, the borrower generates a polynomial of degree 1, where the constant term is the secret key of the borrower. The input for creating a share is the public key of a financial institution. Moreover, the borrower needs to prove that the share is correctly generated by using ZKP and verifiable secret sharing. When two mortgage messages contain the same asset but different institutions, one can recover the secret key and identity of the borrower by utilizing the shares in the two signatures.

The details of the proposed BAPIM are described as follows.

KGen: According to a security parameter λ, TTP chooses an elliptic curve group G, which is equipped with a prime order q and a generator g. TTP also chooses a hash function $H : \{0,1\}^* \rightarrow Z_q$. A borrower first chooses a random $sk_s \in Z_q^*$ and $x_E \in Z_q^*$, and sets $pk_s = g^{sk_s}$ and $pk_E = g^{x_E}$. The key pair (sk_s, pk_s) is used to generate and verify ECDSA signatures, and (pk_E, x_E) is used for ElGamal encryption and decryption. Let n be the number of assets that a borrower can mortgage. The borrower chooses $2n$ random numbers $a_i \leftarrow Z_q^*, i \in [n]$ and $r_i \leftarrow Z_q^*, i \in [n]$. $\{a_i\}_{i\in[n]}$ are used as the coefficients of the polynomials of degree 1. The borrower computes $F_i = (g^{r_i}, pk_E^{r_i} g^{a_i})$, where $i \in [n]$ and gets crs by invoking **NIZPK.Setup** algorithm for the DDH tuple. The secret key of the borrower is $SK_b = (sk_s, (a_i, r_i)_{i\in[n]})$. The public key of the borrower is $PK_b = (pk_s, pk_E, (F_i)_{i\in[n]}, crs)$.

A financial institution chooses a random $SK_c \in Z_q^*$ and computes $PK_c = g^{SK_c}$. The signing key pair of the financial institution is (SK_c, PK_c).

TTP also generates its signing key pair by choosing a random $SK_T \in Z_q^*$ and computing $PK_T = g^{SK_T}$. The secret key of TTP is SK_T, and the public key of TTP is PK_T. The system parameters are shown in Table 5.1.

Register: A financial institution sends its name and public key to TTP for borrowers' retrieval. When a borrower registers with TTP, the borrower sends its ID, public key PK_b and the signature σ_b on (ID, PK_b) to TTP, where σ_b is obtained by invoking **ECDSA.Sign** algorithm. After verifying ID and PK_b of the borrower, TTP chooses a random $\beta \in Z_q^*$. Then, it encrypts ID with the borrower's public key pk_s. The ciphertext is computed as $D_1 = g^{\beta}$, and $D_2 = ID \cdot pk_s^{\beta}$, where $ID \in G$. The ciphertext of ID is $D = (D_1, D_2)$.

Table 3.1: System parameters

Acronym	**Definition**
pk_s	Public key of a borrower for signature verification
sk_s	Private key of a borrower for signature generation
pk_E	Public key of a borrower for encryption
sk_E	Private key of a borrower for decryption
PK_b	Public key of a borrower
SK_b	Private key of a borrower
PK_T	Public key of TTP
SK_T	Private key of TTP
PK_c	Public key of a financial institution
SK_c	Private key of a financial institution
H	Cryptographic hash function
a_i	The coefficient of a polynomial of degree 1
F_i	Ciphertext of g^{a_i}

Then, TTP signs $T_1 = (PK_b, D)$ and gets the signature $\sigma_{T_1} = (r_{T_1}, s_{T_1})$ by invoking **ECDSA.Sign** algorithm. Finally, TTP returns $Crt_1 = (T_1, \sigma_{T_1})$ to the borrower.

ReqGen: When a borrower plan to pledge an asset A_i with $i \leq n$ to obtain a loan from a financial institution C, where $A_i \in \{0,1\}^*$, it first sends its PK_b, ID and the ownership certificate of A_i to TTP, who will generate a signature on $T_2 = (PK_b, A_i)$ by invoking **ECDSA.Sign** algorithm, and the resulting signature is denoted by $\sigma_{T_2} = (r_{T_2}, s_{T_2})$. Then, TTP sends $Crt_2 = (T_2, \sigma_{T_2})$ to the borrower.

After receiving Crt_2, the borrower generates the mortgage message $m = (H(A_i), PK_c)$, where PK_c is the public key of the financial institution. To sign m, the borrower first chooses a random $k \in Z_q^*$ and computes $R = g^k$. We denote the x coordinate of R by η. The borrower computes $s = k^{-1}(H(m) + \eta \cdot sk_s) \bmod q$ and $z = a_i \cdot PK_c + sk_s$. After that, the borrower calculates $F'_{i,2} = F_{i,2} \cdot (pk_s \cdot g^{-z})^{\frac{1}{PK_c}}$, and obtains π by invoking the NIZKP.Proof algorithm for the Decisional Diffie-Hellman (DDH) tuple with the input

$(crs, r_i, (g, pk_E, F_{i,1}, F'_{i,2})$. The signature of the mortgage message m is $\sigma_m = (\eta, s, z, \pi)$.

After that, the borrower sends the loan request $Req = (m, \sigma_m, Crt_1, Crt_2)$ to the financial institution.

ReqVerify: After receiving a loan request Req, a financial institution first verifies the validity of Crt_1 and Crt_2 by invoking the **ECDSA.Verify** algorithm with the input PK_T. Then, the financial institution checks whether Crt_1 is on the blacklist. If not, the financial institution verifies the correctness of the signature σ_m. The verification process is as follows.

The financial institution C first computes $v = H(m)$ and $w = s^{-1} \mod q$. Then, it calculates $u_1 = vw \mod q$, $u_2 = rw \mod q$, and $R = g^{u_1} pk_s^{u_2}$. After that, C checks whether $R_x = \eta$ holds. If this is not true, C aborts and returns failure. Otherwise, C computes $F'_{i,2} = F_{i,2} \cdot (pk_s \cdot g^{-z})^{\frac{1}{PK_c}}$, and verifies that the share z in the signature is generated correctly by invoking **NIZKP.Verify** algorithm for DDH tuple with the input $(crs, (g, pk_E, F_{i,1}, F'_{i,2}), \pi)$. If the returned result is 1, the validity of (m, σ_m) is proved.

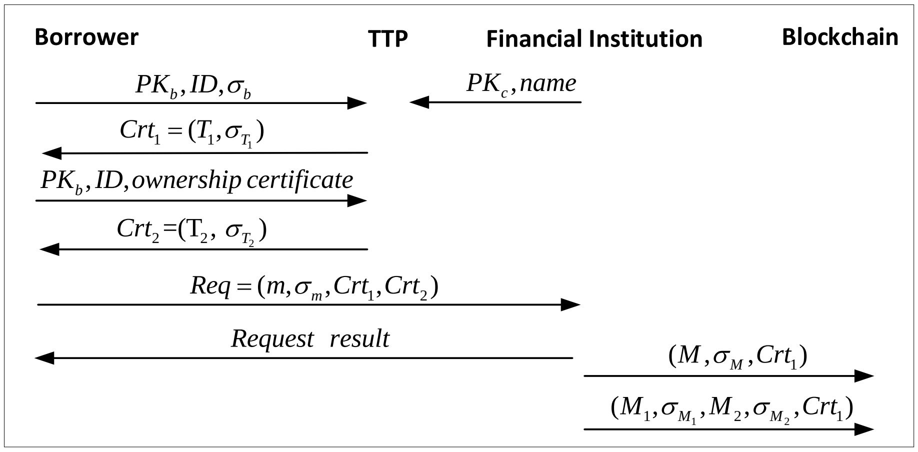
Then, C compares the mortgage message m with the previous messages which are uploaded to the blockchain by different financial institutions. If the first part of the mortgage message $H(A_i)$ is different from the other messages, C signs $M = (m, \sigma_m, t_m)$ by invoking **ECDSA.Sign** with the input SK_C. The obtained signature is denoted by σ_M. Finally, the financial institution uploads (M, σ_M, Crt_1) to the Blockchain and returns success to the borrower.

Extract: If a financial institution C discovers a mortgage message whose pledged asset is same with a previous mortgage message that is still within the term of the loan, indicating that the double-mortgage behavior is detected, the financial institution can extract the secret key of the borrower based on the two signatures $(m_1, m_2, \sigma_{m_1}, \sigma_{m_2})$. The secret key can be calculated as

$$sk_s = z_1 \frac{PK_{c_2}}{PK_{c_2} - PK_{c_1}} + z_2 \frac{PK_{c_1}}{PK_{c_1} - PK_{c_2}}.$$

After recovering the secret key sk_s of the borrower, C can decrypt the ciphertext D with the sk_s, and obtains the identity of the borrower. Finally, C adds the $\{M_1, M_2, \sigma_{M_1}, \sigma_{M_2}, Crt_1\}$, which can be seen as the proof of the misbehavior, and the identity to the mortgage blacklist maintained by the blockchain. The interaction process of BAPIM is shown in Fig.3.2.

If the repayment of a mortgage loan is finished and the borrower needs to pledge the same asset for the second time, the borrower can generate a new public-private key pair and register with TTP. Note that the new key pair is only used for the mortgage of that



Borrower
TTP
Financial Institution
Blockchain
PK_b, ID, σ_b
$PK_c, name$
$Crt_1 = (T_1, \sigma_{T_1})$
$PK_b, ID, ownership\ certificate$
$Crt_2 = (T_2, \sigma_{T_2})$
$Req = (m, \sigma_m, Crt_1, Crt_2)$
Request result
(M, σ_M, Crt_1)
$(M_1, \sigma_{M_1}, M_2, \sigma_{M_2}, Crt_1)$

Figure 3.2: Interaction process of BAPIM

asset. In general, for a borrower, there is only one public-private key pair corresponding to its ID. For the simultaneous double-mortgage, financial institutions can periodically check mortgage messages on the blockchain to detect the misbehavior.

3.4 Correctness and Security Analysis

In this section, we first show the correctness of BAPIM, then demonstrate that BAPIM achieves the properties of anonymity, unforgeability, and accountability under the security model.

3.4.1 Correctness Analysis

The correctness of the scheme can be elaborated as follows. First, a financial institution can verify the share z is correctly generated by calculating:

$$\begin{aligned}
z &= a_i \cdot PK_c + sk_s \\
g^z &= g^{a_i \cdot PK_c} \cdot pk_s \\
(g^{a_i})^{-1} &= (g^{-z} \cdot pk_s)^{\frac{1}{PK_c}} \\
F_{i,2} &= pk_E^{r_i} g^{a_i} \\
pk_E{}^{r_i} &= F_{i,2} \cdot (g^{a_i})^{-1} \\
&= F_{i,2} \cdot (pk_s \cdot g^{-z})^{\frac{1}{PK_c}} \\
&= F'_{i,2}
\end{aligned}$$

Hence, by verifying that $(g, pk_E, F_{i,1}, F'_{i,2})$ is a DDH tuple, the financial institution is convinced that z is a valid share of sk_s. Then, given

$$\begin{aligned}
z_1 &= a_i \cdot PK_{c_1} + sk_s \\
z_2 &= a_i \cdot PK_{c_2} + sk_s
\end{aligned}$$

The financial institution can recover the sk_s by

$$sk_s = z_1 \frac{PK_{c_2}}{PK_{c_2} - PK_{c_1}} + z_2 \frac{PK_{c_1}}{PK_{c_1} - PK_{c_2}}$$

3.4.2 Security Analysis

Under the security model, we analyze the security of BAPIM in this part. We demonstrate that BAPIM achieves anonymity, unforgeability, and accountability by the following theorems.

Theorem 1: The proposed BAPIM achieves anonymity if the ECDSA signature scheme is Existential Unforgeable under the Chosen Message Attacks (EUF-CMA) and the ElGamal encryption is indistinguishable under the Indistinguishable under the Chosen Plaintext Attacks (IND-CPA).

Proof: In the register phase, TTP encrypts the borrower's ID with its public key pk_s. Then, it signs the ciphertext using the **ECDSA.Sign** algorithm. The ciphertext D and its signature are included in the mortgage message that a borrower sends to a financial institution. Because the ECDSA is EUF-CMA secure, one can make sure that

Crt_1 is generated by TTP. Moreover, since ElGamal encryption is IND-CPA secure, the adversaries cannot know any information about the ID. Thus, anonymity is achieved.

Theorem 2: The proposed BAPIM achieves unforgeability if the NIZKP protocol for the DDH tuple is adaptive zero-knowledge, the ElGamal encryption is IND-CPA secure and the ECDSA scheme is EUF-CMA secure.

Proof: This security property is proved by a series of games. In the following games, the successful event in a game G_i is denoted as S_i.

Game 0: Game 0 is the GAME Unforgeability.

Game 1: The process in Game 1 is almost identical as Game 0. The difference is that in the **KGen** algorithm of Game 1, we use the simulator $S_{1,\text{NIZKP}}$ to generate (crs, τ), which is used in the NIZKP protocol.

Analysis: If the NIZKP protocol for DDH tuple is adaptive zero-knowledge, these two games are indistinguishable with a negligible probability ϵ_1.

Game 2: The process in Game 2 is almost identical as Game 1. The difference is that in the **BAPIM.ReqGen** algorithm of Game 2, we use the simulator $S_{2,\text{NIZKP}}(crs, \pi, (g, pk_E, F_{i,1}, F'_{i,2}))$ to generate the proof π used in the NIZKP protocol.

Analysis: These two games are indistinguishable if the NIZKP protocol for DDH tuple is adaptive zero-knowledge, which means that $|Pr[S_2] - Pr[S_1]| \leq \epsilon_2$.

Game 3: The process in Game 3 is almost identical as Game 2. The difference is that in **BAPIM.KGen** algorithm of Game 3, the challenger replaces the results of Elgamal encryption $(F_i)_{i\in[n]}$ with random values in G.

Analysis: Assume the maximum number of assets that a borrower can pledge is n. To compute the difference between these two games, we add $n-1$ additional hybrids and each hybrid has the form of $(F_1, F_2, \cdots, F_n)$. For each $j, 1 \leq j \leq n$, $F_j = (F_{j,1}, F_{j,2})$.

Let $\mathcal{H}_0 = (F_1, F_2, \cdots, F_n)$ where each F_j is generated by invoking the ElGamal encryption. In hybrid $\mathcal{H}_j$, we randomize all F_i for $i < j$. To be specific, for $i < j$, we set $F_i = (g^{r_i}, R \cdot g^{a_i})$, where R is a random value in group G. For $i \geq j$, we set $F_i = (g^{r_i}, (g^{sk_s})^{r_i} g^{a_i})$.

Because the ElGamal encryption is IND-CPA secure, the probability to distinguish the two consecutive hybrids is restricted by a negligible probability ϵ_3. Considering all the transactions together, the difference between these two games is $|Pr[S_3] - Pr[S_2]| \leq n \cdot \epsilon_3$. Considering that in practice, the number of assets that a borrower can mortgage would not be large, which means n is not a large number, thus $n \cdot \epsilon_3$ is a negligible probability.

Game 4: The process in Game 4 is almost identical as game 3. The difference is that in **BAPIM.ReqGen** algorithm of Game 4, z is a random number in group G rather than derived from the secret key sk_s.

Analysis: According to the security of threshold secret sharing scheme, the secret value is information-theoretically hidden for the adversary. Therefore, the probability that an adversary can differentiate these two games only with a negligible probability ϵ_4.

Game 5: The process in Game 5 is almost identical as Game 4. The difference is that we abort in Game 5 if the adversary forges a valid ECDSA signature.

Analysis: The distinguishable probability of these two games is that the adversary outputs a valid ECDSA signature. Since the ECDSA is EUF-CMA secure, the probability ϵ_5 of this event occurring is negligible.

In Game 5, we can see that the adversary can no longer win, which means $Pr[S_5] = 0$. Taking all games together, we have that $Pr[S_0] \leq \epsilon_1 + \epsilon_2 + n \cdot \epsilon_3 + \epsilon_4 + \epsilon_5$, where $Pr[S_0]$ represents the probability that the adversary can succeed in the original game, thus our BAPIM achieves unforgeability.

Theorem 3: The proposed BAPIM achieves accountability if the NIZKP protocol is sound.

Proof: Let $(m_1, m_2, \sigma_1, \sigma_2)$ be the output of the adversary, where $m_j = (H(A), PK_{c_j})$, $\sigma_j = (\cdot, z_j, \pi_j)$ for $j \in [2]$. Recall that $pk_b = (pk_s, pk_E, (F_i)_{i \in [n]}, crs)$ and $F_i = (F_{i,1}, F_{i,2})$. Moreover, for $j \in [2]$, $F'_{j,2} = F_{j,2} \cdot (pk_s \cdot g^{-z_j})^{\frac{1}{PK_j}}$.

Assume the adversary successfully generates two messages and their corresponding signatures with $F'_{1,2} \neq F'_{2,2}$, where the messages have the same first part but different second part. In **BAPIM.ReqGen** algorithm, the adversary needs to prove that both $(g, pk_E, F_{j,1}, F'_{j,2})$, $j \in [2]$ are valid DDH tuples. However, only one of the two tuples can be successfully verified because of the correctness of ElGamal encryption, which means the adversary cannot forge two tuples that satisfy the demands that the two messages have the same first part and different second part. Or else, these two tuples break the soundness of DDH. Let E be the event that $F'_{1,2} \neq F'_{2,2}$, we have $Pr[E] \leq 2 \cdot \epsilon_s$, where ϵ_s is the soundness error of DDH.

Since $F'_{1,2} = F'_{2,2}$, (PK_{c_1}, z_1) and (PK_{c_2}, z_2) are two valid points on the same polynomial with degree 1. Thus, sk_s can be calculated as $sk_s = z_1 \frac{PK_{c_2}}{PK_{c_2} - PK_{c_1}} + z_2 \frac{PK_{c_1}}{PK_{c_1} - PK_{c_2}}$.

Thus, the borrower's identity is calculated by invoking the decryption algorithm of ElGamal encryption scheme with the input (sk_s, D).

3.5 Performance Evaluation

In this section, we first numerically analyze the communication cost and computational cost of BAPIM and show the simulation results.

3.5.1 Numerical Analysis

Communication cost: In BAPIM, after a borrower generates its public key and private key, it needs to register with TTP. During the registration phase, the borrower sends TTP its ID, public key and the signature on them, which contains $2n + 4$ group elements. Here, n is denoted as the maximum number of the assets that the borrower can mortgage. In general, n is not a large number in practice. In BAPIM, to save the communication cost, the borrower can only send pk_s, ID and the signature to TTP. To respond to the borrower, TTP needs to return the ciphertext of the ID and the signature of the ciphertext, which contains $2n + 6$ group elements.

After the request generation process, the borrower transmits the loan request to the financial institution. The loan request includes m, σ_m, Crt_1 and Crt_2, where m contains 2 group elements and σ_m contains 3 group elements and a proof π. For the financial institution, if the loan request of the borrower is valid, the financial institution needs to sign the message M and upload M and its corresponding signature σ_M to Blockchain. If a double-mortgage behavior is detected, the financial institution needs to upload $\{M_1, M_2, \sigma_{M_1}, \sigma_{M_2}, Crt_1\}$ to the blacklist.

Computation cost: BAPIM consists of five algorithms: KGen, Register, ReqGen, ReqVerify, and Extract. Among these algorithms, KGen is a prepossessing procedure, and in general, the key generation is only performed once for all entities. In the following, MUL and EXP denote the multiplication complexity and exponentiation complexity in group G. To calculate a public-private key pair, a borrower needs to perform $2n + 2$ EXP and n MUL operations, while TTP and a financial institution only need 1 EXP operation. In the registration phase, encryption and signing are required for TTP, which would cost 1 hash, 4 EXP and 2 MUL operations. During the ReqGen phase, the borrower needs 2 hash, 4 EXP, and 5 MUL operations.

To verify the correctness of the loan requeset, the financial institution needs to perform 3 hash, 11 EXP and 11 MUL operations. If the signature of the borrower is valid, the computational cost for the financial institution to sign the message m is 2 MUL, 2 EXP operations and 1 hash. If the double-mortgage behavior of a borrower is discovered, to extract the identity of the borrower, the financial institution needs to perform 2 EXP and 4 MUL operations.

3.5.2 Simulation Results

We conduct a simulation to collect the running time of each phase of BAPIM on a notebook with an Intel(R) Core(TM) i7-7500U and CPU @ 2.9 GHz. The RAM is 8 GB. In the simulation, we employ the NIST p192 elliptic curve to generate system parameters, which is generated by Miracl library. SHA-1 is chosen as the hash function.

We first measure the performance of ElGamal encryption in terms of encryption and decryption, and evaluate the execution time of the ECDSA in terms of signing and verification. The simulation results are shown in TAB.3.2.

Table 3.2: Time cost of cryptographic algorithms

Algorithm	**Time (Unit:ms)**
Elgamal.Encrypt	0.6
Elgamal.Decrypt	0.68
ECDSA.Sign	0.7
ECDSA.Verify	0.95

Then, we evaluate the performance of the algorithms in BAPIM. Let n denote the number of assets that a borrower can mortgage. As shown in Fig.3.3, the key generation cost for the borrower increases linearly with n, since for each asset, the borrower needs to do an ElGamal encryption and generates an $F_i = (g^{r_i}, pk_E^{r_i} g^{a_i})$, where $i \in [n]$. According to the simulation results, if n is changed from 10 to 40 with an increment of 10, the time consumption of key generation for the borrower is 6.42 ms, 12.45 ms, 18.48 ms and 24.5 ms, respectively. Note that key generation can be performed offline once for each borrower, the overhead is acceptable for borrowers. For the algorithms **ReqGen** and **ReqVerify**, borrowers and financial institutions only need to do a fixed number of operations without changing with n, hence, the time cost for **ReqGen** algorithm is almost 2.68 ms, and the time consumption for **ReqVerify** is about 4.25 ms when n is changed from 10 to 40. For the identity extraction, a financial institution only needs 2 EXP operations and 2 MUL operations to extract the secret key of the malicious borrower and 1 ElGamal decryption to obtain the identity. Therefore, the time cost of **Extract** algorithm for the financial institution is small.

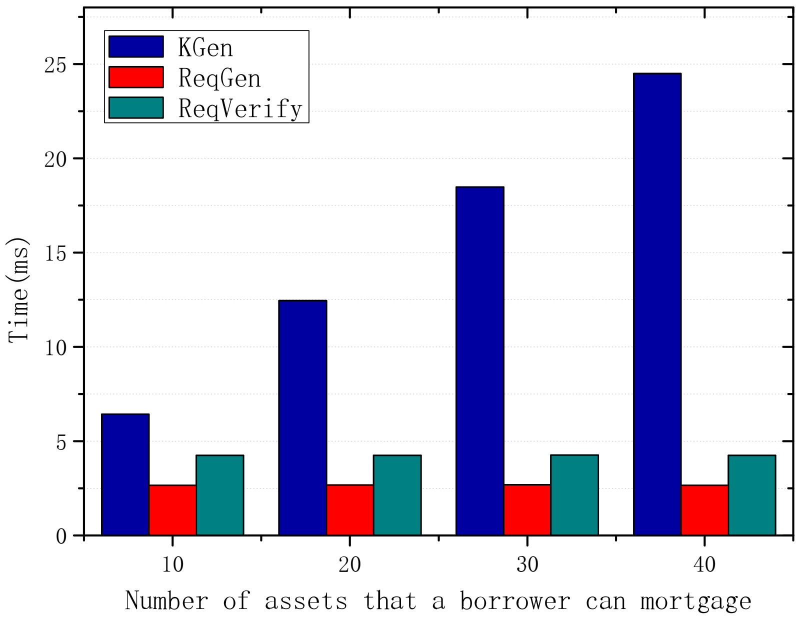


Figure 3.3: Computation cost of algorithms in BAPIM

3.6 Summary

In this chapter, we have proposed a blockchain-based accountable and privacy-preserving scheme (BAPIM) for mortgage management. BAPIM enables a financial institution to share the mortgage records with others, such that the institution can detect the double-mortgage behavior of a borrower. To protect the borrowers' privacy, anonymous authentication of borrowers is realized and other financial institutions have no knowledge about the mortgage data of honest borrowers. If the double-mortgage behavior of a borrower is detected, the secret key and the identity of the borrower would be disclosed to the corresponding financial institution. BAPIM has high security guarantees and computational efficiency, and is suitable to be implemented to support the mortgage management.

Chapter 4

Achieving Identity Privacy and Trading-Misbehavior Regulation for Blockchain-based Trading

4.1 Motivations

With the world going through the digital revolution, data have been one of the most valuable assets in our economy. Analytics of data can help companies make better decisions and facilitate their long-term success. For example, by analyzing sales data of products, one can obtain advertising insights and market strategies. Also, medical data can be collected and leveraged to train artificial intelligence models, which can be used to assist disease diagnosis and provide improved medical services. The performance of these applications highly depends on the quantity and quality of data. However, much high-quality data end up sitting idle on hard drives and servers without being used and analyzed. To promote better data utilization and circulation, one of the best approaches is data market [103], such as Dawex [104], Datacoup [105], and Xignite [106].

Fair data trading requires that a data buyer can obtain the required data if it pays the corresponding funds, and a data seller can receive the funds if it sends the data to the data buyer and the latter obtains the data. However, in data trading, there may be no trust between a data seller and a data buyer. Mistrust between them would lead to a deadlock point where a data seller is reluctant to pay first until a data seller sends the data to it, and the data seller would not provide its data until it receives the reward. The party who acts first would be at a disadvantage since the other party may be dishonest and disappear.

In the traditional data trading model, there is a data trading platform that facilitates the matching of data buyers and data sellers. However, the trading platform may be dishonest and try to steal and resell data sellers' data. In addition, it can be a target of attackers and be vulnerable to the single point of failure attack.

The decentralized nature of blockchain makes it have the potential to achieve distributed data trading without a trusted third party [30]. Blockchain suits the role because of its transparency, programmability, and the embedded cryptocurrencies [107]. Smart contracts on the blockchain can achieve automated payment when predefined conditions are satisfied. The process of blockchain-based data trading can be described as follows: A data buyer publishes the data requirements and deposits the reward of data on a smart contract. Then, a data seller demonstrates the availability of data to the data buyer. Considering that the size of data may be large and the storage of a smart contract is limited, a data seller can first encrypt the data with a key and store the ciphertexts at an off-chain storage platform, such as the InterPlanetary File System (IPFS) [108] or a cloud server. Once the data availability is verified, the data seller only needs to upload the corresponding key to the smart contract. If the key is correct, the reward is transferred to the data seller.

For blockchain-based fair data trading, there are two major challenging issues: 1) How to verify the data availability, i.e., data satisfy the requirements of a data buyer, without viewing the plaintext data. If the check is based on the plaintext, the data buyer can refuse to pay for the data even if data is indeed what it needs as it has already owned the data; 2) How to guarantee the data retrievability. Since a data seller first sends the data ciphertext to a data buyer for data availability verification, it should be guaranteed that after paying the funds, the data buyer can obtain the decryption key of the ciphertext and recover the plaintext data for the ciphertext.

For these two challenges, existing works utilize public predicate functions to validate the availability of data [73, 109, 110]. For example, a public hash value of a movie can be utilized to verify the validity of data. To demonstrate data availability, a data seller first encrypts data and proves that the encrypted data satisfy the predicate function using zero-knowledge proof. For data retrievability, a data seller reveals the decryption key to the data buyer, and if the key cannot be used to decrypt the ciphertext, the data buyer can complain to a judge smart contract by sending a concise proof of misbehavior. However, there can be no public predicate function for certain data, such as Internet of Things (IoT) data and email data. In scheme [24], an authority is introduced to authenticate the authenticity of data by signing the data using the private key of the authority. A data buyer can verify the data availability by checking the signature. In the scheme, the data retrievability is achieved by using a re-encryption scheme that can transfer a ciphertext under a data seller's public key into a ciphertext under the data buyer's public key. A re-encryption key and a zero-knowledge proof are uploaded to a smart contract to demonstrate the data retrievability.

State-of-the-art blockchain-based fair data trading schemes mainly address the fairness issue in data trading, and few of them consider the privacy issues it might cause. For example, in healthcare data trading, an Electronic Medical Record (EMR) contains much personal information, such as the name, age, and address of a patient. It is important to accomplish data trading without compromising the privacy of data sellers. For fair data trading, the data availability of an EMR can be verified by a signature signed by a hospital. With the original encrypted EMR and its signature, data authenticity is verified. However, it is a challenging task to prove data availability without sending the whole file to a data buyer, i.e., trading only part of a file while data availability can be verified. Moreover, the identity of a hospital can be used to infer the home address information of a patient. Thus, the identities of hospitals should be hidden in verifying data availability.

In this work, to address the above issues, we propose a blockchain-based fair and fine-grained data trading scheme with privacy preservation. We take healthcare data as an example. A data buyer wants to collect healthcare data from EMRs for further analysis. It can publish the data requirements for the data it needs, which include the acceptable hospitals that EMRs are issued from and data fields in EMRs that are required. For example, a data buyer needs data related to *Diabetes*, and data fields for *Blood Pressure, Insulin, Glucose* are required. For a data seller who has an EMR that satisfies the data requirements, it is desired that only the required data items are sent to the data buyer at the end. Other sensitive information, including the identities of the data seller and the hospital, is concealed from the data buyer. In our proposed scheme, for the demonstration of data availability, EMRs are signed by hospitals to guarantee the authenticity of data. By utilizing the attribute-based anonymous credential [111, 112], an authenticated data structure, and zero-knowledge proof, a data seller can prove that its EMR is signed by one of the acceptable hospitals and contains the data fields required by a data buyer, while other information is not exposed to the data buyer even after data trading. A data buyer cannot resell the data as the private key of the data seller is involved in proving the data availability. For data retrievability, an EMR is first encrypted with a symmetric key, which is then encrypted with the public key of a data seller. To enable a data buyer to decrypt the ciphertext of trading data, the data seller encrypts the symmetric key with the buyer's public key and uploads the ciphertext and a correctness proof of the ciphertext to the smart contract. With the public key of the data buyer and other auxiliary information, the proof can be verified and funds are transferred to the data seller by the smart contract. The contributions of the paper can be summarized as follows:

- We define the desired functionalities of fine-grained and fair data trading, which include fairness, privacy preservation, and fine-grained trading. We propose the first blockchain-based fair and fine-grained data trading protocol, where a data seller can sell data in part without affecting data availability verification. Privacy preservation for data sellers is achieved.

- To demonstrate the data availability, the data fields or attributes in an EMR are signed by the hospital that issues the EMR. The corresponding attribute values are encrypted separately, and the order of the attributes and their values are ensured by an authenticated data structure. Zero-knowledge proofs are used to preserve the privacy of data sellers and achieve fair data trading. Only the required data can be recovered by the data buyer, and other information in the EMR is concealed;

- We formulate a security model for blockchain-based fair and fine-grained data trading and formally prove that our scheme can achieve the desired properties. We simulate the proposed scheme, and experiment results show the practicability of our scheme. Also, the gas cost for the trading smart contract is low.

4.2 Problem Statement

4.2.1 System Model

In our system model, there are five types of entities, a health department (HD), hospitals, data sellers, data buyers, and smart contracts, as shown in Fig. 4.1.

- *HD* – HD generates public system parameters for hospitals so that they can sign EMRs with the same public parameters. The hospitals are registered with HD. After that, HD publishes the public keys of hospitals;

- *Hospital* – A hospital can generate EMRs for patients and sign EMRs with its private key. Others can verify the authenticity of data by checking the signature of an EMR. A data buyer can define its acceptable hospitals in data requirements. For data trading, the identities of the hospitals are hidden from data buyers. A hospital is only responsible for generating EMRs for patients and not involved in the data trading process;

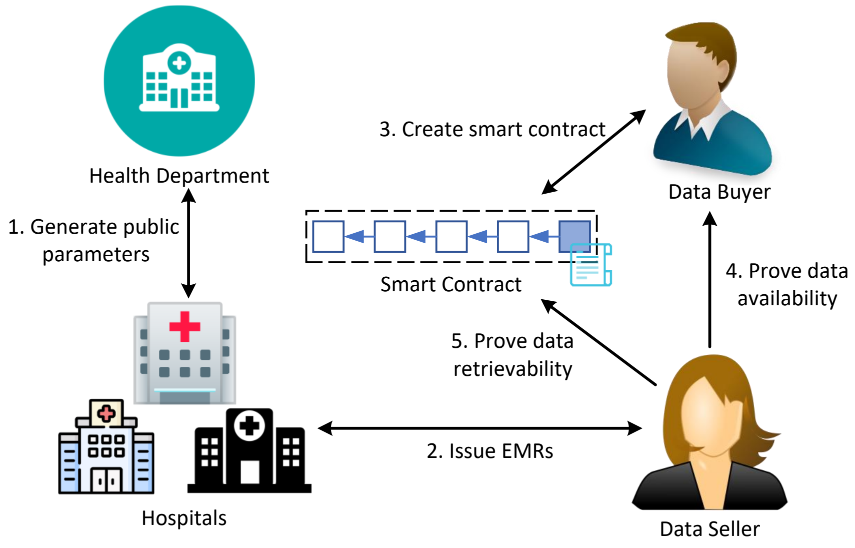


Figure 4.1: System model

- *Data seller* – A data seller is a patient that owns EMRs and wants to monetize the data. When a data seller has data that satisfy the requirements of a data buyer, it can interact with the data buyer and a smart contract deployed by the data buyer to finish the data trading. Moreover, a data seller is reluctant to leak its personal information to the data buyer;

- *Data buyer* – A data buyer is a collector of healthcare data, which can be government organizations, pharmaceutical industries, and healthcare professionals. It creates a smart contract and publishes the data requirements that include the acceptable hospitals and required data fields in EMRs. The data buyer also deposits the reward of the data on the smart contract. A data buyer pays funds only if the data provided by a data seller satisfy its requirements;

- *Smart contract* – A smart contract is a self-executing contract deployed on a blockchain that runs when predetermined conditions are verified [113]. For data trading, a smart contract is initiated by a data buyer. When certain conditions are met, the smart contract can automate the transfer of funds to guarantee the atomicity of a transaction.

There are two phases in our scheme: pre-trading and trading. In the pre-trading phase, HD generates the public parameters for the hospitals. Hospitals can issue and sign EMRs for patients. In the data trading phase, a data buyer first creates a smart contract in which data requirements are defined, which are called policies in our scheme. The data buyer also deposits the rewards of data on the smart contract. If a data seller has EMRs that satisfy the defined policy, it sends the ciphertext of data that can be decrypted by a symmetric key and a zero-knowledge proof that demonstrates data availability to the data buyer off-chain. Once data availability is verified by the data buyer, it sends a confirmation message to the smart contract. Then, the data seller uploads an encryption of the symmetric key that can be decrypted by the data buyer's private key and a zero-knowledge proof that shows the data retrievability to the smart contract. Once the correctness of the ciphertext is verified, the smart contract transfers the reward to the data seller.

4.2.2 Attack Model

In our system, an attacker can be a malicious data buyer, who obtains the desired data from a data seller and refuses to pay the rewards. Moreover, a malicious buyer tries to resell the data seller's data and get benefits. The attacker also wants to learn the real identity of a data seller and link the on-chain data trading transactions of the same data seller. On the other hand, an attacker can be a malicious data seller, who wants to send the data that do not satisfy the requirements of a data buyer and get rewards from the data buyer. The attacker also tries to learn the real identity of the data seller. Furthermore, an attacker can be an external observer of the blockchain who can access the on-chain data and wants to learn the real identities of trading participants and the trading data from the transactions stored on the blockchain.

4.2.3 Desired Goals

The design goals for blockchain-based fair and fine-grained data trading include fairness, privacy preservation, fine-grained trading, and efficiency, which are described as follows:

- Fairness – For a data seller, fairness implies that after it sends the data satisfying the requirements of a data buyer and the latter obtains the data, it can receive the rewards. A data buyer cannot resell the data seller's data to other honest data buyers. For a data buyer, before the payment, it should be able to verify the data availability. If the data buyer pays the funds, it can obtain the data it needs;
- Privacy preservation – For a data seller, only the required data are disclosed to the data buyer. Sensitive information in an EMR is not obtained by the data buyer after

data trading. A data seller should be able to prove data availability without leaking the personal information of the data seller and the hospital's identity. The smart contract does not reveal the identity of the data seller and transactions of the same data seller cannot be linked;

- Fine-grained trading – Data should be able to be traded in a fine-grained way. In data trading, a data seller can split data into many chunks and only send the data blocks that are required by a data buyer without affecting the verification of data availability.
- Efficiency – The off-chain computation, communication, and storage overhead for hospitals, data buyers, and data sellers should be small. Moreover, the execution cost or the gas fees consumed for the smart contract should be low.

4.2.4 System Components

In the pre-trading phase, there are four algorithms, which are listed as follows:

- $\mathsf{Setup}(k)$: In this algorithm, HD takes as input a security parameter k, and outputs public parameters for hospitals;
- $\mathsf{Hospital\ Key\ Generation}(pp)$: This algorithm is run by a hospital. With inputs pp, the hospital generates a public-private signing key pair (hpk, hsk) and registers itself with HD;
- $\mathsf{Issue}(hsk)$: In this algorithm, a hospital interacts with a patient to create an EMR for the patient. The patient first generates its public-private key pair (upk, usk), and submits upk and a proof that the patient knows usk to the hospital. The hospital generates an EMR for the patient based on the attributes and the patient's attribute values $(\{attri_i, D_i\}_{i\in[1,n]})$ in the EMR, where D_i denotes the attribute value for the attribute $attri_i$ and n denotes the number of attributes in an EMR;
- $\mathsf{Verify}(hpk, EMR)$: This algorithm is run by a patient. After receiving the EMR from the hospital, the patient verifies the validity of the EMR using hpk.

In the data trading phase, a data buyer interacts with a data seller, which is a patient in the pre-trading phase, and a smart contract to complete data trading. There are five algorithms in the data trading phase, which are listed below:

- Policy Definition($\{hpk\}_{i\in[1,\eta]}$): This algorithm is run by a data buyer. The data buyer first generates its public-private key pair (bpk, bsk). Then, given the acceptable hospitals whose public keys are $\{hpk\}_{i\in[1,\eta]}$, where η denotes the number of accepted hospitals, the data buyer signs $\{hpk\}_{i\in[1,\eta]}$ and obtains the signatures $(\{\sigma_i\}_{i\in[1,\eta]})$. It also defines the attribute requirements for the desired data. The output of this algorithm is the defined policy (pol). The data buyer creates a data trading smart contract SC and uploads pol to SC. It also deposits the data reward on SC;

- Policy Verification(pol, bpk): This algorithm is run by a data seller. Given pol and bpk, the data buyer verifies the validity of pol;

- Present(EMR, pol, hpk): In this algorithm, a data seller generates a zero-knowledge proof proving that it has an EMR that satisfies pol. The output of the algorithm is a token pt. The data seller sends pt and auxiliary information, including the ciphertext of the trading data with a symmetric key (k) being the secret encryption key, to the data buyer;

- Present Verify(pol, pt): This algorithm is run by a data seller. Given pol, pt, and the auxiliary information, the data seller checks the data availability of the data seller. If the requirements are satisfied, the data buyer uploads a confirmation message CM to SC;

- Ciphertext Generation(CM, bpk): Given CM and bpk, the data seller generates a ciphertext (CB) of k that can be decrypted by the data buyer's private key bsk. It also creates a zero-knowledge proof (π_b) that proves the correctness of CB. The data seller then uploads $\{CB, \pi_b\}$ to SC. After $\{CB, \pi_b\}$ are verified by the validators of the blockchain, SC transfers the reward to the data seller.

4.2.5 Security Definitions

Based on our design goal, a blockchain-based data trading scheme should satisfy two security properties: fairness and privacy preservation.

To guarantee the fairness to a data seller, it should satisfy that if a data seller sends the data to a data buyer and the buyer obtains the data, the data seller can receive the rewards. Thus, the atomicity of transactions should be satisfied. Moreover, a data buyer should not be able to resell a data seller's data, which implies that the data ownership should be verified. For the fairness to a data buyer, it should satisfy that if a data buyer transfers the funds, it should obtain the data it required instead of invalid data. Therefore, the data availability should be validated. The atomicity of transactions can be guaranteed by the automatic execution of smart contracts. For data ownership and data availability verification, it should be guaranteed that a data seller should be able to prove the ownership of an EMR, and it cannot forge an EMR with an acceptable hospital's signature, which we describe as unforgeability.

Unforgeability requires that a data seller cannot forge a valid present token if it does not receive an EMR satisfying the requirements of a data buyer. The proposed scheme should be EUF-CMA. The secure game for unforgeability, where a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ are involved, is defined as follows.

Definition 2.1 (GAME Unforgeability)

Setup: Given a security parameter k and the number of hospitals n_H in the system, where n_H can be an arbitrary integer, $\mathcal{C}$ runs Setup algorithm to obtain the public parameters pp. $\mathcal{C}$ then runs Hospital Key Generation algorithm to generate the public-private key pairs (hpk_i, hsk_i) for $i \in [1, n_H]$. $\mathcal{C}$ sets the set $\mathcal{Q}_{issue}$ and the set $\mathcal{Q}_{present}$ to be $\emptyset$. After that, $\mathcal{C}$ sends pp and $\{hpk_i\}_{i\in[1,n_H]}$ to $\mathcal{A}$.

Issue Query: $\mathcal{A}$ can submit issue queries to $\mathcal{C}$. A j-th issue query can be denoted as $(i_j, \vec{attri}_j, \vec{D}_j)$, where i_j represents a hospital with public key hpk_{i_j}, and $\{\vec{attri}_j, \vec{D}_j\}$ are attributes and attribute values in an EMR. After receiving an issue query, $\mathcal{C}$ acts as an hospital who has hsk_{i_j} and generates an EMR for $\mathcal{A}$. Then, $\mathcal{C}$ adds $\{i_j, \vec{attri}_j, \vec{D}_j\}$ to $\mathcal{Q}_{issue}$ and sends the resulting EMR to $\mathcal{A}$.

Present Query: $\mathcal{A}$ can also make present queries to $\mathcal{C}$, which can be denoted as (j, pol, EMR). With the inputs (EMR, hpk_{i_j}, pol), $\mathcal{C}$ runs Present algorithm, and sends the resulting pt and auxiliary information to $\mathcal{A}$. Then, $\mathcal{C}$ adds $\{j, pol\}$ to $\mathcal{Q}_{present}$.

Challenge Policy Definition: $\mathcal{A}$ generates an acceptable hospital set S^* and sends S^* to $\mathcal{C}$. If S^* are included in $\{hpk_i\}_{i\in[1,n_H]}$, $\mathcal{C}$ runs Policy Definition algorithm to create pol^* and returns pol^* to $\mathcal{A}$.

Output: $\mathcal{A}$ outputs a pt^* and wins if

- Present Verify $(pt^*, pol^*) = 1$ and pt^* belong to $\mathcal{A}$;
- $(pol^*) \notin \mathcal{Q}_{present}$;
- $\nexists (i_j, \vec{attri}_j) \in Q_{issue}$ such that pol^* is satisfied.

We say that a blockchain-based data trading scheme is existentially unforgeable under chosen message attack if no polynomial-time adversary can win GAME Unforgeability with a non-negligible probability.

For privacy preservation, it should be guaranteed that only the required data can be obtained by a data buyer, i.e., personal information is not leaked, which we describe as anonymity of personal data. Moreover, since the data on smart contracts are public, the identity of a data seller should be protected, and transactions of the same data seller should not be linked, which we describe as unlinkability. The definitions of anonymity and unlinkability are listed below.

Anonymity requires that a data buyer cannot obtain the identity information of a data seller. We describe this requirement using Game Anonymity, where $\mathcal{C}$ and $\mathcal{A}$ are involved.

Definition 2.2 (GAME Anonymity)

Setup: Given a security parameter k and the number of hospitals n_H in the system. $\mathcal{C}$ runs Setup algorithm to generate public parameters pp. Then, by invoking Hospital Key Generation algorithm, $\mathcal{C}$ creates public-private key pairs $(hpk_i, hsk_i)_{i\in[1,n_H]}$ for these hospitals. $\mathcal{C}$ sends pp and $\{hpk_i\}_{i\in[1,n_H]}$ to $\mathcal{A}$.

Issue queries: $\mathcal{A}$ can submit *Issue queries* that are the same as the issue queries in GAME Unforgeability. $\mathcal{C}$ responds to $\mathcal{A}$ based on the queries.

Present queries: $\mathcal{A}$ can submit *present queries* as in GAME Unforgeability. $\mathcal{C}$ returns the results to $\mathcal{A}$.

Challenge: $\mathcal{A}$ defines a data trading policy (pol), which includes the public keys of acceptable hospitals and data requirements. $\mathcal{A}$ then sends pol to $\mathcal{C}$. $\mathcal{C}$ generates two public-private key pairs $(upk_i, usk_i)_{i\in\{0,1\}}$, and sends upk_0 and upk_1 to $\mathcal{A}$. $\mathcal{C}$ chooses $b \in \{0,1\}$, and uses usk_b creating a pt_b that satisfies pol. Then $\mathcal{C}$ returns pt_b to $\mathcal{A}$.

Guess: $\mathcal{A}$ outputs a guess of b for pt_b.

$\mathcal{A}$ wins the game if b is correct. The advantage of $\mathcal{A}$ is calculated by $Pr[b' = b] - \frac{1}{2}$.

We say that a blockchain-based data trading scheme achieves anonymity if no polynomial-time adversary can win GAME Anonymity with non-negligible advantage.

Unlinkability requires that after a data seller uploads (CB, π_b) to the smart contract, no external adversary can link two data sellers together based on (CB, π_b). In the security game for Unlinkability, we let $\mathcal{A}$ outputs a public key $\{bpk\}$ for a data buyer, a policy pol, and two sets of $\{EMR_i, hpk_i\}_{i\in\{0,1\}}$ for different data sellers and they both satisfy pol. Then, $\mathcal{C}$ generates (CB, π_b) from one of the two EMRs and sends (CB, π_b) to $\mathcal{A}$. $\mathcal{A}$ needs to decide which EMR is used to derive (CB, π_b). GAME Unlinkability is defined as follows:

Definition 2.3 (GAME Unlinkability)

Setup: $\mathcal{A}$ runs Setup algorithm and outputs public parameters pp.

Issue: $\mathcal{A}$ runs Issue algorithm and generates $\{EMR_i, hpk_i\}_{i\in\{0,1\}}$ for different patients whose private key is $\{usk\}_{i\in\{0,1\}}$. $\mathcal{A}$ also defines a public key $\{bpk\}$ for a data buyer and a policy pol, where $\{hpk_i\}_{i\in\{0,1\}}$ are acceptable hospitals. $\mathcal{A}$ sends the resulting EMRs and the corresponding hospital public keys to $\mathcal{C}$.

Ciphertext Generation: $\mathcal{C}$ runs Verify to check the validity of $\{EMR_i, hpk_i\}_{i\in\{0,1\}}$. It also verifies that $\{EMR_i\}_{i\in\{0,1\}}$ satisfy pol. Then, it randomly chooses a bit $\xi \in \{0,1\}$ and generates (CB, π_b) based on EMR_ξ and bpk. $\mathcal{C}$ then sends (CB, π_b) to $\mathcal{A}$.

Output: $\mathcal{A}$ decides which EMR is used to derive (CB, π_b) and outputs ξ^*. $\mathcal{A}$ wins if $\xi^* = \xi$.

We say that a blockchain-based data trading scheme satisfies unlinkability if no polynomial-time adversary can win GAME Unlinkability with a non-negligible probability.

4.3 Proposed Blockchain-based Privacy-preserving and Fair Data Trading

In this section, we first describe the overview of the proposed scheme, then present the detailed construction.

4.3.1 Overview

There are two phases in our proposed fair and fine-grained data trading protocol: the pre-trading phase and the data trading phase. In the pre-trading phase, a hospital generates an EMR for a patient, which includes attributes of the record and the corresponding attribute values. The attributes of EMR can be the *name, age, gender*, and other attributes related to the diagnosed disease. A toy example is shown in TABLE. 4.1. The patient creates a pubic-private key pair (upk, usk) and sends upk to the hospital. The hospital acts as follows: 1) It first encrypts the attribute values separately using a symmetric key k, and then encrypts k with upk; 2) It signs the public key of the patient, attributes, and the ciphertexts of the attribute values; 3) It sends the data and the signature to the patient. By combining zero-knowledge proof and anonymous credentials, we enable the patient to selectively send partial attributes and corresponding ciphertexts of attribute values to a data buyer. Moreover, by utilizing our modified authentication data structure, which is

Table 4.1: An example of an EMR

Attributes	**Attribute Values**
Patient Name	Alice
Gender	Female
Age	50
Address	ABCDEFG
Social Insurance Number	987654321
Diastolic Blood Pressure	72
Glucose	119
Tricep Skinfold Thickness	35
Serum Insulin	140
Body Mass Index	34
A Pedigree Function for Diabetes	0.46
Diagnosis	Diabetes
Medications	*****

called Sequence-based Merkle Hash Tree (SMHT), the order consistency of attributes and their values are guaranteed.

In the data trading phase, the hospital is not involved. A data buyer first generates a one-time encryption public-private key pair (bpk, bsk). Then, the data buyer deploys a smart contract (SC) on the blockchain and deposits the reward in the SC. Moreover, the information on SC includes bpk and data requirements, which can specify the attribute set required and the attribute values of certain attributes. For example, a data buyer wants to gather data related to diabetes. As a result, the attribute value for the attribute *Disease* is *diabetes*, and attribute values for attributes *Disease, Glucose, Diastolic Blood Pressure, Serum Insulin, Age* are required [114]. If a data seller has the data that satisfy the requirements of the data buyer, the data seller and the data buyer act as follows: 1) The data seller proves to the data buyer that it has the required data. To be specific, it sends the attributes in the required attribute set (S), the signature on the attributes, the ciphertexts of attribute values, and auxiliary information to the data buyer; 2) If the data buyer is convinced that the data seller has the required data, it sends a confirmation message to the smart contract; 3) The data seller encrypts k with public key bpk, and generates a zero-knowledge proof (π_b) that demonstrates the ciphertext (CB) of k under upk and the ciphertext of k under bpk are encryptions of the same message. The data seller uploads CB and π_b to the smart contract; 4) The smart contract checks whether CB is valid by verifying π_b. If CB is valid, the smart contract records CB and sends the reward to the data seller; 5) The data buyer can decrypt CB by using its private key bsk and obtain k. Then, it uses k to decrypt the ciphertexts of the attribute values.

4.3.2 Detailed Scheme

In the pre-trading phase, HD generates public parameters for the hospitals, and a hospital and a patient interact to generate an EMR. There are four algorithms in the pre-trading phase.

Setup: In this algorithm, HD generates the pubic parameters. On input a security parameter (k) and the maximum number of attributes (n) that can be signed in an EMR, HD generates group G_1, G_2 and G_T of order p with $\{g, Y\}$ being two generators of G_1 and $\{\hat{g}, \hat{Y}\}$ being two generators of G_2. HD also randomly chooses $n+4$ generators, $Z_0, \ldots, Z_{n+3}$ of G_1. Then, HD defines three hash functions: $H : \mathbb{Z}_p^n \rightarrow G_1$, where $H(\vec{a}) = \prod_{i=0}^{n+3} Z_i^{A_i}$ and $\vec{a} = \{A_0, \ldots, A_{n+3}\} \in Z_p^{n+4}$; $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$; and $\mathsf{H} : \{0,1\}^* \rightarrow G_2$.

The public parameters pp is $pp = \{e, G_1, G_2, G_T, p, g, \hat{g}, Y, \hat{Y}, (Z_i)_{i=0}^{n+3}, H, \mathcal{H}, \mathsf{H}\}$.

Hospital Key Generation: This algorithm is run by a hospital. The hospital generates a public-private signing key pair (hpk, hsk) by randomly selecting $u \in \mathbb{Z}_p$, assigning it to hsk, and computing $hpk = \hat{U} = \hat{g}^u$.

Issue: In this algorithm, a hospital interacts with a patient to generate an EMR for the patient. Let the number of attributes in the EMR be l. For attributes $\{Attri_i\}_{i\in[1,l]}$, the corresponding attribute values are denoted as $\{D_i\}_{i\in[1,l]}$. The interaction between the patient and the hospital is as follows:

- The patient generates its public-private key pair (upk, usk) by randomly selecting $usk \in \mathbb{Z}_p^*$ and calculating $upk = {Z_0}^{usk}$;
- The patient generates a zero-knowledge proof (π_u) for usk and sends $\{\pi_u, upk\}$ to the hospital;
- The hospital verifies π_u. If π_u is valid, the hospital encrypts the attribute values in the EMR. To be specific, 1) the hospital chooses a symmetric key ($k \in \mathbb{Z}_p^*$), and encrypts k with upk using the ElGamal encryption algorithm. The ciphertext of k is $CK = \{CK_1, CK_2\}$, where $CK_1 = g^v, CK_2 = k \cdot upk^v$, and v is randomly chosen from $\mathbb{Z}_p^*$; 2) The hospital then encrypts $\{D_i\}_{i\in[1,l]}$ using the AES encryption algorithm

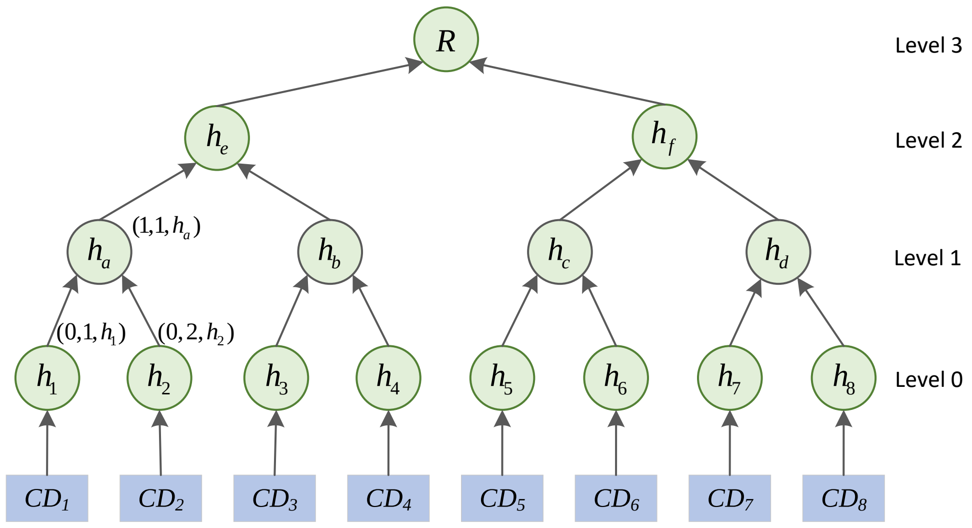


Figure 4.2: Sequence-based Merkle Hash Tree

with k being the encryption key. The ciphertext of D_i is denoted as CD_i; 3) It builds a SMHT based on $\{CD_i\}_{i\in[1,n]}$, where CD_i are the leaf nodes, and $\{CD_i\}_{i\in[l+1,n]}$ are randomly chosen from $\{0,1\}^*$. SMHT is slightly different from Merkle Hash Tree (MHT) [115], where each nodes is represented as three elements: (level, serial number, hash value), which denote the level of the node in the tree, the serial number of the node in that level, and the hash value of the node. In the tree, the hash value of an internal node is generated from its two child nodes. An example is shown in Fig. 5.1, where we denote $\mathcal{H}(CD_i)$ as h_i and they are ordered in level 0. $h_a = \mathcal{H}(h_1||h_2)$, and the node a is denoted as $(1, 1, h_a)$. The hash value of the root node of SMHT is $h_{\mathcal{R}}$; 4) The hospital also generates an SMHT based on $\{D_i\}_{i\in[1,n]}$. The resulting hash value of the root node is $h_{\hat{\mathcal{R}}}$;

- The hospital signs $\{upk, \{Attri_i\}_{i\in[1,n]}, h_{\mathcal{R}}, h_{\hat{\mathcal{R}}}\ CK\}$ as follows: The hospital first computes $A_i = \mathcal{H}(Attri_i)$ for $i \in [1,l]$. Let A_i be 0 for $i \in [l+1,n]$, $A_{n+1} = h_{\mathcal{R}}$, $A_{n+2} = h_{\hat{\mathcal{R}}}$, and $A_{n+3} = \mathcal{H}(CK_1||CK_2)$. Then, the hospital randomly chooses $r \in \mathbb{Z}_p^*$, and computes $H(\vec{a}) = upk \cdot \prod_{i=1}^{n+3} Z_i^{A_i}$, $\hat{R} = \hat{g}^r$, $S = (Y \cdot g^u)^{1/r}$, and $T = (Y^u \cdot H(\vec{a}))^{1/r}$. The resulting signature is $\sigma = (\hat{R}, S, T)$. The hospital sends EMR, which includes $\{CK, \{Attri_i, CD_i\}_{i\in[1,n]}, h_{\mathcal{R}}, h_{\hat{\mathcal{R}}}, \sigma\}$, to the patient.

Verify: In this algorithm, the patient verifies the validity of the received EMR. The patient first verifies the correctness of $h_{\mathcal{R}}$ by re-generating a SMHT based on $\{CD_i\}_{i\in[1,n]}$

and checks whether the hash value of the root is $h_{\mathcal{R}}$. If $h_{\mathcal{R}}$ is valid, the patient checks the correctness of σ. To be specific, the patient calculates $A_i = \mathcal{H}(Attri_i)$ for $i \in [1, l]$. Let $A_i = 0$ for $i \in [l+1, n]$, $A_{n+1} = h_{\mathcal{R}}$, $A_{n+2} = h_{\hat{\mathcal{R}}}$, and $A_{n+3} = \mathcal{H}(CK_1 || CK_2)$. Then, the patient checks whether $e(S, \hat{R}) = e(Y, \hat{g}) \cdot e(G, \hat{U})$ and $e(T, \hat{R}) = e(Y, \hat{U}) \cdot e(H(\vec{a}), \hat{g})$ hold. If σ is valid, the patient can decrypt CK using usk and obtain k. Then, it decrypts CD_i by utilizing k. Based on D_i, the patient builds an SMHT and checks whether $A_{n+2} = h_{\hat{\mathcal{R}}}$. If it is correct, the patient keeps the EMR locally, which is $\{CK, \{Attri_i, CD_i\}_{i \in [1,n]}, h_{(\mathcal{R})}, h_{(\hat{\mathcal{R}})}, \sigma\}$.

In the data trading phase, a data seller, which is a patient in the pre-trading phase, a data buyer, and a smart contract interact to complete data trading. A data buyer first creates a smart contract, which defines the data requirements, its public key, and the reward of data. For data requirements, the data buyer can specify which attributes and attribute values need to be disclosed and the hospitals it accepts. If a data seller has the data that satisfy the requirements, it can trade data with the data buyer in the data trading phase. This phase includes five algorithms, which are listed as follows:

Policy Definition: In this algorithm, the data buyer generates its public key and private key. Then, it defines the hospitals it accepts and the attribute requirements for the desired data. The data buyer first randomly selects $w \in \mathbb{Z}_p^*$, and calculates $W = g^w$. The public-private key pair of the data buyer is $(bpk, bsk) = (W, w)$.

Then, let the set of the public keys of the acceptable hospitals be $\mathcal{S} = \{hpk_1, \ldots, hpk_\eta\}$. For $i \in [1, \eta]$, the buyer signs the public keys hpk_i by choosing a random ξ_i and computing $R_i = g^{\xi_i}$, $\hat{S}_i = (\hat{Y} \cdot \hat{g}^w)^{\frac{1}{\xi_i}}$, $\hat{T}_i = (\hat{Y}^w \cdot hpk_i)^{\frac{1}{\xi_i}}$. The signatures $\{\sigma_i\}_{i \in [1,\eta]}$ for $\{hpk_i\}$ are $\{R_i, \hat{S}_i, \hat{T}_i\}$.

The data buyer then defines the set I to be the attributes that need to be disclosed and J to be the attributes whose attribute values need to be leaked to the data buyer before the payment. Let JV be a set that specifies the attribute values corresponding to attributes in J. For example, if the data buyer wants to collect data related to diabetes and data should include values for the attributes $\{\mathsf{Disease}, \mathsf{Age}, \mathsf{Diastolic\ Blood\ Pressure}, \mathsf{Serum\ Insulin}\}$, it specifies $\mathsf{I} = \{\mathsf{Disease}, \mathsf{Age}, \mathsf{Diastolic\ Blood\ Pressure}, \mathsf{Serum\ Insulin}\}$, $\mathsf{J} = \{\mathsf{Disease}\}$, and $\mathsf{JV} = \{\mathsf{Diabetes}\}$. By disclosure of the attributes in I and attribute values (JV) of attributes in J, the data buyer learns that the data seller has the data it needs. The data buyer then computes the hashes of I and $\{J, JV\}$ by using hashing function H and signs them.

Considering that the size of $\mathcal{S}$ may be large, the data buyer can store S and $\{\sigma_i\}_{i \in [1,\eta]}$ at an off-chain storage platform and uploads the storage position, $p, g, Y, \hat{g}, \hat{Y}, bpk, I, J, JV$, and their signatures to the smart contract. The defined policy pol is $pol = (S, I, J, JV)$. The data buyer also deposits the data reward to the smart contract.

Policy Verification: In this algorithm, a data seller verifies the validity of the policy uploaded by the data buyer and the correctness of the signatures for pubic keys in $\mathcal{S}$. Let $hpk_j = hpk$, which is the hospital that issues an EMR to the data seller. Given hpk_j and $\{R_j, \hat{S}_j, \hat{T}_j\}$, the data seller verifies whether $e(R_j, \hat{S}_j) = e(g, \hat{Y}) \cdot e(W, \hat{g})$ and $e(R_j, \hat{T}_j) = e(W, \hat{Y}) \cdot e(g, hpk_j)$. If the equations hold, the signatures for S are valid. The data seller also verifies the signature of I and $\{J, JV\}$. If they are valid, the policy defined by the data buyer is verified.

Present: In this algorithm, the data seller generates a zero-knowledge proof that proves it has an EMR that satisfies the defined policy. Denote Π as the set of indexes of the attributes in I and Ω as the set of indexes of the attributes in J.

Let j be the index of the hospital that issues the EMR to the data seller, i.e., $hpk_j = hpk$. Given the policy $pol = (S, I, J, JV)$, the data seller generates a zero-knowledge proof as follows:

- To hide the identity of the hospital, it randomizes the signature ($\sigma_j = (R_j, \hat{S}_j, \hat{T}_j)$) on the hospital. That is, it randomly selects $r'' \in \mathbb{Z}_p^*$, and computes $R'_j = R^{r''}$, $\hat{S}'_j = \hat{S}^{\frac{1}{r''}}$,$\hat{T}'_j = \hat{T}^{\frac{1}{r''}}$;
- It generates a new ciphertext for k by choosing a random $\delta \in \mathbb{Z}_p^*$ and computing $CK'_1 = CK_1^{\delta}$ and $CK'_2 = CK_2 \cdot upk^{\delta}$. Denote $CK' = \{CK'_1, CK'_2\}$. The data seller also generates a commitment of usk by choosing ϵ and calculating $B = g^{usk} Y^{\epsilon}$;
- It chooses random values $\alpha, \beta, \gamma, \phi \in \mathbb{Z}_p^*$ and computes the blinded signature (σ_{attri}) on $H(\vec{a})$ under hpk as $(\hat{R}', S', T') = (\hat{R}, S^{\frac{1}{\alpha}}, T^{\frac{1}{\beta}})$. Then, it computes the blinded public key of the hospital (hpk'_j) as $hpk_j^{\frac{1}{\gamma}}$. After that, it calculates the blinded signature (σ_{hpk}) on hpk_j under the data buyer's public key bpk as $(R'_j, \hat{S}_j{}', \hat{T}''_j) = (R'_j, \hat{S}_j{}', \hat{T}_j'^{\frac{1}{\phi}})$;
- The data seller generates a zero-knowledge proof π_d with the witness $(\alpha, \beta, \gamma, \phi, usk,$

$\{A_i\}_{i\notin\Pi}, \epsilon)$. The statement for π_d is listed as follows:

$$e(S', \hat{R}')^{\alpha} = e(Y, \hat{g}) \cdot e(g, hpk'_j)^{\gamma} \tag{4.1}$$

$$\wedge\ e(T', \hat{R}')^{\beta} = e(Y, hpk'_j)^{\gamma} \cdot e(Z_0^{usk} \prod_{i=1}^{n+3} {Z_i}^{A_i}, \hat{g}) \tag{4.2}$$

$$\wedge\ e(R'_j, \hat{S}'_j) = e(g, \hat{Y}) \cdot e(bpk, \hat{g}) \tag{4.3}$$

$$\wedge\ e(R'_j, \hat{T_j}'')^{\phi} = e(bpk, \hat{Y}) \cdot e(G, hpk'_j)^{\gamma} \tag{4.4}$$

$$\wedge\ CK'_2/CK_2 = (CK'_1/CK_1)^{usk} \tag{4.5}$$

$$\wedge\ B = g^{usk}Y^{\epsilon} \tag{4.6}$$

In the statement, equation (1) and equation (2) check the validity of the signature on $H(\vec{a})$, and equation (3) and equation (4) verify the validity of the signature on hpk. Equation (5) guarantees the consistency of the ciphertexts of k. Equations (6) demonstrates that B is a Peterson commitment [116] for usk.

The data seller then generates the auxiliary authentication information (AAI) for $\{CD_i\}_{i\in\Pi}$. The AAI for leaf node i contains the sibling nodes from i to the root node. For example, the AAI for leaf node 1 includes the nodes $\{2, b, f\}$, i.e., $\{(0, 2, h_2)$, $(1, 2, h_b)$, $(2, 2, h_f)\}$. To construct the AAI for leaf nodes $\{i\}_{i\in\Pi}$, the data seller first generates AAI for the nodes whose indexed are in Π, and the resulting set is called AAI_{Π}. Then, the data seller deletes the ancestor nodes of leaf nodes $\{i\}_{i\in\Pi}$ from AAI_{Π}. Note that from $\{CD_i\}_{i\in\Pi}$ and AAI_{Π}, we can recover $h_{\mathcal{R}}$. Similarly, the data seller generates AAI_{Ω} for $\{D_i\}_{i\in\Omega}$.

The data seller also generates a one-time public-private address for blockchain transaction and registers its public address (PA) on the smart contract. Denote pt as $pt = (\sigma_{attri}, hpk'_j, \sigma_{hpk}, \pi_d)$. The data seller sends $(pt, \{CD_i, (i, Attri_i)\}_{i\in\Pi}$, AAI_{Π}, AAI_{Ω}, $h_{\mathcal{R}}$, $h_{\hat{\mathcal{R}}}$, CK, CK', $\{D_j\}_{j\in J}$, $PA)$ to the data buyer.

Present Verify: In this algorithm, the data buyer verifies whether the data of the data seller satisfy the requirements. Given pt, and $(i, Attri_i)\}_{i\in\Pi}$, the data buyer sets $A_{n+1} = h_{\mathcal{R}}$, $A_{n+2} = h_{\hat{\mathcal{R}}}$, and $A_{n+3} = H(CK_1||CK_2)$. Then, it verifies π_d. If π_d is valid, the data buyer ensures that σ_{attri} is valid, and it is derived from a signature generated from an acceptable hospital.

Then, the data seller verifies whether $\{CD_i\}$ matches with the $\{Attri_i\}$, i.e., they have the same order in EMR. The data seller recomputes the hash value of the root ($h_{\mathcal{R}'}$) of

SMHT with $\{CD_i\}$ and AAI_Π. If $h_{\mathcal{R}} = h_{\mathcal{R}'}$, The positions of $\{CD_i\}$ are ensured. The serial numbers of $\{Attri_i\}$ are verified by checking the validity of π_d. The data seller then verifies the validity of D_i by recomputing the $h_{\hat{\mathcal{R}}'}$ based on D_i and AAI_Ω. If $h_{\hat{\mathcal{R}}} = h_{\hat{\mathcal{R}}'}$, $\{D_i\}_{i\in\Omega}$ are valid.

Then, given the ciphertext CK', the commitment B, the data buyer uploads a confirmation message $CM = \{CK', B\}$ to the smart contract.

Ciphertext Generation In this algorithm, the data seller encrypts k under the data buyer's public key bpk and generates a zero-knowledge proof that proves the ciphertext is created correctly. To be specific, the data seller randomly chooses $t \in \mathbb{Z}_p^*$, and computes $CB_1 = g^t$ and $CB_2 = k \cdot bpk^t$. Denote $CB = \{CB_1, CB_2\}$. Notice that if (CB_1, CB_2) and (CK_1', CK_2') are two encryptions of k, $CB_2/CK_2' = CK_1'^{-usk} bpk^t$. Thus, the data seller generates a zero-knowledge proof π_b, which proves that $PK : \{(usk, t, \epsilon) : CB_2/CK_2' = CK_1'^{-usk} bpk^t \wedge CB_1 = g^t \wedge B = g^{usk} Y^\epsilon\}$. The details of generating π_b are as follows:

- The data seller randomly chooses $\rho_1, \rho_2, \rho_3 \in \mathbb{Z}_p^*$, and computes $T_1 = CK_1'^{-\rho_1} \cdot bpk^{\rho_2}$, $T_2 = g^{\rho_2}$, and $T_3 = g^{\rho_1} Y^{\rho_3}$;
- The data seller computes $c_1 = \mathcal{H}(T_1||T_2||T_3)$ as a challenge;
- The data seller computes $z_1 = \rho_1 - c_1 \cdot usk$, $z_2 = \rho_2 - c_1 \cdot t$, and $z_3 = \rho_3 - c_1 \cdot \epsilon$.

The resulting π_b includes $\{T_1, T_2, T_3, c_b, z_1, z_2, z_3\}$. The data seller then uploads π_b and CB to the smart contract. Given $\{CB, CK', B, g, Y, bpk\}$, the validators of the blockchain verify whether the equation

$$c_b = H(CK_1'^{-z_1} bpk^{z_2} \cdot (CB_2/CK_2')^{c_1} || g^{z_2} CB_1^{c_1} || g^{z_1} Y^{z_3} B^{c_1})$$

holds. If π_b is valid, the smart contract transfers the data reward to the data seller.

4.4 Security Proof

In this section, we prove that the proposed scheme achieves fairness and privacy preservation.

4.4.1 Fairness

As mentioned in Section 4.2, a blockchain-based data trading scheme achieves fairness if atomicity, data ownership, and data availability are satisfied. The atomicity of transactions can be guaranteed by the smart contract. We use unforgeability to formally describe the data ownership and data availability. In the following, we prove that our scheme achieves unforgeability.

Theorem 1. If Groth_1 and Groth_2 are existential unforgeable, hash function H is collision-resistant, and the NIZK is simulation-sound extractable and zero-knowledge, our proposed scheme satisfies unforgeability.

In our scheme, an adversary can break the unforgeability by either forging a signature on a hospital hpk_j that is not included in the accepted hospital set defined in pol; Or the adversary can forge a signature on an EMR that is supposed to be signed by an honest hospital in pol, and the signature can be verified by Verify algorithm. In the following, we will prove that Theorem 1 holds.

Proof: We prove the unforgeability property by a series of games, which are listed as follows.

- $Game_0$: $Game_0$ is the same as GAME Unforgeability, where a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ are involved;
- $Game_1$: In $Game_1$, we modify the way to answer present queries, where π_d is generated by invoking a zero-knowledge and extractable NIZK simulator. Because the NIZK is zero-knowledge, $Game_1$ and $Game_0$ are indistinguishable.

In $Game_1$, $\mathcal{A}$ wins if $\mathsf{Present\ Verify}(pt^*, pol^*) = 1$. Given pt^*, we can utilize the NIZK extractor to extact the witness $(\alpha, \beta, \gamma, \phi, usk, \{A_i\}_{i \notin \Pi}, \epsilon)$. Then, with the witness, we can obtain signature σ, signature σ_j, usk, hpk, and $\{Attri\}_{i \in [1,n]}$. Let F be the event that the extractor fails to extract a witness. Let NF_1 be the event that $\mathcal{A}$ wins the game and the extractor extracts a valid witness, but the obtained hpk is not included in pol^*. Let NF_2 be the event that $\mathcal{A}$ wins the game, the extractor extracts a witness, and the recovered hpk is indeed in pol^*.

Note that $Pr[\mathcal{A}\ wins] = Pr[\mathcal{A}\ wins \wedge F] + Pr[NF_1] + Pr[NF_2] \leq Pr[F] + Pr[NF_1] + Pr[NF_2]$. Since that the Σ NIZK protocol is extractable, $Pr[F]$ is negligible. Next, we show that $Pr[NF_1]$ and $Pr[NF_2]$ are also negligible.

Case 1: In this case, $\mathcal{A}$ wins and the NIZK extractor extracts a witness, but *hpk* is not included in pol^*. In the following $Game_a$, we demonstrate that if case 1 happens, we can construct another adversary Λ that can break the existential unforgeability of Groth_2.

- Given a security parameter k, Λ generates *pp*, which is sent to $\mathcal{A}$. Λ can access a signing oracle $\mathcal{O}_{sign}$ of Groth_2. When receiving a message, $\mathcal{O}_{sign}$ can return a signature of the message;
- Λ acts as $\mathcal{C}$ in the GAME Unforgeability, except that Λ obtains the signatures of the acceptable hospitals by asking $\mathcal{O}_{sign}$. That is, Λ runs Hospital Key Generation algorithm to generate the public-private key pairs (hpk_i, hsk_i) for $i \in [1, n_H]$. Then, it answers the issue queries and present queries honestly by using (hsk_i) for $i \in [1, n_H]$;
- After receiving S^* from $\mathcal{A}$, Λ submits the singing queries of $\{(hsk_i)\}_{i \in S^*}$ for $i \in [1, n_H]$ to $\mathcal{O}_{sign}$. When obtaining the signatures $\{\sigma_i\}_{i \in S^*}$, Λ includes them in pol^*. Then, Λ returns pol^* to $\mathcal{A}$, and continues to answer issue and present queries from $\mathcal{A}$;
- Λ aborts the game if $\mathcal{A}$ outputs a valid pt^* and wins. Since one of conditions that $\mathcal{A}$ wins is that $pol^* \notin \mathcal{Q}_{present}$;
- From pt^*, Λ extracts a witness $(\alpha, \beta, \gamma, \phi,\ usk,\ \{A_i\}_{i \notin \Pi}, \epsilon)$. If the obtained hpk_j is not included in pol^*, Λ outputs hpk_j and its signature σ_j as a valid forgery of Groth_2 signature, and wins $Game_a$.

In the above game, $Pr[\Lambda \text{ wins } Game_a] = Pr[NF_1]$. Since Groth_2 is existentially unforgeable under the discrete logarithm assumption [35] and the hash function H is collision-resistent, $Pr[NF_1]$ is negligible.

Case 2: In this case, $\mathcal{A}$ wins the game, the extractor extracts a witness and the extracted *hpk* is included in pol^*.

In the following $Game_b$, we demonstrate that if case 2 happens, we can construct an adversary Γ that can break the existential unforgeability of Groth_1.

- Γ takes as input a security parameter k and generates public parameters *pp*. Then, Γ sends *pp* to $\mathcal{A}$. Γ can access a Groth_1 signing oracle $\mathcal{O}'_{sign}$, which will returns a Groth_1 signature when receiving a message to be signed;
- Γ randomly chooses τ from $[1, n_H]$, and sets hpk_τ as *hpk*. Then, Γ runs Hospital Key Generation algorithm to generate the public-private key pairs (hpk_i, hsk_i) for $i \in [1, n_H] \backslash \tau$;

- Γ answers the issue queries and present queries from $\mathcal{A}$ as follows: For issues queries, if $i_j = k$, Γ generates a signature of hpk_τ by using $\mathcal{O}'_{sign}$. For issue queries in which $i_j \neq k$, Γ runs Issue algorithm with hsk_{i_j} being an input and obtains an EMR. Then, Γ sends the resulting EMR to $\mathcal{A}$. For Present queries, Γ creates a valid pt by using the NIZK simulator;

- $\mathcal{A}$ generates an acceptable hospital set S^* and submits it to Γ. After receiving the S^*, Γ creates pol^* by running Policy Definition algorithm and returns pol^* to $\mathcal{A}$. Γ continues to answer issue and present queries as above;

- If $\mathcal{A}$ outputs a pt^* and the wining conditions in GAME Unforgeability are satisfied, Γ aborts. Γ extracts a witness from pt^*. If the obtaining $hpk = hpk_\tau$, Γ wins $Game_b$ and outputs a Groth_1 forgery, which is $\vec{attri}_\tau$ and the extracted signature on it.

  Note that if NF_2 happens and $hpk = hpk_\tau$, according to the soundness of the NIZK extractor, $\vec{attri}_\tau$ and its signature is a valid Groth_1 signature. The winning conditions of $\mathcal{A}$ guarantee that the signature is created by $\mathcal{A}$ and is a valid forgery. From $Game_b$, $Pr[\Gamma \text{ wins}] \leq Pr[NF_2 \wedge hpk = hpk_\tau] = \frac{1}{n_H} \cdot Pr[NF_2]$. Since Groth_2 is existentially unforgeable under the discrete logarithm assumption, $Pr[NF_2]$ is negligible.

4.4.2 Privacy Preservation

As discussed in Section 4.2, for privacy preservation, anonymity and unlinkability should be satisfied. In the following, we prove that our scheme achieves anonymity and unlinkability, respectively.

Theorem 2. If ElGamal encryption is indistinguishable under the chosen plaintext attacks (IND-CPA), the Peterson commitment scheme is hiding, and the NIZK is zero-knowledge, our proposed scheme satisfies anonymity.

Proof: We prove the anonymity property by a series of games, which are listed below.

- **Game 0**: This game is the same as GAME Anonymity defined in Section 4.2;

- **Game 1**: As Game 0, but we modify the Setup algorithm by using a simulator S_1 to generate the common reference string crs and a trapdoor τ for the zero-knowledge proof scheme. Furthermore, we use a simulator S_2 of the zero-knowledge proof scheme to generate π_d. We can observe that Game 1 and Game 0 are indistinguishable if the zero-knowledge proof scheme for the DDH tuple has zero-knowledge property;

- **Game 2**: As Game 1, but we modify CK_1, CK_2, CK_1', CK_2', which are results of ElGamal encryption, by choosing four random numbers in Group G. Since the ElGamal encryption is IND-CPA secure, Game 2 and Game 1 are indistinguishable;

- **Game 3**: As Game 2, but we modify the commitment B by randomly choosing a number in Group G. Since the Peterson commitment satisfies hiding property, the probability of distinguishing Game 3 and Game 2 is a negligible number.

In the Game 3, $\mathcal{A}$ can guess b correctly with probability $\frac{1}{2}$, i.e., the advantage of $\mathcal{A}$ in Game 3 is 0. To sum up, $\mathcal{A}$ has a negligible advantage in GAME Anonymity.

Theorem 3. If ElGamal encryption is indistinguishable under the chosen plaintext attacks (IND-CPA) and the NIZK is zero-knowledge, our proposed scheme satisfies unlinkability.

Proof: We prove the unlinkability property by a series of games, which are listed as follows. In the games, we define the successful event in game G_i as SE_i.

- **Game 0**: This game is the same as GAME Unlinkability defined in Section 4.2;

- **Game 1**: Game 1 is almost identical to Game 0, except that in the Ciphertext Generation algorithm, $\mathcal{C}$ replaces CB, which is the result of ElGamal encryption, with two random values in Group G. Since the ElGamal encryption is IND-CPA secure, the probability of distinguishing Game 1 and Game 0 is restricted by a negligible probability ϵ_1, i.e., $|Pr[SE_1] - Pr[SE_0]| \leq \epsilon_1$;

- **Game 2**: Game 2 is almost identical to Game 1. The difference between the two games is that for the Setup algorithm, we use a simulator S_1 to generate the common reference string crs for the zero-knowledge proof scheme as well as a trapdoor (τ). Moreover, in the Ciphertext Generation algorithm, we use a simulator S_2 to generate proof π_b using crs and τ. If the zero-knowledge proof for the DDH tuple is zero-knowledge, Game 2 and Game 1 are indistinguishable with a negligible probability ϵ_2.

Since in Game 2, the view of $\mathcal{A}$ is independent of b, the advantage for $\mathcal{A}$ in Game 2 is 0. Taking all games together, the advantage for the $\mathcal{A}$ in Game 0 is that $Pr[SE_0] \leq \epsilon_1 + \epsilon_2$, which is an negligible number. Thus, our scheme achieves unlinkability.

4.5 Performance Evaluation

In this section, we first analyze the computation, communication, and storage complexity of the proposed blockchain-based fair and fine-grained data trading scheme. We also compare the complexity of our scheme with [24]. Then, we conduct experiments to measure the performance of the proposed scheme.

4.5.1 Complexity Analysis

In our scheme, five entities are involved, which include the health department (HD), a hospital, a patient (the data seller), a data seller, and a smart contract. In the pre-trading phase, HD generates the public parameters that can be used by other entities, and the patient and the hospital interact to create an EMR for the patient. HD and the hospital do not participate in the data trading phase. In the following, we analyze the overheads of different entities in terms of computation, communication, and storage. For simplicity, we denote the exponentiation operation in G_1, G_2, and G_T as E_1, E_2, E_T. M_T denotes a multiplication in G_T, and P represents a pairing operation. We omit the computational cost of hash functions and symmetric encryption. The sizes of an element in Z_p, G_1, G_2, and G_T are represented as $|Z_p|$, $|G_1|$, $|G_2|$, $|G_T|$.

Computation overhead: In the pre-trading phase, HD generates the public parameters in the setup algorithm. Let n be the maximum number of attributes that can be signed in an EMR. The hospital creates its public-private key pair and is involved in the issuing process. The computational overhead for the hospital is at most $(l+11)E_1 + 2E_2$, where l is the number of attributes that are included in the EMR. For the patient, it needs to generate its public-private key pair and a zero-knowledge proof (π_u) for its private key. Also, after receiving the EMR, the patient verifies the signature in EMR and decrypts the ciphertext CK. The computational overhead for the patient is $3E_1 + 2M_T + 6P$.

In the trading phase, the data buyer needs to define a policy for the data trading and verify whether a data seller has data that satisfy the policy. Let the number of acceptable hospitals for the data buyer is η, the computational overhead for the data buyer is $(n+\eta+12)E_1 + (4\eta+8)E_2 + 9E_T + 8M_T + 13P$. For the data seller, it first verifies the policy published by the data buyer, which includes checking the signatures on the hospital and the roots of MHTs. Moreover, the data seller needs to prove to the data seller the availability of its data and generates a ciphertext for symmetric key k and zero-knowledge proof π_b. Let v be the number of elements in set I. In this phase, the computational overhead for the data seller is about $(n-v+23)E_1 + 5E_2 + 6E_T + 6M_T + 13P$. For the smart contract, the computation involved on-chain is the verification of proof π_b, which causes $8E_1$ computational overhead.

Communication overhead: In the pre-trading phase, the patient needs to send its public key and π_u to the hospital, which results in a $2|Z_p| + 2|G_1|$ communication overhead. For the hospital, besides the original EMR, which contains $\{Attri_i, CD_i\}_{i\in[1,n]}$, it also sends the ciphertext of k, two roots of MHTs, and signature σ to the patient. Thus, the communication overhead of the hospital is $4|G_1| + 1|G_2| + 2|Z_p|$ as well as the size of the original EMR (EMR_O).

In the data trading phase, after the data buyer defines the trading policy, it transmits the public keys of acceptable hospitals and the signatures on the public keys, set I, and $\{J, JV\}$ to an off-chain storage platform. The public key of the data buyer and signatures on set I and set J are uploaded on the smart contract. In addition, after the data buyer validates the data availability of a data seller, it needs to send a confirmation message to the smart contract. Let $|I|$ and $|J|$ be the size of data in set I and J. The communication overhead for the data buyer in this phase is $(\eta+8)|G_1|+(3\eta+4)|G_2|+1|Z_p|$, where η is the number of acceptable hospitals. For the data seller, after the policy verification, the data seller sends pt and the auxiliary data to the data buyer off-chain. When a confirmation message is uploaded to the smart contract, the data seller uploads ciphertext CB and zero-knowledge proof π_b to the smart contract. In sum, besides $\{CD_i, (i, Attri_i)\}_{i\in\Pi}$ and $\{D_j\}_{j\in J}$, the communication overhead for the data seller is about $17|G_1|+4|G_2|+2|G_T|+(2logn+13)|Z_p|$.

Storage overhead: In the pre-trading phase, HD needs to store the public parameters that include $n+6$ elements in G_1 and 2 elements in G_2, which results in a storage cost of $(n+6)|G_1|$ and $2|G_2|$. For the hospital, the storage overhead includes the public-private key pair, the original EMR, signature σ, and the key k. The storage overhead for the hospital is $3|Z_p| + 2|G_1| + 3|G_2| + |EMR_O|$. The patient needs to store its public-private key pair and the EMR, which causes $2|Z_p| + 6|G_1| + 1|G_2| + |EMR_O|$.

In the data trading phase, the data seller stores the public key of the data buyer and the signature on the hospital. The storage overhead for the data seller is $2|G_1|+2|G_2|$. For the data buyer, it saves its public-private key pair, the ciphertext of key k and required data, which is about $4|G_1|+|CD|$, where $|CD|$ is the size of the ciphertexts $\{CD_i\}_{i\in\Pi}$. For the smart contract, it maintains zero-knowledge proof π_b and elements that are useful for verifying the proof. The storage overhead for the smart contract is $5|Z_p|+13|G_1|+4|G_2|$. Table 4.2 shows the comparison of computational overhead in the data trading phase for this work and [24] in terms of the data seller, data buyer, and the smart contract.

Table 4.2: Comparison of computational overhead

Computational Cost	This work	[24]
Data seller	$(n-v+23)E_1+5E_2$ $+\,6E_T+6M_T+13P$	$16E_1+1E_2$
Data buyer	$(n+\eta+12)E_1\ +(4\eta+8)$ $E_2+9E_T+8M_T+13P$	$9E_1+1E_T$ $+2M_T+6P$
Smart contract	$8E_1$	$10E_1$

4.5.2 Simulation Results

We conduct simulations for our proposed scheme and test its performance on a notebook with Intel(R) Core(TM) i5-1135G7 @ 2.40GHz. The RAM is 16GB. We employ the Miracl library [117] to implement pairing-based cryptographic operations. We instantiate the asymmetric prime-order groups with the Barreto-Naehrig (BN) elliptic curve [118]. The security parameter is set as 128. The exponentiations for generators in G_1 and G_2 are precomputed in our simulations. We omit the computational cost for hash functions. The language for the smart contract is Solidty, and the contract can be deployed on Ethereum.

In the experiments, we test the computational cost for the hospital, the patient, and the data buyer. In the pretrading phase, the patient needs to generate its public-private key pair, generate a zero-knowledge proof for its private key, and verify the validity received from the hospital. The computational overhead for the patient is about 15.63 ms. Figure 4.3 shows the running time of the issue algorithm on the hospital side. We can see that the computational overhead for the hospital increases when the number of attributes included in the EMR grows. In particular, when an EMR contains 20 attributes, the issuing cost for the hospital is 2.55 ms.

In the trading phase, the data buyer first defines data trading policies in Policy Definition algorithm, where signatures for the acceptable hospitals' public keys are required. The data buyer also needs to verify the data availability of a data seller. The computational overhead of the data buyer is shown in Fig. 4.4(a), where the number of acceptable hospitals (η) changes from 10 to 40 and the number of attributes (n) in an EMR is 10, 20, and 30, respectively. We can see that the time cost for the data buyer rises when η and n increase. From the figure, we can observe that when there are 40 acceptable hospitals and 30 attributes in an EMR, the computational cost for the data buyer is 82.37 ms. For the data seller, the computational overhead is mainly from the generation of zero-knowledge

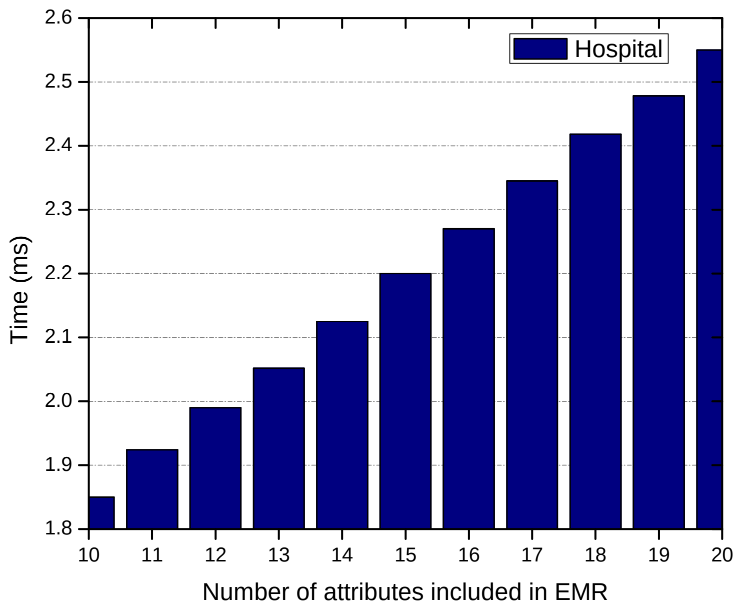
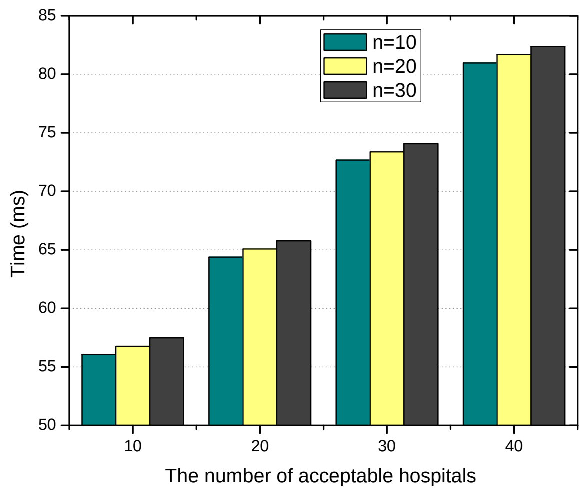


Figure 4.3: Computational cost for the hospital in the pretrading phase

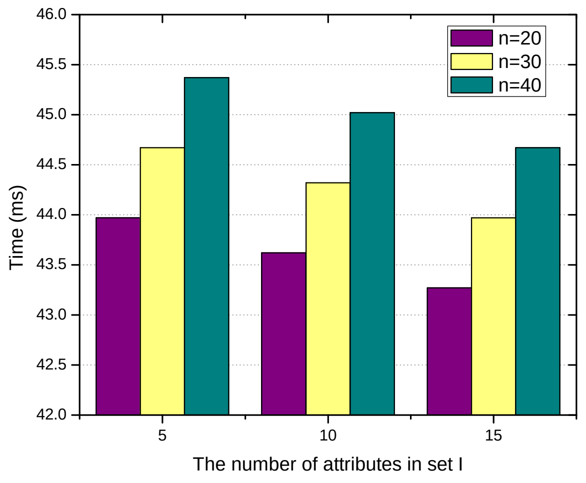
proof π_d, which is used to demonstrate that the data seller has the corresponding data that satisfy the requirements of the data buyer. The proof generation cost depends on the number of attributes (v) in set I, which is disclosed for verifying the data availability, and the number of attributes (n) in an EMR. Fig. 4.4(b) shows the computational cost for the data seller in the data trading phase. When v increases, the time cost decreases as there are fewer hidden values in the proof statement. If v is fixed, the computational cost for the data seller increases when n grows. From Fig. 4.4(b), we can see that when $v = 15$ and $n = 40$, the time cost for the data seller is 44.67 ms.

We estimate the gas cost of storing data and performing cryptographic operations in the contract. Every opcode in the Ethereum Virtual Machine (EVM) specification has an associated gas fee [119]. For example, storing one word of data (32 bytes) costs 5000 gas for a zero value and 20000 gas for a nonzero value. For the precompiled contract that implements ate pairing check on the elliptic curve alt-bn128, the gas cost for the scalar multiplication on the elliptic curve is 6000, and point addition on the curve is 150.

As shown in Fig. 4.5, in the trading phase, the data buyer first uploads the defined policy on the smart contract, which includes the data requirements. Then, the data seller registers itself on the contract. If the off-chain verification of the data availability passes, the data buyer uploads a confirmation message ($CM = (CK', B)$). After that, the data



(a) Computational cost for the data buyer



(b) Computational cost for the data seller

Figure 4.4: Computational overhead

buyer generates ciphertext CB and zero-knowledge proof π_b and sends (CB, π_b) to the smart contract, which will verify whether π_b is valid. If π_b is verified, the data buyer can retrieve CB from the blockchain and obtain the symmetric encryption key of the trading data. In Table 4.3, we summarize the storage, computation, and gas cost for policy definition, data buyer confirmation, and ciphertext verification. For the bn128 curve, Z_p is 32 bytes, G_1 is 64 bytes, and G_2 is 128 bytes. We regard the size of an attribute in set I and set J as 8 bytes. From the table, we can see that after the data buyer uploads the defined policy, the gas cost for the data buyer to send confirmation message is about 120000, and it consumes about 328900 gas for validators to verify the zero-knowledge proof π_b.

4.6 Summary

In this chapter, we have proposed a blockchain-based fair and fine-grained data trading scheme that achieves fairness and privacy preservation. In our scheme, data buyers can define data trading policies that specify the sources of data and requirements for data contents. A data seller can only send the required data to a data buyer without leaking other information. The security proof demonstrates that our scheme achieves fairness and privacy preservation. Besides fair healthcare data trading, our scheme can be utilized to

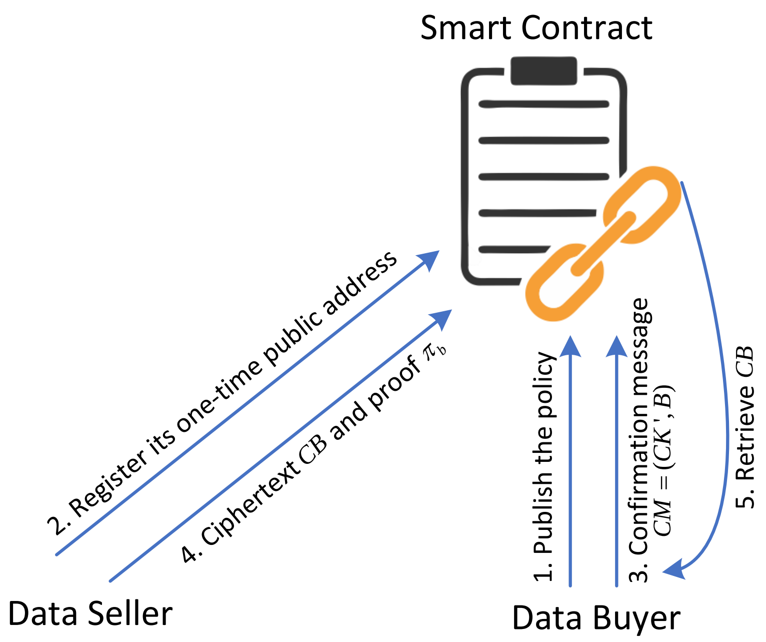

Smart Contract
2. Register its one-time public address
4. Ciphertext CB and proof π_b
1. Publish the policy
3. Confirmation message $CM = (CK', B)$
5. Retrieve CB
Data Seller
Data Buyer

Figure 4.5: Interaction between the trading participants and the smart contract

achieve flexible and practical data trading in other areas where data privacy is critical.

Table 4.3: Computational and storage overhead for the smart contract

Function	**Storage**	**Computation**	**Gas Cost**
Policy Definition	$\lvert Z_p\rvert + 5\lvert G_1\rvert + 6\lvert G_2\rvert + \lvert I\rvert + \lvert J\rvert$	NP	$70000 + 5000(n_I + n_J)$
Data Buyer Confirmation	$3\lvert G_1\rvert$	NP	120000
Ciphertext Verification	$5\lvert G_1\rvert + 4\lvert Z_p\rvert$	$6M_1 + 8E_1$	328900

$|I|+|J|$ denotes the storage cost for sets I and J. n_I and n_J denote the number of elements in sets I and J respectively.

Chapter 5

Enabling Transaction Privacy and Anti-Money Laundering Regulation for Blockchain-based Payment

5.1 Motivations

Blockchain, which can record information in a transparent and unalterable manner through the use of cryptography and decentralization, has attracted much attention from academia and industry. Due to its immutability, verifiability, and programmability, blockchain has found its applications in various areas, such as financial services [120], data storage [121], and IoT [122]. Among them, the most successful application is cryptocurrency, as the blockchain can facilitate safe and easy transactions without a trusted authority. Moreover, cryptocurrency enables rapid payments and can greatly reduce the transaction costs. It provides an innovative approach to transaction execution, and has the potential to revolutionize the financial industry and drive economic changes on a large scale. According to the CoinMarketCap [123], there have been 8795 cryptocurrencies on the market, with the market value reaching more than \$1809 billion.

Although cryptocurrencies have many attractive characteristics, transactional privacy is one of the most challenging problems when applying them in practice. The nature of blockchain makes that all transactions can be accessed by each node in the blockchain network. Thus, the sensitive information such as payers, payees, and transferred amounts can be obtained by anyone in the network. In Bitcoin system, the pseudonym mechanism is employed to disguise identities of users. However, researchers have shown that one can de-anonymize users and trace transactions by using graph analysis and address clustering [29, 30]. To improve the privacy of cryptocurrencies, DAP schemes are proposed, such as Zerocash [84] and Monero [82]. A DAP scheme enables users to pay others privately, where the addresses of transaction participants and transferred amounts are hidden from others by using cryptographic techniques, including zero-knowledge proofs and pseudorandom functions.

However, the strong degree of privacy, which enables anonymity and untraceability of transactions, creates new regulatory concerns [86, 124]. By using blockchain, traditional currency regulatory processes have been bypassed. Moreover, since cryptocurrencies are not legal tender in any jurisdiction and are not issued by a monetary authority, payers and recipients of cryptocurrencies are not under control of any third authority, causing cryptocurrency markets are largely unregulated. Thus, the convertible cryptocurrencies can be exploited to conduct illicit activities, such as money laundering and terrorist financing [125].

In traditional banking industry, there is an authority who controls the details of the users' identities and transaction amounts, making it can support financial regulation inherently. However, in distributed setting, no authority can be relied on, and due to the public visibility of the blockchain, the participants and amounts of transactions need to be protected, causing the regulation of cryptocurrencies an intractable problem. In the past years, many efforts have been made by countries to address the regulatory concerns of cryptocurrencies [25, 26]. In the US, FinCEN issued a guidance document that extends MSB of the US Bank Secrecy Act to cryptocurrencies, in which companies working with cryptocurrencies should comply with *know-your-customer* regulations, the same rules that apply to banks and other financial institutions. Moreover, for the third-party service providers, such as exchanges, payment services provides and wallet provides, licensing regimes [27] are introduced to protect customers and combat money laundering. Canada has also enacted Proceeds of Crime and Terrorist Financing Act, and transactions over CA $10,000 should be reported. To meet regulatory requirements, reputable exchanges and online wallets enforce real-name identity registration. Thus, they hold a large amount of information of their users, including the participants of transactions and the balance of their accounts, which results in little privacy for users.

Blockchain technology also provides an effective opportunity for governments to issue their own digital currencies, which is called Central Bank Digital Currency (CBDC). It uses blockchain-based token to represent a country's official currency and are regulated by monetary authority of the country. One of the benefits of CBDC is visibility to the movement of money, which will significantly improve the insight into the economy and is helpful for monetary policy making. However, since the central authority can monitor and regulate the transactions within the network, CBDC has serious privacy implications, which hinders its wide adoption. For many countries, a positive developmental environment for digital currencies, supported by regulation, is desired. Thus, a balance between the regulation and privacy of decentralized payment must be struck.

In this paper, for regulatory compliance with anti-money laundering, we introduce regulators into our system, and define three regulatory policies, which include: 1) the total amount of the cryptocurrency one can transfer in a time period is limited; 2) the frequency of transactions in a time period is restricted; 3) When suspicious transactions are detected, the identities of transaction participants can be recovered by regulators. Based on the defined policies, we propose a regulated and decentralized anonymous payment scheme, called RDAP, which can enforce the regulatory policies while preserving users' privacy. When conducting a transaction, a user needs to prove that the transaction complies with the policies, or else the user needs to include its identity information in the transaction. Note that, if the transaction is policy-compliant, identity of the user and transaction content are concealed from regulators. When illegal or suspicious transactions are discovered, regulators can recover the real identities of payers and conduct investigations on the transactions. The challenges in designing a regulated and decentralized anonymous payment scheme include: 1) How to design a mechanism that can audit the amount of transactions from one user as well as the number of transactions by the user, while the unlinkability of transactions of the same user is guaranteed. An external observer should not be able to associate a counter with a user, and the connection between an old counter and an updated counter should be hidden. 2) How to achieve the regulatory enforcement, i.e., the validity and compliance of regulation for the transactions can be verified by validators. 3) How to achieve accountability, i.e., when a transaction violates the policies, the identity of the user can be recovered. Note that a simple encryption scheme will give the regulator too much power since it can decrypt the identities of users for each transaction.

To address the challenges, we designed a transaction counter, which is represented in an algebraic form and is used to obtain the cumulative transferred amounts and number of transfers in a time period. We use ZKP to break the linkability of transactions and prove the compliance of a transaction to the regulatory policies. Both Zk-SNARKs [72] and sigma protocol-based Zero-knowledge proofs [126] are utilized and bound together to prove the validity of transactions. A tracing mechanism is embedded in our scheme to enable regulators to reveal the identities of users. The contributions of the paper are summarized as follows:

- We define the regulatory policies for cryptocurrency and propose a decentralized anonymous payment scheme supporting regulatory enforcement, where regulators are introduced into the system and they can detect and prevent suspicious transactions. To ensure both privacy protection and regulatory compliance, the total transferred amount and number of transfers can be accumulated without leaking the links between transactions, i.e. our scheme achieves anonymity and unlinkability of transactions. Moreover, the identities of users can be recovered when they violate the regulatory policies.

- Considering that there are arithmetic statements and algebraic statements that need to be proved in the scheme, we use both Zk-SNARKs and sigma protocols, which are suitable for proving arithmetic statements and algebraic statements respectively, to generate the zero-knowledge proofs for regulation compliance. Moreover, the two kinds of proofs are elaborately connected to each other. Compared with the one that only adopts Zk-SNARKs, our scheme is more efficient for users.

- We define a security model for the regulated and decentralized anonymous payment schemes, with the consideration of both regulatory compliance and privacy protection, and prove that the proposed RDAP achieves the desired security properties. Moreover, simulation results demonstrate the low computation cost for both regulators and users.

5.2 Problem Statement

5.2.1 System Model

As shown in Figure 5.1, we focus on a decentralized payment system supporting regulatory compliance and enforcement. The involved entities are described as follows:

- Regulators: To allow the regulation of the flow of cryptocurrencies, we assume in each jurisdiction, there is a regulator, which is responsible for generating pseudo-identity certificates for users and tracing users' identities for suspicious transactions. Only the regulators have the right to recover users' identities.

- Users: Users need to first register with a regulator to obtain initial counter certificates and pseudo-identity certificates, which are used for potential identity tracing. Users can be payers or recipients in the system. They can create address public keys and private keys for coin minting and transferring.

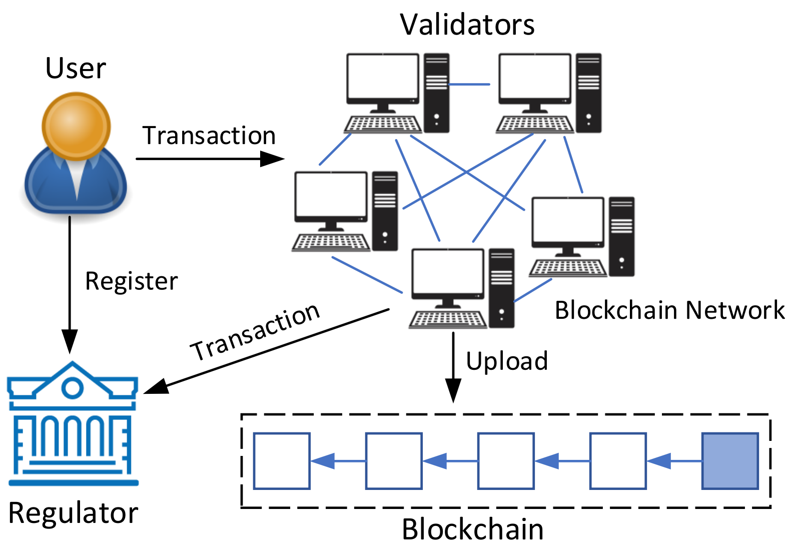


Figure 5.1: System model

- Validators: Validators are responsible for verifying the correctness of transactions and maintaining the public ledgers. If a transaction is valid and complies with the regulatory policies, they will add it on the blockchain. Otherwise, validators will send the transaction to the corresponding regulator, who can recover the user's real identity and conduct the follow-up investigation.

- Trust Authority (TA): TA is responsible for initializing the system and generating public parameters for users and validators.

Our system works as follows: Users in the system should first register with the local regulator. Validators maintain a public and append-only ledger. To spend the cryptocurrency, users should first mint coins and propose a mint transaction. The transaction is put on the ledger only when the user has paid the corresponding amount of Bitcoin or other currency to a backing escrow pool. To hide the origin of the payer, a commitment of the coin is included in the mint transaction, and the commitment is added to the list of coin commitments. When the user spends the coin, he needs to prove that the coin commitment is in the list and the serial number of the coin is exposed to prevent the double-spending. Users can transfer coins through pour transactions and spend the received coins. To preserve users' transaction privacy, a zero-knowledge proof is included in a pour transaction to prove that the transaction is valid without leaking the participants and amount transferred. Moreover, a regulatory compliance proof should be embedded in the transaction to ensure the effective regulation. When violations are detected by validators, they will send the corresponding transactions to the local regulator, who can recover the identities of the users and investigate the transactions.

5.2.2 Attack Model

In our system, the attackers can be users in the decentralized payment system and external observers of the blockchain. For transactions they are not involved in, they try to learn the identities of payers, payees, and the transferred amount. Moreover, an attacker can be a malicious user who tries to violate the defined anti-money laundering policies without being discovered. That is, for the attacker, the transferred amounts and the number of transfers in a time period exceed the predefined limits. We assume that users employ anonymous communication techniques such as Tor to transmit their transactions to validators. Attackers can eavesdrop on the traffics in the system and try to learn the details of payment transactions.

5.2.3 Security Model

For a regulated and decentralized anonymous payment system, the desired properties include anonymity, authentication, balance, and traceability. In the following, we formally define the properties by games [127] involving a challenger $\mathcal{C}$, an adversary $\mathcal{A}$, and an RDAP oracle $\mathcal{O}^{RDAP}$, where $\mathcal{A}$ is a polynomial time adversary. $\mathcal{A}$ can send $\mathcal{C}$ different types of queries, which include Register, CreateAddress, Mint, Pour, and Receive queries. After the sanity checks of the queries, $\mathcal{C}$ forwards the queries to $\mathcal{O}^{RDAP}$, which maintains a ledger and executes the queries according to the RDAP scheme, and outputs the resulting transactions. In this way, $\mathcal{A}$ can elicit the actions of honest users, and learn the public outputs. Note that $\mathcal{A}$ cannot obtain the private inputs in generating a transaction. $\mathcal{O}^{RDAP}$ can also send mint or pour transactions to $\mathcal{C}$, which is called insert queries. After the sanity check, $\mathcal{C}$ forwards the transactions to $\mathcal{O}^{RDAP}$.

GAME Anonymity: For the anonymity, as in [127], we define the property by using ledger indistinguishability, i.e., the ledger reveals no information about users other than public information on a ledger, such as the total number of transactions and transaction fees paid by a user. $\mathcal{C}$, $\mathcal{A}$, and two oracles $\mathcal{O}_0^{RDAP}$, $\mathcal{O}_1^{RDAP}$ are involved in the game.

Initialization: $\mathcal{C}$ randomly selects a bit $b \in \{0, 1\}$ and initializes $\mathcal{O}_0^{RDAP}$ and $\mathcal{O}_1^{RDAP}$. For $i \in 0, 1$, $\mathcal{O}_i^{RDAP}$ maintains a ledger L_i.

Query: $\mathcal{A}$ can submit different queries, such as Register, Mint, and Pour queries, to the two oracles. Each time, $\mathcal{A}$ submits two queries Q, Q' with the same type and identical public information to $\mathcal{C}$. $\mathcal{C}$ provides the view of two ledgers L_0, L_1 to $\mathcal{A}$ with a random order, i.e., $L_{left} = L_b, L_{right} = L_{1-b}$, where $b \in \{0, 1\}$. If the queries are Register, Mint, Pour, Receive queries, after the sanity check of the two queries, $\mathcal{C}$ sends Q to $\mathcal{O}_0^{RDAP}$, and Q' to $\mathcal{O}_1^{RDAP}$. If the queries are Insert queries, Q is sent to $\mathcal{O}_{left}^{RDAP}$, and Q' is sent to $\mathcal{O}_{right}^{RDAP}$.

Guess: After the queries, $\mathcal{A}$ needs to determine that whether the ledgers it sees are $L_{left} = L_0, L_{right} = L_1$, which means $b = 0$, or $L_{left} = L_1, L_{right} = L_0$, i.e., $b = 1$. $\mathcal{A}$ returns a bit b' to $\mathcal{C}$, which is the guess of $\mathcal{A}$.

$\mathcal{A}$ wins the game if the $b' = b$. The anonymity property requires that for a polynomial-time $\mathcal{A}$, the advantage of $\mathcal{A}$, i.e., $Pr[b' = b] - \frac{1}{2}$, is negligible.

GAME Authentication: Authentication property captures the requirement that users in the system should register with a local regulator to ensure the effective supervision of the cryptocurrency. The GAME Authentication involves $\mathcal{A}$, $\mathcal{C}$, and an oracle $\mathcal{O}^{RDAP}$.

Initialization: $\mathcal{C}$ initializes $\mathcal{O}^{RDAP}$, which maintains a ledger L.

Queries: $\mathcal{A}$ adaptively interacts with $\mathcal{O}^{RDAP}$ by sending different queries to $\mathcal{C}$, which forwards the queries to $\mathcal{O}^{RDAP}$ if the queries pass the sanity checks. $\mathcal{O}^{RDAP}$ executes the queries and provides $\mathcal{A}$ with the view of the ledger L.

Output: $\mathcal{A}$ outputs a pour transaction tx' that satisfies that: 1) the transaction is valid; 2) the payer of the transaction is not registered with the regulator.

$\mathcal{A}$ wins GAME Authentication if $\mathcal{A}$ outputs a transaction that satisfies the above two conditions. The authentication property requires that no adversary can have a non-negligible probability in winning the above game.

GAME Balance: Balance property captures the requirement that the total input value and the output value of a transaction should be equal. The GAME Balance involves $\mathcal{A}$, $\mathcal{C}$, and an oracle $\mathcal{O}^{RDAP}$.

Initialization: $\mathcal{C}$ initializes $\mathcal{O}^{RDAP}$, which maintains a ledger L.

Queries: $\mathcal{A}$ can submit queries to $\mathcal{O}^{RDAP}$, which simulates the behavior of honest users. $\mathcal{O}^{RDAP}$ executes the queries and provides $\mathcal{A}$ with the view of the ledger L.

Output: After the queries, $\mathcal{A}$ outputs a set of coins $\mathcal{S}_{coin}^{\mathcal{A}}$.

Let $v_{unspent}$ be the value of unspent coins of $\mathcal{A}$ in $\mathcal{S}_{coin}^{\mathcal{A}}$ and v_{public} be the sum of public values in the Pour transactions inserted by $\mathcal{A}$. v_{Mint} denotes the total value of coins $\mathcal{A}$ has minted. $v_{ADDR\to\mathcal{A}}$ is the total value that $\mathcal{A}$ received from honest users, and $v_{\mathcal{A}\to ADDR}$ is the total value that transferred by $\mathcal{A}$ to honest users. $\mathcal{A}$ wins GAME Balance if the total value it has spent and the remaining coins it can spend is greater than it has minted or received from others, i.e.,

$$v_{unspent} + v_{public} + v_{\mathcal{A}\to ADDR} \geq v_{mint} + v_{ADDR\to\mathcal{A}}.$$

The balance property requires that $\mathcal{A}$ can win the above game with only a negligible probability.

GAME Traceability: Traceability property captures the requirement that when a transaction violates the regulatory policies, the regulator can recover the user's real identity from the transaction. The GAME Traceability involves $\mathcal{A}$, $\mathcal{C}$, and an oracle $\mathcal{O}^{RDAP}$.

Initialization: $\mathcal{C}$ initializes $\mathcal{O}^{RDAP}$, which maintains a ledger L.

Queries: $\mathcal{A}$ can interact with the oracle $\mathcal{O}^{RDAP}$ by sending different types of queries to $\mathcal{C}$, which proxies queries to $\mathcal{O}^{RDAP}$. $\mathcal{O}^{RDAP}$ executes the queries and provides $\mathcal{A}$ with the view of the ledger L, which simulates the behavior of honest parties. When transactions violate the regulatory policies, $\mathcal{O}^{RDAP}$ can recover the identities of parties by using the Trace algorithm in RDAP. $\mathcal{A}$ can also obtain the recovered identities of parties.

Output: In this phase, $\mathcal{A}$ outputs a pour transaction tx^*.

$\mathcal{A}$ wins the game if tx^* violates the regulatory policies, whereas it pass the verification of validators, or the regulator fails to recover the identity of the user, or the recovered user does not register with the regulator. RDAP achieves traceability if $\mathcal{A}$ can win the above game with no more than a negligible probability.

5.3 Decentralized Anonymous Payment Scheme with Regulatory Compliance

5.3.1 Components of DAP schemes

A DAP scheme [84] is a decentralized electronic currency scheme that enables anonymous payments among individuals. By which, a user can mint coins and transfer coins to others without leaking the participants and the transferred amount. A DAP scheme contains six polynomial algorithms: **Setup**, **CreateAddress**, **Mint**, **Pour**, **VerifyTransaction**, and **Receive**.

Setup: This algorithm is used to generate system parameters. Given a security parameter λ, the algorithm returns public parameters pp.

CreateAddress: Users can use this algorithm to create address key pairs. Given pp, the algorithm outputs an address public and private key pair $(addr_{pk}, addr_{sk})$, which can be used to send and receive coins.

Mint: Before users transfer coins to others, they need to mint coins first. Given pp, $addr_{pk}$, and a coin value v, the algorithm returns a mint transaction tx_{Mint} and a coin $\mathbf{c}$.

Pour: This algorithm is run by users. It can be used to transfer coins from payers to payees. The inputs of the algorithm include pp, the old coins c_1^{old}, c_2^{old}, the secret address keys $addr_{sk_1}^{old}, addr_{sk_2}^{old}$, and the public address keys of payees. The outputs are two new coins c_1^{new} and c_2^{new} for the recipients and a pour transaction tr_{Pour}.

VerifyTransaction: The algorithm is run by validators. By using this algorithm, they can verify whether a pour transaction or a mint transaction is valid. The algorithm takes as input pp, a mint or a pour transaction, and a public ledger L. If the validity of the transaction is verified, the algorithm outputs a bit 1. Otherwise, it outputs 0.

Receive: Users can use this algorithm to receive coins from others. The algorithm takes as input an address key pair $(addr_{pk}, addr_{sk})$ and a public ledger L, and outputs the received coins.

5.3.2 Our Policies

Considering that cryptocurrency can be exploited for illegal transactions, regulation policies should be enforced to ensure compliance with anti-money laws and counter-terrorist financing. In many countries, there is a restriction on the amount of the cryptocurrency one can transfer at a time. For example, in the US, transactions over \$1000 should be reported to FinCEN. Exchanges such as Huobi and Binance also set a threshold for the total amount of cryptocurrency one can purchase in a day. According to existing requirements, We have similar restrictions on transactions. Moreover, in order to realize the supervision of cryptocurrency, users in the system should register with the regulators, who will issue them pseudo-random identity certificates that are used for identity tracing if needed. Taking into account users' privacy concerns, users' transaction privacy should be preserved, which means a user's identity, balance, and transferred amount should be hidden from others. Meanwhile, there should be a mechanism that can detect the policy violations. When suspicious transactions are discovered, regulators can recover the identities of users and take corresponding measures.

Thus, the regulatory policies in our decentralized anonymous payment scheme are listed as follows:

- The total amount of cryptocurrency one can transfer in a time period is limited to v_{limit}.
- The number of the transactions one can conduct in a time period is limited to n_{limit}.
- Users should register with the local regulator. When violations are detected, the identities of the corresponding users can be recovered.

Note that by adjusting the length of a period and the amount one can transfer, regulators can detect and prevent the suspicious transactions as needed.

5.3.3 Overview of RDAP

To effectively enforce the regulatory policies defined above, a counter is built for each user at each time period to accumulate the amount of cryptocurrency transferred and the number of transfers in this time period. The initial counter is signed by the local regulator. After each transaction, the counter is updated according to the amount transferred. To guarantee the unlinkability of different transactions conducted by the same user, we build a MHT $\mathcal{T}_{cID}$ for counters. When a user transfers cryptocurrency to others, the user needs to prove that the counter used in the transaction is a valid counter, which means it is a leaf node of the tree $\mathcal{T}_{cID}$, and the counter is the latest counter of the user.

To guarantee the correctness and regulation enforcement of transactions, two kinds of proofs are embedded in a pour transaction. The first proof π_1 proves that the transaction is valid, i.e., the coins to be spent belong to the user and are not spent before, the new coins generated for the receiver are well-formed, and the input value and the output value are balanced. Thus, validators can verify the validity of transactions by checking π_1. Since the Pseudorandom Function (PRF) and the commitment scheme used to construct a coin can be instantiated via SHA256 hash function, π_1 can be achieved by using the zk-SNARKs technique. On the other hand, to guarantee the enforcement of regulatory policies, i.e., the whole number of cryptocurrency and the number of transactions one can conduct in a period are limited, or else, a regulator can recover the identities of the payer, we accumulate the values one transferred and the number of transfers. This is more convenient to represent it algebraically. Thus, we construct a transaction counter, where the cumulative values are in the exponential positions. In each transaction, the counter is updated according to the transaction amount. To guarantee that the counter is updated correctly, the transaction is in accordance with the regulatory policies, and a user cannot use others' counter to avoid supervision, we designed the second zero-knowledge proof π_2. If the transaction complies with the defined policies, the proof proves that the amount and number of times transferred are within limits, otherwise, the user needs to include the pseudo-identity certificate in the transaction and proves that it belongs to him/her. With the second proof, if a transaction violates the policies, the corresponding regulator can recover the identity of the user and investigate the transaction. If the transaction is judged legal, the regulator will sign the transaction and send it to validators, who will add the transaction to the blockchain.

5.3.4 Details of RDAP

The proposed RDAP consists of seven algorithms, namely, Setup, Register, Mint, Pour, Receive, Verify, and Trace.

Setup: With the input of a security parameter λ, TA chooses an elliptic curve group G of order p, and the points of the elliptic curve are defined over a field $\mathbb{F}_t$. $g, g_1, g_2, g_3, g_4, g_5, \bar{g}, h$ are generators of group G and for each two of them, the discrete logarithm of an element with respect to the other one is unknown. Let $q > 2t^3$ be a prime number, and $\mathbb{G}$ be an elliptic curve group of order q. $\hat{g}$ and $\hat{h}$ are generators of $\mathbb{G}$, and $log_{\hat{g}}\hat{h}$ is unknown. TA also generates three pseudorandom functions PRF_x^{addr}, $\mathrm{PRF}_{\mathrm{x}}^{\mathrm{pk}}$, $\mathrm{PRF}_{\mathrm{x}}^{\mathrm{sn}}$, and two secure hash function $H : G \times Z_p \to \{0,1\}^k$ and $H_1 : \{0,1\}^* \to \{0,1\}^k$, where $k > \lambda$. Let COM and Com be two commitment schemes and CRH be a collision-resistant hash function that is used for constructing Merkle trees. The main parameters in our scheme are listed in Table 5.1.

Register: Users in a jurisdiction need to register with the corresponding regulator. A regulator $\mathcal{R}$ selects a random number σ as its master secret key, and calculates the public key $P_{pub} = g^{\sigma}$. $\mathcal{R}$ also generates an ECDSA signing key pair (ssk_R, spk_R). For a time period, $\mathcal{R}$ sets the amount of cryptocurrency one can transfer as v_{limit}, and the limit of total number of transactions one can conduct to be n_{limit}.

When a user $\mathcal{U}$ registers with $\mathcal{R}$, they interact as follows:

1. $\mathcal{U}$ chooses a random number $r_p \in Z_p$ and calculates $PID_1 = g^{r_p}$ and $R_p = g_1^{r_p}$. Then, $\mathcal{U}$ generates a proof π_r for the following statement:

$$PK\{(r_p) : PID_1 = g^{r_p} \wedge R_p = g_1^{r_p}\}.$$

$\mathcal{U}$ also creates an encryption public-private key pair (epk, esk). After that, $\mathcal{U}$ sends PID_1, R_p, π_r, epk, and its real identity RID to $\mathcal{R}$. Here, R_p is used for initial counter generation.

2. $\mathcal{R}$ chooses a $V \in Z_p$, which represents a period and is public to all validators and users. If RID and π_r are valid, $\mathcal{R}$ generates a pseudo-identity $PID = \{PID_1, PID_2\}$ for $\mathcal{U}$, where
$$PID_2 = RID \oplus H(PID_1^{\sigma}, V).$$

3. $\mathcal{R}$ randomly selects $\xi_0, \phi_0 \in Z_p$ and calculates the initial counter of $\mathcal{U}$ as $cID_0 = g_1^{r_p} g_2^V g_5^{\xi_0} h^{\phi_0}$. Note that a counter of $\mathcal{U}$ in epoch V has the form
$$cID = g_1^{r_p} g_2^V g^{\theta} g_4^{\mu} g_5^{\xi} h^{\phi},$$
where $\theta, \mu, \xi, \phi \in Z_p$. θ denotes the accumulated transferred amount in epoch V, μ denotes the accumulated number of transfers in epoch V. ξ and ϕ are used for randomization. Since this is the initial counter, the transferred amount and the number of transfers for the user are 0. Thus, the initial counter obtained by the user is $cID_0 = g_1^{r_p} g_2^V g_5^{\xi_0} h^{\phi_0}$. $\mathcal{R}$ also encrypts ξ_0, ϕ_0 with epk, and sends the ciphertext to $\mathcal{U}$.

4. $\mathcal{R}$ signs PID and cID_0 with ssk, and obtains the signature $Sig(PID)$ and $Sig(cID_0)$. $\mathcal{R}$ sends the $(PID, Sig(PID), cID_0, Sig(cID_0))$ to $\mathcal{U}$, where $(PID, Sig(PID))$ is a pseudo-identity certificate PIC for $\mathcal{U}$, and $(cID_0, Sig(cID_0))$ is the initial counter certificate for $\mathcal{U}$.

5. $\mathcal{R}$ builds a MHT tree $\mathcal{T}_{cID}$ on the $cIDs$, and sends the authentication path of cID_0 to $\mathcal{U}$. $\mathcal{R}$ also sends $(cID_0, Sig(cID_0))$ to validators, who also maintain the MHT tree $\mathcal{T}_{cID}$ on $cIDs$ for transaction verification.

Note that $\mathcal{U}$ can only register once in a period. $\mathcal{U}$ generates its address secret key and public key (a_{sk}, a_{pk}) by choosing a random seed $a_{sk} \in \{0,1\}^*$, and computes $a_{pk} = \mathrm{PRF}_{a_{sk}}^{addr}(r_p)$. $\mathcal{U}$ can generate many address keys by itself.

Mint: When $\mathcal{U}$ mints a coin **c** with a value v, it first chooses a random ρ, and calculates the serial number of the coin as $sn = \mathrm{PRF}_{a_{sk}}^{sn}(\rho)$. Then, to generate the coin, $\mathcal{U}$ computes $k = \mathrm{COM}_r(a_{pk}||\rho)$ for a random r, and $cm = \mathrm{COM}_s(v||k)$ for a random s, where cm is the coin commitment. The mint transaction tx_{Mint} is $tx_{Mint} = (cm, s, v, k)$. Moreover, $\mathcal{U}$ needs to deposit v Bitcoin to make the mint transaction to be accepted.

Validators check whether $cm = \mathrm{COM}_s(v||k)$, and if the equation holds, tx_{Mint} is considered valid and can be published on blockchain. Validators build a Merkle tree $\mathcal{T}_{coin}$ over the list of coin commitments using CRH, where leaf nodes are coin commitments of minted coins. The minted coin $\mathbf{c}$ is denoted as $\mathbf{c} = (a_{pk}, \rho, r, s, cm)$, and is kept by $\mathcal{U}$.

Pour: When $\mathcal{U}$ transfers coins to others, it needs to create a pour transaction. Without loss of generality, we assume the number of input coins and output coins is 2. The values of input coins and output coins are denoted as v_1^{old}, v_2^{old} and v_1^{new}, v_2^{new}. Similar to Zerocash, for $i \in \{1, 2\}$, $\mathcal{U}$ chooses a random number ρ_i^{new}, and generates the coin as $k_i^{new} = \mathrm{COM}_{r_i^{new}}(a_{pk_i}^{new}||\rho_i^{new})$, and $cm_i^{new} = \mathrm{COM}_{s_i^{new}}(v_i^{new}||k_i^{new})$ with a random r_i^{new}, the address public key of recipient pk_i, and a random s_i^{new}. The new generated two coins are

$$\mathbf{c}_i^{new} = (a_{pk,i}^{new}, \rho_i^{new}, r_i^{new}, s_i^{new}, v_i^{new}, cm_i^{new}).$$

$\mathcal{U}$ generates the encryption of $(\rho_i^{new}, r_i^{new}, s_i^{new}, v_i^{new})$ using the public encryption key of receiver $\mathcal{R}_i$, which is included in the transaction.

Let rt be the current root of Merkle tree $\mathcal{T}_{coin}$, and $path$ be the authentication path from a coin commitment to the root of $\mathcal{T}_{coin}$. A pour transaction includes $tx_{Pour} = (rt, sn_1^{old}, sn_2^{old}, cm_1^{new}, cm_2^{new}, v_{pub}, *)$, where sn_1^{old} is used to prevent double-spending of coins, and * denotes other auxiliary information, such as a zero-knowledge proof and the corresponding regulator of $\mathcal{U}$. v_{pub} is the value that denotes transaction fees. To guarantee the non-malleability of a pour transaction, $\mathcal{U}$ generates a signing key pair (pk_{sig}, sk_{sig}) for a one-time signature scheme. Then, $\mathcal{U}$ calculates $h_{Sig} = CRH(pk_{sig})$, and computes the values $h_1 = PRF^{pk}_{a^{old}_{sk,1}}(h_{Sig})$ and $h_2 = PRF^{pk}_{a^{old}_{sk,2}}(h_{Sig})$. $\mathcal{U}$ uses sk_{sig} to sign the pour transaction, and obtain a signature σ_{Pour}. Values h_{Sig}, h_1, h_2, σ_{Pour}, and pk_{sig} are included in tx_{Pour}.

To prove that a pour transaction tx_{Pour} is valid and complies with regulatory policies, two proofs are included in tx_{Pour}, which are π_1 and π_2. Given the witness

$$\omega = (c_1^{old}, c_2^{old}, c_1^{new}, c_2^{new}, addr_{sk,1}^{old}, addr_{sk,2}^{old}, path_1, path_2)$$

and the instance tr_{Pour}, π_1 proves the following statement:

- c_1^{old} and c_2^{old} are minted coins, i.e., $path_1$ and $path_2$ are valid authentication paths for cm_1 and cm_2.
- $\mathcal{U}$ owns c_1^{old} and c_2^{old}, i.e., $a_{pk_i}^{old} = \mathrm{PRF}^{addr}_{a^{old}_{sk,i}}(r_p)$.
- sn_1^{old} and sn_2^{old} are serial number of c_1^{old} and c_1^{old}, i.e., for $i \in \{1, 2\}$, $sn_i^{old} = \mathrm{PRF}^{sn}_{a^{old}_{sk,i}}(\rho_i^{old})$.

- all the coin commitments are well-formed, i.e., for $i \in \{1, 2\}$,

  $$cm_i^{new} = \mathrm{COM}_{s_i^{new}}(\mathrm{COM}_{r_i{}^{new}}(a_{pk,i}^{new}||\rho_i^{new})||v_i^{new}).$$

  Same as cm_i^{old}.

- The total number of input coins is equal to output values, i.e., $v_1^{old} + v_2^{old} = v_1^{new} + v_2^{new} + v_{pub}$.

- The address secret key $a_{sk,i}^{old}$ ties h_{Sig} to h_i, i.e., $h_i = PRF^{pk}_{a_{sk,i}^{old}}(h_{Sig})$.

The second proof π_2 is generated for regulatory policy enforcement. For a pour transaction, the user needs to prove the following statements in π_2:

1. The counter in the transaction belongs to the user.
2. The counter is a leaf node of the tree $\mathcal{T}_{cID}$.
3. The counter is updated correctly based on the transaction.
4. The common inputs values in π_2 and π_1 are the same.
5. If the transaction complies with the polices, the user proves that the transferred amounts and the number of transfers are within the limits in the time period. Otherwise, the user includes the pseudo-identity certificate PIC in the transaction and proves that the certificate belongs to him/her.

To be specific, $\mathcal{U}$ creates the proofs for the corresponding statements as follows:

1. $\mathcal{U}$ needs to prove that the counter in the transaction belongs to him/her. Assume the current counter is of the form $cID = g_1^{r_p} g_2^V g_3^\theta g_4^\mu g_5^\xi h^\phi$, where θ, μ, ξ and ϕ are known to $\mathcal{U}$. Let the coordinate values of cID be (cID_x, cID_y). $\mathcal{U}$ computes the commitment of cID as $C_1 = Com_q(cID_x)$, and $C_2 = Com_q(cID_y)$ [44]. Given $cID_{sn} = g_1^{r_p} g_2^V g_3^\theta g_4^\mu g_5^\xi$, which is the serial number of cID, and the commitment of cID, $\mathcal{U}$ proves that

$$\begin{aligned} PK\{(r_p, \theta, \mu, \phi, cID) :& \\ C_1 =& Com_q(cID_x) \wedge \\ C_2 =& Com_q(cID_y) \wedge \\ cID =& cID_{sn} h^\phi \wedge \\ cID_{sn} =& g_1^{r_p} g_2^V g_3^\theta g_4^\mu g_5^\xi \wedge \\ C_{r_p} =& Com_p(r_p)\}. \end{aligned} \tag{5.1}$$

The construction of the proof for this statement is shown in Appendix.

2. $\mathcal{U}$ proves that the counter is a valid counter, i.e., the counter is a leaf node of the tree $\mathcal{T}_{cID}$. Given $cID = (cID_x, cID_y)$ and the commitment $(C_1, C_2) = (Com_q(cID_x), Com_q(cID_y))$ of cID, $\mathcal{U}$ proves that cID is in the tree $\mathcal{T}_{cID}$, and the input cID is equal to cID in the commitment. Let $cID_1 = cID_x, cID_2 = cID_y$. The user utilizes a zk-SNARK scheme to prove that cID is in the tree $\mathcal{T}_{cID}$. Then, the user proves

$$\begin{aligned} PK\{(cID_1, cID_2, \eta) : y' = H^{(1/2)\eta} \prod_{i=1}^{2} G_i^{cID_i} \wedge \\ y_1 = Com_q(cID_1) \wedge \\ y_2 = Com_q(cID_2)\ \}. \end{aligned} \tag{5.2}$$

Here, H and $G_i, i \in \{1, 2\}$ are included in the CRS of the zk-SNARK (section 4.2 of [44]) and G_i corresponds to cID_i. $\eta \in Z_p$ is a random number used to achieve zero-knowledge SNARK. Note that this proof can be achieved by using non-interactive zero-knowledge proofs for composite statements [44].

3. $\mathcal{U}$ proves that the counter is updated correctly based on the transferred amount. $\mathcal{U}$ first computes $cID' = cID_{sn} \times g_3^{v^{old}} g_4 g_5^{\xi'} h^{t'}$, where $v^{old} = v_1^{old} + v_2^{old}$, and ξ', t' are random numbers in Z_p. Note that this updates the counter by adding the transferred amounts by v^{old} and the number of transfers by 1. Then, $\mathcal{U}$ proves that

$$\begin{aligned} PK\{(r_p, v^{old}, \xi', t', v_1^{old}, v_2^{old}, t_1, t_2, t_3) : \\ Com_p(v_1^{old}) = \bar{g}^{v_1^{old}} h^{t_1} \wedge \\ Com_p(v_2^{old}) = \bar{g}^{v_2^{old}} h^{t_2} \wedge \\ Com_p(v^{old}) = \bar{g}^{v^{old}} h^{t_3} \wedge \\ v^{old} = v_1^{old} + v_2^{old} \wedge \\ cID' = cID_{sn} \times g_3^{v^{old}} g_4 g_5^{\xi'} h^{t'} \}. \end{aligned} \tag{5.3}$$

4. $\mathcal{U}$ proves that the common inputs values in two zero-knowledge proofs π_1 and π_2 are the same. Given the commitments of v_1^{old}, v_2^{old}, and r_p, $\mathcal{U}$ proves that v_1^{old}, v_2^{old} and r_p used in π_2 are same as the ones used in π_1. Let $a_1 = v_1^{old}, a_2 = v_2^{old}$ and $a_3 = r_p$. It

proves

$$\begin{aligned} PK\{(a_1, a_2, a_3, \eta_1, t_1', t_2', t_r) : & \\ y' = H^{(1/2)\eta_1} \prod_{j=1}^{3} G_j^{a_j} \wedge & \\ Com_p(a_1) = \bar{g}^{a_1} h^{t_1} \wedge & \\ Com_p(a_2) = \bar{g}^{a_2} h^{t_2} \wedge & \\ Com_p(a_3) = \bar{g}^{r_p} h^{t_{r_p}}\}. & \end{aligned} \tag{5.4}$$

Here, H and $G_j, j \in \{1, 2, 3\}$ are included in the CRS of the zk-SNARK (section 4.2 of [44]) and G_j corresponds to a_j. η_1 is used to achieve zero-knowledge SNARK, and $Com_p(a_3)$ is the same as $Com_p(r_p)$ in step 1.

5. According to whether the transaction violates the regulation policies, $\mathcal{U}$ acts as follows:

   - When the transferred amounts and the number of transfers are within the limit in the time period, $\mathcal{U}$ generates the following proof: Given the current cID', the commitment of the total transferred coins $Com_p(\bar{v})$ in the period, and the commitment of the total number of transfers $Com_p(\bar{n})$ in the period, $\mathcal{U}$ proves that

     $$\begin{aligned} PK\{(\bar{v}, \bar{n}, r_p, \bar{\xi}, \bar{t}, t_{\bar{v}}, t_{\bar{n}}) : & \\ cID' = g_1^{r_p} g_2^{V} g_3^{\bar{v}} g_4^{\bar{n}} g_5^{\bar{\xi}} h^{\bar{t}} \wedge & \\ Com_p(\bar{v}) = \bar{g}^{\bar{v}} h^{t_{\bar{v}}} \wedge & \\ Com_p(\bar{n}) = \bar{g}^{\bar{n}} h^{t_{\bar{n}}} \wedge & \\ \bar{v} \in [0, v_{limit}] \wedge & \\ \bar{n} \in [0, n_{limit}]\}. & \end{aligned} \tag{5.5}$$

     The range proof of $\bar{v}$ and $\bar{n}$ can be generated using bulletproof [128].

   - When the accumulated transferred value or number of transfers exceeds the predefined limit, $\mathcal{U}$ adds the pseudo-identity certificate PIC to tx_{Pour}, and prove that

     $$PK\{r_p : PID_1 = g^{r_p}\}.$$

     The proof binds $\mathcal{U}$'s real identity to the tx_{Pour} and can be generated by using sigma protocol.

Verify: Given the public parameters, transaction tx_{Pour}, and the proof in the transaction, validators verify the correctness and regulation compliance of the transaction before it can be added to the ledger. If the total amount of transferred coins and total number of transfers are within the corresponding limits, validators add cID' in tree $\mathcal{T}_{cID}$. The cID_{sn} prevents a user from using a previous counter. For the subsequent transactions, $\mathcal{U}$ needs to prove that cID' is in the $\mathcal{T}_{cID}$ given the commitment of cID'. If the transaction violates the regulation policies, validators send the transaction to the corresponding regulator, who will investigate and deal with it accordingly.

Receive: Given the current ledger, the ciphertext sent from $\mathcal{U}$, the decryption key of $\mathcal{R}_i$, the receiver first checks whether ρ_i^{new} is different from the ρ received before. If not, $\mathcal{R}_i$ notifies $\mathcal{U}$ and requires a replacement of ρ_i^{new}, since the same ρ will result in a same serial number. Otherwise, $\mathcal{R}_i, i \in \{1, 2\}$, outputs the new coin $\mathbf{c}_i^{new} = (a_{pk,i}^{new}, \rho_i^{new}, r_i^{new}, s_i^{new}, v_i^{new}, cm_i^{new})$. The new coin can be spent by using pour transaction.

Trace: When receiving a pour transaction tx_{Pour}, a regulator parses the pseudo-identity certificate PIC as $\{PID, Sig(PID)\}$ and PID as $\{PID_1, PID_2\}$. The regulator first verifies the validity of the signature and the zero-knowledge proof of $PK\{r_p : PID_1 = g^{r_p}\}$. If it is valid, the regulator calculates the real ID of the user as $RID = PID_2 \oplus H(PID_1^{\sigma}, V)$.

5.4 Security Proof

In this section, we prove the security of the proposed RDAP, i.e., showing that the proposed RDAP satisfies the anonymity, authentication, balance, and traceability.

Theorem 1: RDAP achieves **Anonymity** if the zk-SNARK scheme is zero knowledge, the encryption scheme Enc is indistinguishable under the chosen plaintext attacks (IND-CPA), the COMM scheme is statistically-hiding, and the pseudorandom function PRF is indistinguishable from a random function.

Proof: The anonymity of our scheme can be proved by ledger indistinguishability, i.e., the ledger reveals no information about users other than the publicly-revealed information, such as the total number of transactions, public strings in transactions, and values of minted coins.

We construct a sequence of games (G_{real}, G_1, G_2, G_3, G_{sim}), in which $\mathcal{C}$ makes a modification of the original game G_{real} defined in the security model. We will show that the difference between the advantages of $\mathcal{A}$ in G_{real} and in G_{sim} is negligible.

In game G_1, after sampling $b \in \{0, 1\}$, $\mathcal{C}$ modifies the G_{real} by using a simulator $\mathcal{S}$ for simulating Key generation and Proof generation of zk-SNARKs. Instead of invoking

KeyGen(λ, C) of zk-SNARKs, $\mathcal{C}$ uses $\mathcal{S}$ to generate the proving key pk, verification key vk, and a trapdoor $trap$. C sends the public parameters to $\mathcal{A}$, and initializes two RDAP oracles $\mathcal{O}_0^{RDAP}$ and $\mathcal{O}_1^{RDAP}$. For the Prove algorithm in zk-SNARKs, $\mathcal{S}$ generates the proof π using the statement and $trap$ without utilizing the witness w. Since the zk-SNARK scheme is zero knowledge, the distributions of proofs generated by $\mathcal{S}$ and Prove algorithm are identical. Thus, the difference between the G_1 and G_{real} is zero.

In game G_2, $\mathcal{C}$ modifies G_1 by replacing the ciphertexts that are generated by using the public keys of the receivers with the encryptions of random strings. Specifically, when $\mathcal{A}$ submits a Pour query, in which the output addresses $(addr_{pk,1}^{new}, addr_{pk,2}^{new})$ are previous generated by CreatAddress queries. For each oracle, it creates the ciphertexts as follows: First, the address public keys are replaced by random generated public keys; Second, the plaintext to be encrypted is replaced by a random string chosen form the plain space. let q_P be the total number of Pour queries sent by $\mathcal{A}$. If $\mathcal{A}$ has the advantage ϵ_1 in Enc's IND-CPA experiments, the difference between game G_2 and G_1 is at most $4 \cdot q_P \cdot \epsilon_1$.

In game G_3, $\mathcal{C}$ modifies G_2 by replacing the generated PRF values with random strings. Specifically, for a CreatAddress query, the public key address returned is a random string of the same length. For the Pour query, the serial numbers sn_1^{old} and sn_2^{old} generated are also replaced by random strings of the same length. Let q_{CA} be the number of the CreatAddress queries sent by $\mathcal{A}$. Assume the advantage of $\mathcal{A}$ in distinguishing a PRF output from a random one is ϵ_2. The difference between the game G_3 and G_2 is at most $q_{CA} \cdot \epsilon_2$.

In game G_4, for the Register queries, $\mathcal{C}$ modifies G_3 by replacing the generated PID_2 with a random string of the same length. Let q_R be the number of the Register queries sent by $\mathcal{A}$. Assume the advantage of $\mathcal{A}$ in the discrete logarithm experiments is ϵ_3. The difference between the game G_4 and G_3 is at most $q_R \cdot \epsilon_3$.

In game G_{sim}, $\mathcal{C}$ modifies G_4 by replacing the generated coin commitment with the commitment of a random string. Specifically, for the Mint queries, the $v||k$ is substituted with a random input of the same length. For the Pour queries, if the output address $addr_{pk,j}^{new}$, $j \in \{1, 2\}$ belongs to the address set generated by $\mathcal{C}$, cm_j^{new} is replaced by a commitment to a random string of the same length. Let q_M be the number of Mint queries and ϵ_4 be the advantage of $\mathcal{A}$ against the hiding property of COMM. The difference between the game G_{sim} and G_4 is at most $(q_M + 4 \cdot q_P) \cdot \epsilon_4$ [84].

Since in game G_{sim}, the responses and ledgers shown to $\mathcal{A}$ are independent to b, the advantage of $\mathcal{A}$ in G_{sim} is 0. As a result, the advantage of $\mathcal{A}$ in G_{real} is

$$Adv_A \leq 4 \cdot q_P \cdot \epsilon_1 + q_{CA} \cdot \epsilon_2 + q_R \cdot \epsilon_3 + (q_M + 4 \cdot q_P) \cdot \epsilon_4$$

Theorem 2: The proposed RDAP achieves **Authentication** if the signature scheme used is existential unforgeable under the chosen message attacks (EUF-CMA) and the hash function CRH is collision-resistant.

Proof: Since the anonymity and unlinkability need to be guaranteed in the scheme, where unlinkability means the transactions that belong to the same user are unlinkable. The pseudo-identity and the address public key of a user should not be included in a transaction. Nevertheless, the counter ID cID is verified by validators in Verify process, which can ensure that the user is a legitimate user who have registered with the local regulator, and the regulator can determine its jurisdiction for a transaction.

For a cID, we consider two cases: For the initial counter cID_0, there is a signature $Sig(cID_0)$ that is signed by the regulator at the Register phase. If $\mathcal{A}$ forges a signature that can pass the verification, $\mathcal{A}$ can be invoked to attack the unforgeability of the signature scheme. For a subsequent cID, given the pubic key of the regulator, validators can check whether cID is in the tree $\mathcal{T}_{cID}$ that corresponds to the regulator. If $\mathcal{A}$ can forge the proof and pass the verification, $\mathcal{A}$ can be used to break the collision resistance of the hash function. Thus, authentication of RDAP is proved.

Theorem 3: RDAP achieves **Balance** if the commitment scheme COMM is computationally binding, the hash function CRH is collision resistant, the Zk-SNARK scheme is sound, and the PRF is indistinguishable from a random function.

Proof: To prove the balance property, we modify GAME balance defined in the security model by letting challenger $\mathcal{O}$ obtain an augmented ledger where for each pour transaction, besides the instance $x = (rt, sn_1^{old}, sn_2^{old}, cm_1^{new}, cm_2^{new}, v_{pub}, *)$, a witness $\omega = (c_1^{old}, c_2^{old}, c_1^{new}, c_2^{new}, addr_{sk,1}^{old}, addr_{sk,2}^{old},\ path_1, path_2)$ is also attached. But for $\mathcal{A}$, the view of the ledger is not changed. We denote the augmented ledger as $(L, \vec{a})$, where a_i is the witness for the i-th pour transaction instance x_i. Note that for the transactions that are generated by O^{RDAP}, $\mathcal{C}$ can obtain the witness just by asking the O^{RDAP}. For the Pour transactions that are created by $\mathcal{A}$ and inserted in the ledger, the corresponding witness can be obtained by using the knowledge extractor of the zk-SNARK.

We say that the balance property holds for a ledger $(L, \vec{a})$ if the following conditions are met.

- For a pour transaction tx_{Pour}, the distinct old coin commitments cm_1^{old} and cm_2^{old} open to two different values. Moreover, the old commitments are the outputs of mint or pour transactions that precedes tx_{Pour} on L.
- No two pour transactions contain different openings of a same coin commitment.

- For the v_1^{old}, v_2^{old} of input values and $v_1^{new}, v_2^{new}, v_{pub}$ of output values, the condition $v_1^{old} + v_2^{old} = v_1^{new} + v_2^{new} + v_{pub}$ holds.
- For each tx_{Pour} and its witness a, if cm_i^{old}, $i \in \{1, 2\}$ is the output of a mint transaction tx_{Mint}, the public value v in tx_{Mint} is equal to v_i^{old} in a. If cm_i^{old} is the output of another pour transaction tx'_{Pour}, the opening v' to cm_i^{old} is equal to v_i^{old}.
- For the pour transactions generated by $\mathcal{A}$, if cm_i^{old}, $i \in \{1, 2\}$ is the output of a mint or pour transaction tx, the public address of the receiver for tx belongs to $\mathcal{A}$, which means $\mathcal{A}$ can only spend the coins minted or received by itself.

If Ledger L is not balanced, it implies that $\mathcal{A}$ violates at least one of the above conditions with a non-negligible probability. We analyze each condition as follows and show the contradictions with the assumption.

If condition 1 does not hold, it means that $cm_1^{old} = cm_2^{old}$, or cm_i^{old} is not on the ledger. Since for a pour transaction, validators will verify that the two serial numbers are different. If $cm_1^{old} = cm_2^{old}$, the fact that $sn_1^{old} \neq sn_2^{old}$ means that there two openings of cm_1^{old}, one is derived from ρ_1^{old}, and the other is derived from ρ_2^{old}, which violates the binding property of COMM scheme. If cm_i^{old} is not on the ledger precedes tx_{Pour}, there does not exist a valid authentication path which can prove that cm_i^{old} is unspent. If the authentication path passes the verification, there exists a collision for the hash function CRH, which violates the collision resistance of CRH.

If condition 2 does not hold, it implies that there are two pour transactions tx_{Pour} and tx'_{Pour} that spend the same coin cm twice, i.e., cm can be opened to two different values, which corresponds to two serial numbers. This contradicts the binding property of COMM scheme.

If condition 3 does not hold, it means that $v_1^{old} + v_2^{old} \neq v_1^{new} + v_2^{new} + v_{pub}$, which will be checked by validators during Verify process. If the inequality holds, the soundness of the Zk-SNARK is violated.

If condition 4 does not hold, L contains a pour transaction that opens cm_{old} to a value v_{old}. But for the mint or pour transaction that corresponding to cm_{old} that precedes tx_{Pour} on L, the transaction opens cm_{old} to a value v'_{old}. This breaks the binding property of COMM scheme.

If condition 5 does not hold, it implies that $\mathcal{A}$ can spend a coin that belongs to an honest user whose $a_{pk} = PRF_{ask}^{addr}(r_p)$. Since the witness a of the pour transaction tx_{Pour} inserted by $\mathcal{A}$ contains an ask and r_p that can generate a_{pk}, the security of PRF is violated. Thus, RDAP achieved the balance property of the ledger.

Theorem 4: The proposed RDAP achieves **Traceability** if the signature scheme is EUF-CMA and the zero-knowledge proofs are sound.

Proof: If a transaction violates the defined regulatory policies, the adversary cannot generate a valid range proof, otherwise, it can be utilized to break the soundness of the range proof scheme. Therefore, a pseudo-identity certificate is included in a transaction when the transaction does not comply with policies. Since the pseudo-identity certificate is signed by the local regulator, by verifying the signature and the sigma proof which proves that the user knows r_p, the regulator ensures that the user is a legitimate user that previously registered. Based on the pseudo-identity certificate and its secret key, the regulator can recover the user's real identity. If $\mathcal{A}$ can forge the signature and pass the verification, $\mathcal{A}$ can be utilized to break the unforgeability of the underlying signature scheme. Moreover, since the initial counter cID of a user is also signed by the regulator and included in the tree $\mathcal{T}_{cID}$, and the updated cID must be calculated based on the initial counter, the user cannot get around the regulation policies by generating multiple address public-private keys.

5.5 Performance Evaluation

To evaluate the performance of RDAP, we conduct simulations and analysis on a computer with Intel(R) Core(TM) i7-7500U of 2.9 GHz and 8GB RAM. We use Barreto-Naehrig (BN) curve as the elliptic curve whose order is 256-bit length. The secure hash functions used in the scheme are SHA-256 hash functions. As in [84], we instantiate the COMM and PRF functions by using SHA-256 hash functions.

Compared with Zerocash, we add the algorithm of Register and Trace, and a zero-knowledge proof π_2 in a pour transaction, which is used for regulation compliance. A user needs to prove that the accumulated transferred value is within the predefined limit, or else, its pseudo-identity certificate needs to be involved in the transaction to help recover the identity of the user. We simulate and analyze the overhead of the added algorithms in RDAP. The details are shown as follows.

For the Register algorithm, a user needs to compute PID_1 and R_p, and a zero-knowledge proof which proves that $PK\{(r_p) : PID_1 = g^{r_p} \wedge R_p = g_1^{r_p}\}$. Moreover, the user also generates its address public-private key pair and encryption public-private key pair. The overall computation cost for the user is 5.95 ms. For the regulator, it first verifies the zero-knowledge proof, and then generates PID_2, and an initial counter cID for the user. After that, the regulator signs $PID = \{PID_1, PID_2\}$, cID, and sends the pseudo-identity

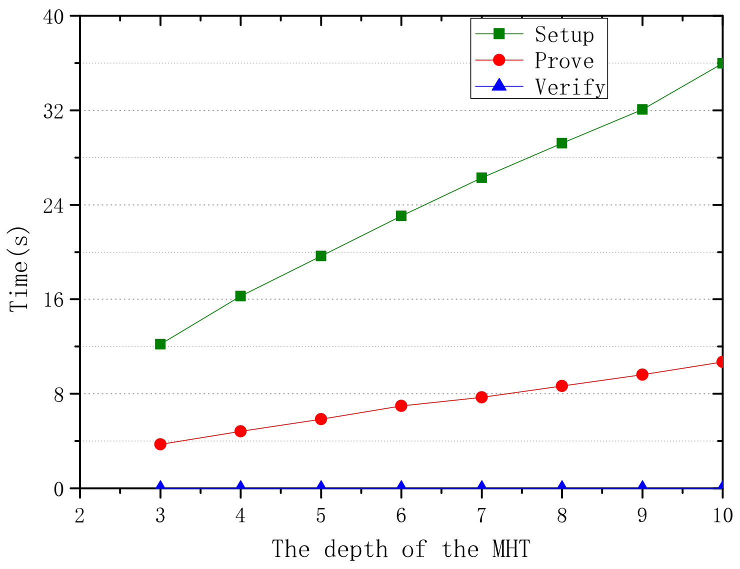


Figure 5.2: Computation cost for the MHT proof

certificate and initial counter certificate to the user. In Register phase, the time cost for the regulator is 9.51 ms.

For the Pour algorithm, we add a zero-knowledge proof π_2, which is related to regulatory policy enforcement. For π_2, we simulate the time overhead for both users and validators. Note that in step 2, the user needs to prove that cID is in the tree $\mathcal{T}_{cID}$. We simulate the corresponding MHT proof generation and verification by invoking the libsnark library [129]. We test the time cost of key generation, proof generation, and verification for the MHT proof when the number of the leaf nodes changes from 8 to 1024, i.e., the depth of the tree grows from 3 to 10. The experiment results are shown in figure 5.2, from which we can observe that the computation time grows linearly with the depth of the MHT, due to the number of nodes on the authentication path is the same as the depth of the tree. The time cost for creating the proving key and verification key for the MHT proof in π_2 is about 36 s when there are 1024 leaf nodes. As the key generation is performed only once, the time cost is acceptable.

In π_2, given all the leaf nodes and a specific cID, the user needs to generate a zk-SNARK proof. Based on the root of $\mathcal{T}_{cID}$ and the proof, a verifier can check whether a leaf node is in the tree. From figure 5.2, we can see that when there are 1024 leaf nodes, it costs 10.7 s to generate a proof. The verification for a proof is fast, and the time spent stays at a constant 0.007 s when the depth of $\mathcal{T}_{cID}$ varies. We also evaluate the storage cost of the

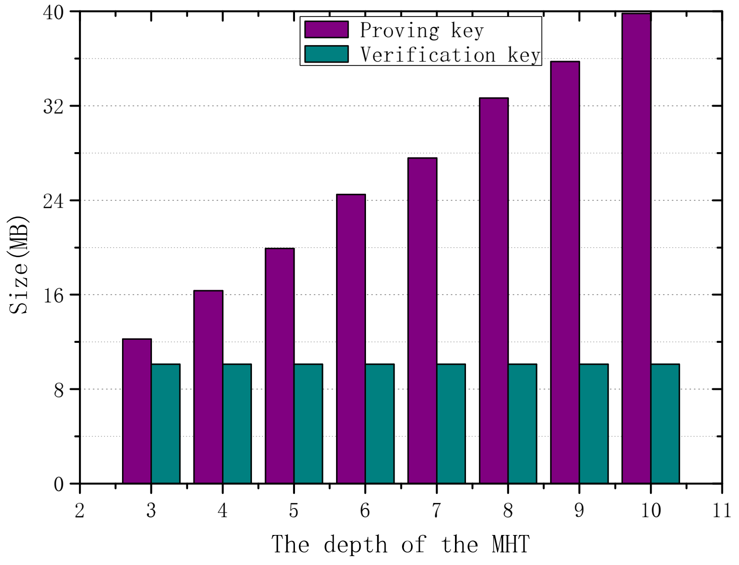


Figure 5.3: Storage cost for the MHT proof

proving key, the verification key, and the generated proof when the depth of $\mathcal{T}_{cID}$ changes from 3 to 10. Results show that the sizes of the verification key and the generated proof are fixed, which are 10.12 KB and 128 bytes, respectively. From figure 5.3, we can see that the size of the proving key increases as the depth of $\mathcal{T}_{cID}$ grows. When the depth of the tree is 10, the size of the generated proving key is 39.8 MB. For the sigma proof in π_2, we analyze the time overhead of different sigma proofs based on the cryptographic operations involved. For the third statement in π_2, the time cost for a user to generate the sigma proof is about 7.88 ms, and a validator needs about 14.03 ms to verify the proof. For the fourth statement in π_2, it takes 8.76 ms for a user to create the corresponding sigma proof, and the time cost for a validator to verify the sigma proof is about 12.28 ms.

For the Trace algorithm, a regulator first verifies the validity of the signature and the proof of $PK\{(r_p) : PID_1 = g^{r_p}\}$, which costs 2.7 ms. If they are valid, it takes 3.18 ms for the regulator to obtain the real identity of the user. Hence, the whole cost for the regulator is 5.88 ms in the Trace phase.

From the above experiment results, we can see that the designed RDAP can achieve efficient proof generation and verification for regulatory enforcement. Moreover, when suspicious transactions are discovered, only small overheads are required for regulators to recover users' identities. The simulation code can be found at https://github.com/l34xue/experiment.

5.6 Summary

In this chapter, we have proposed a regulated and decentralized anonymous payment scheme (RDAP) where regulatory policies can be enforced and users' privacy can be protected. Payers and payees of transactions and the transferred amounts are hidden from others, while a user's total amount of transferable cryptocurrency and the number of transactions in a time period are limited. The validity and regulatory compliance of transactions are guaranteed by zero-knowledge proofs. RDAP achieves anonymity, authentication, balance, traceability, and may shed light on the further research on decentralized anonymous payment with regulatory compliance.

Table 5.1: Parameters in the system

Acronym	**Definition**
$\mathbf{c}$	Coin
v	Coin value
v_{pub}	Transaction fee
G	Elliptic curve group of order p
$g, g_1, g_2, g_3, g_4, g_5, \bar{g}, h$	Generators of group G
$\mathbb{G}$	Elliptic curve group of order q
$\hat{g}, \hat{h}$	Generators of group $\mathbb{G}$
σ	Master secret key of the regulator R
P_{pub}	g^{σ}, public key of R
spk_R, ssk_R	Signing key pair of R
r_p	A random number associated with a user
$PID = \{PID_1, PID_2\}$	Pseudo-identity of a user
RID	Real identity of a user
V	A time period
cID	Transaction counter of a user
cID_{sn}	Serial number of cID
θ	Accumulated transferred amount
μ	Accumulated number of transfers
$\mathcal{T}_{cID}$	MHT built based on $cIDs$
a_{pk}, a_{sk}	Address public key and private key
sn	Serial number of a coin
cm	Coin commitment
$\mathcal{T}_{coin}$	MHT built based on coin commitments
s, r	Random numbers used to construct a coin
ρ	Random number used to construct a serial number
pk_{sig}, sk_{sig}	One time signing key pair
v_1^{old}, v_2^{old}	Input values of a pour transaction
v_1^{new}, v_2^{new}	Output values of a pour transaction

Chapter 6

Conclusions and Future Works

In this thesis, we have investigated privacy and regulation in blockchain-based financial services, and a suite of privacy-preserving and accountable mechanisms have been proposed for blockchain-based lending, data trading, and anonymous payment. In the following, we summarize the main contributions of this thesis and discuss the future research directions.

6.1 Conclusions

The contributions of this thesis can be summarized as follows:

- To achieve privacy preservation and accountability in blockchain-based lending, we have proposed a privacy-preserving blockchain-based mortgage data management scheme, which enables a financial institution to share mortgage records with others in a manner that the privacy of honest users is preserved and double-mortgage fraud can be detected. Based on hash functions, ElGamal encryption, and zero-knowledge proof, the identity and asset privacy of borrowers are protected. We utilize a hash value to denote an asset, and part of identity information is attached to a mortgage request. When multiple pledges of the same asset are detected, a financial institution can recover the identity of the malicious borrower by reconstructing the shared secret, which is the private key of the borrower. With the private key, one can further obtain the real identity of the borrower. Moreover, to guarantee that the identity information included in a mortgage request is valid, verifiable secret sharing is incorporated to ensure that the secret share is correctly generated based on the borrower's private key. As a result, when the double-mortgage fraud is detected, a financial institution can reveal the identity of the borrower. The provided security proof and performance evaluation demonstrate that the scheme achieves all the desired security properties with a low computational overhead.

- To enable fairness and privacy preservation in data trading, we have proposed a blockchain-based fair and fine-grained data trading scheme, where a data buyer can declare data requirements and acceptable data issuers, and a data seller can conduct privacy-preserving and fine-grained data selling. The data requirements specify the particular data fields or attributes that are required. To solve the issue of data availability verification without leaking the data issuer and the original data of a data seller, data are divided into chunks, and the chunks are encrypted with a secret key separately. Then, an authenticated data structure is built based on the resulting ciphertexts. By utilizing anonymous attribute-based credentials and zero-knowledge proof, a data seller can prove that the ciphertext data satisfy the requirements of a data buyer. With the authenticated data structure of ciphertexts and a signature on its root, a data seller can trade data in part without affecting data availability verification. By re-encrypting the secret key using the data buyer's public key and proving the validity of the ciphertext, it is ensured that the data buyer can obtain the data at the end. The security proof demonstrates that our scheme achieves fairness and privacy preservation. Besides the application scenario shown in the scheme, the proposed scheme can be utilized to achieve flexible and practical data trading in other areas where data privacy is critical.

- To solve the regulatory enforcement issue in decentralized anonymous payment systems, we have proposed a novel regulated anonymous payment scheme, which can enforce the regulatory policies while preserving users' privacy. To achieve regulatory compliance of anti-money laundering, we introduce regulators into our system, and three regulatory policies are defined, which include: 1) the total amount of the cryptocurrency one can transfer in a time period is limited; 2) the frequency of transactions in a time period is restricted; and 3) when suspicious transactions are detected, the identities of transaction participants can be recovered by regulators. When conducting a transaction, a user needs to prove that the transaction complies with the policies, or else the user needs to include its identity information in the transaction. Note that if the transaction is policy-compliant, the identity of the user and the transaction contents are concealed from regulators. When illegal or suspicious transactions are discovered, regulators can recover the real identities of payers and conduct investigations on the transactions. In our scheme, the total transferred amount and number of transfers can be accumulated without leaking the links between transactions, i.e., our scheme achieves anonymity and unlinkability of transactions. Both Zk-SNARKs and sigma protocol-based Zero-knowledge proofs are utilized and bound together to prove the validity of transactions. A tracing mechanism is designed in our scheme to enable regulators to reveal the identities of users. We prove the security of the proposed regulated and decentralized anonymous payment scheme, and simulation results demonstrate the low computation cost for both regulators and users.

6.2 Future Research Directions

This thesis identifies the challenges in achieving both privacy preservation and regulation for blockchain-based financial services, and proposes several promising solutions to resolve the contradiction in blockchain-based lending, data trading, and anonymous payment. Although some preliminary results are provided in privacy-preserving blockchain-based financial services, there are still several open issues that need to be addressed including but not limited to the followings.

6.2.1 Auditability and Accountability for Exchanges

In blockchain-based payment systems, many users escrow their cryptocurrencies on centralized exchanges such as Coinbase and Binance. Since users' private keys as well as their payment system accounts are controlled by a specific exchange, and the internal operations of the exchange are opaque to users, the exchange can transfer users' funds without being discovered [130]. For example, an exchange can utilize the funds for investment and thus becomes insolvent. Therefore, an exchange needs to demonstrate solvency to an authority. Meanwhile, minimal information disclosure should be satisfied. That is, apart from the fact, other information should not be leaked, including the number of users, total assets of the exchange, and account addresses belonging to the exchange. The solvency of an exchange refers to that the exchange indeed owns the funds stored by its users [91].

An exchange can prove its solvency from two aspects: one is Proof of Assets (PoA) [131], and the other one is Proof of Liabilities (PoL) [92]. When the assets of an exchange exceed the liabilities, we say that its solvency is demonstrated. For PoA, an exchange only needs to prove that it controls enough funds without showing the exact amount. In general, a lower bound of the total assets is sufficient. A simple way to prove PoA for a payment system is to provide signatures for a set of accounts owned by the exchange, and by adding up the funds in these accounts, one can obtain the minimum funds it has. However, for the privacy-preserving payment systems, where account addresses and balances of users are concealed from others, advanced cryptographic techniques are needed to prove the PoA of an exchange, such as homomorphic commitment schemes and zero-knowledge proof. For PoL, an exchange needs to disclose its liabilities, which is the sum of users' balances. When proving PoL, other information related to users should not be leaked, such as the identities of users. Note that for PoA, there may exist collusion among different exchanges to deceive the auditor. For the demonstration of PoL, when transaction privacy is protected, the exchange may attempt not to include liabilities of a fraction of users, which should be prevented. Another issue needed to be addressed is how to solve a dispute between an auditor and an auditee. A dishonest auditor can accuse an auditee of failing an audit.

6.2.2 Privacy-preserving Auditing for Blockchain-based Ledgers

For financial institutions, auditing is critical to proving that they are complying with regulations. In traditional auditing, the institutions engage the auditing companies such as Pricewaterhouse Coopers to audit the financial status of institutions. This type of auditing is time-consuming and burdensome. Blockchain-based ledgers enable the regulators and auditors to get real-time access to the information of the institutions. As a result, blockchain-based settlement and auditing can achieve high levels of transparency, real-time settlement, and low verification and reconciliation costs. However, the transaction data are usually sensitive for financial institutions. They may be reluctant to disclose all the contents of transactions to auditors, since the data can reveal the investment and trading strategies of institutions to others. Moreover, according to GDPR, a data protection regulation, it is not allowed to share users' data with others unless permission is obtained. Thus, the blockchain-based ledgers should not be public and cannot support the third-party auditing without disclosing all the details of transactions to auditors.

For the auditing functionalities, different audit-related calculations should be supported on ciphertexts, such as sum, product, and ratios. Moreover, the correctness and completeness of the auditing results should be guaranteed, which means financial institutions cannot leave out some transactions or hide assets from the auditor. Achieving both privacy and regulation is a challenging problem, especially taking into efficiency and scalability. To guarantee completeness, current privacy-preserving auditing schemes [22,132] use the table construction in the distributed ledger. Moreover, a transaction involves every participant in the system, which may cause high storage overhead and makes the system inefficient when there is a large number of participants in the system. To reduce the transaction delay and storage overhead, payment channel and ledger pruning are employed in some payment schemes [22, 132]. However, blockchain-based auditing is not incompatible with the ledger compression techniques since only the data on the chain can be audited by the auditor. Furthermore, providing regulatory control for cross-chain atomic swap is also a problem we will focus on in future work.

6.3 Final Remarks

In this thesis, we have designed and evaluated a suite of privacy-preserving and regulation-enabled schemes to resolve the privacy and accountability issues in blockchain-based financial services. The design ideas in ensuring transaction confidentiality and identity privacy as well as identity recovery mechanisms may shed light on future research on blockchain-based systems where privacy preservation is critical. Future research directions have been discussed to provide insight toward achieving more practical blockchain-based financial services.

References

[1] "IBM Blockchain for financial services means more trust for all," https://www.ibm.com/blockchain/industries/financial-services.

[2] D. Jayasuriya Daluwathumullagamage and A. Sims, "Fantastic beasts: Blockchain based banking," *Journal of Risk and Financial Management*, vol. 14, no. 4, p. 170, 2021.

[3] S. B. Patel, P. Bhattacharya, S. Tanwar, and N. Kumar, "Kirti: A blockchain-based credit recommender system for financial institutions," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1044–1054, 2020.

[4] G. Su, W. Yang, Z. Luo, Y. Zhang, Z. Bai, and Y. Zhu, "Bdtf: A blockchain-based data trading framework with trusted execution environment," in *Proc. 2020 16th International Conference on Mobility, Sensing and Networking (MSN)*, 2020, pp. 92–97.

[5] "Blockchain Market," https://www.marketsandmarkets.com/Market-Reports/blockchain-technology-market-90100890.html.

[6] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2019.

[7] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2015, pp. 281–310.

[8] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," *self-published paper, August*, vol. 19, p. 1, 2012.

[9] A. Clement, E. L. Wong, L. Alvisi, M. Dahlin, and M. Marchetti, "Making byzantine fault tolerant systems tolerate byzantine faults." in *NSDI*, vol. 9, 2009, pp. 153–168.

[10] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *OSDI*, vol. 99, no. 1999, 1999, pp. 173–186.

[11] C. Cachin *et al.*, "Architecture of the hyperledger blockchain fabric," in *Workshop on distributed cryptocurrencies and consensus ledgers*, vol. 310, no. 4, 2016.

[12] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *2014 {USENIX} Annual Technical Conference*, 2014, pp. 305–319.

[13] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi, "On security analysis of proof-of-elapsed-time (poet)," in *International Symposium on Stabilization, Safety, and Security of Distributed Systems*, 2017, pp. 282–297.

[14] U. Jafar, M. J. A. Aziz, and Z. Shukur, "Blockchain for electronic voting system—review and open research challenges," *Sensors*, vol. 21, no. 17, p. 5874, 2021.

[15] M. U. Hassan, M. H. Rehmani, and J. Chen, "Optimizing blockchain based smart grid auctions: A green revolution," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 1, pp. 462–471, 2021.

[16] D. Hu, Y. Li, L. Pan, M. Li, and S. Zheng, "A blockchain-based trading system for big data," *Computer Networks*, vol. 191, p. 107994, 2021.

[17] G. Caldarelli and J. Ellul, "The blockchain oracle problem in decentralized finance—a multivocal approach," *Applied Sciences*, vol. 11, no. 16, p. 7572, 2021.

[18] S. Saxena, B. Bhushan, and M. A. Ahad, "Blockchain based solutions to secure iot: background, integration trends and a way forward," *Journal of Network and Computer Applications*, vol. 181, p. 103050, 2021.

[19] J. Moosavi, L. M. Naeni, A. M. Fathollahi-Fard, and U. Fiore, "Blockchain in supply chain management: a review, bibliometric, and network analysis," *Environmental Science and Pollution Research*, pp. 1–15, 2021.

[20] P. Praitheeshan, L. Pan, and R. Doss, "Private and trustworthy distributed lending model using hyperledger besu," *SN Computer Science*, vol. 2, no. 2, pp. 1–19, 2021.

[21] Y.-N. Li, X. Feng, J. Xie, H. Feng, Z. Guan, and Q. Wu, "A decentralized and secure blockchain platform for open fair data trading," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 7, p. e5578, 2020.

[22] P. Chatzigiannis and F. Baldimtsi, "Miniledger: compact-sized anonymous and auditable distributed payments," in *Proc. European Symposium on Research in Computer Security*, 2021, pp. 407–429.

[23] A. I. Ali and D. T. Smith, "Blockchain and mortgage lending process: A study of people, process, and technology involved," *Online Journal of Applied Knowledge Management (OJAKM)*, vol. 7, no. 1, pp. 53–66, 2019.

[24] Y. J. Galteland and S. Wu, "Blockchain-based privacy-preserving fair data trading protocol," *Cryptology ePrint Archive*, 2021.

[25] E. Fletcher, C. J. Larkin, and S. Corbet, "Cryptocurrency regulation: Countering money laundering and terrorist financing," *Available at SSRN 3704279*, 2020.

[26] H. Nabilou, "How to regulate bitcoin? decentralized regulation for a decentralized cryptocurrency," *International Journal of Law and Information Technology*, vol. 27, no. 3, pp. 266–291, 2019.

[27] T. S. Sobh, "An intelligent and secure framework for anti-money laundering," *Journal of Applied Security Research*, vol. 15, no. 4, pp. 517–546, 2020.

[28] S. Delgado-Segura, C. Pérez-Solà, G. Navarro-Arribas, and J. Herrera-Joancomartí, "A fair protocol for data trading based on bitcoin transactions," *Future Generation Computer Systems*, vol. 107, pp. 832–840, 2020.

[29] M. Conti, E. S. Kumar, C. Lal, and S. Ruj, "A survey on security and privacy issues of bitcoin," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3416–3452, 2018.

[30] X. Shen, C. Huang, D. Liu, L. Xue, W. Zhuang, R. Sun, and B. Ying, "Data management for future wireless networks: Architecture, privacy preservation, and regulation," *IEEE Network*, vol. 35, no. 1, pp. 8–15, 2021.

[31] J. Camenisch and A. Lysyanskaya, "Signature schemes and anonymous credentials from bilinear maps," in *Proc. International Cryptology Conference*, 2004, pp. 56–72.

[32] M. Bellare, "A note on negligible functions." *Journal of Cryptology*, vol. 15, no. 4, 2002.

[33] Y. Tsiounis and M. Yung, "On the security of elgamal based encryption," in *Proc. International Workshop on Public Key Cryptography*, 1998, pp. 117–134.

[34] R. Gay, D. Hofheinz, L. Kohl, and J. Pan, "More efficient (almost) tightly secure structure-preserving signatures," in *Proc. International Conference on the Theory and Applications of Cryptographic Techniques*, 2018, pp. 230–258.

[35] J. Groth, "Efficient fully structure-preserving signatures for large messages," in *Proc. International Conference on the Theory and Application of Cryptology and Information Security*, 2015, pp. 239–259.

[36] J. Camenisch, M. Drijvers, and M. Dubovitskaya, "Practical UC-secure delegatable credentials with attributes and their application to blockchain," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 683–699.

[37] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)," *International Journal of Information Security*, vol. 1, no. 1, pp. 36–63, 2001.

[38] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[39] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," in *Proc. 28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, 1987, pp. 427–438.

[40] X. Sun, F. R. Yu, P. Zhang, Z. Sun, W. Xie, and X. Peng, "A survey on zero-knowledge proof in blockchain," *IEEE Network*, vol. 35, no. 4, pp. 198–205, 2021.

[41] R. Cramer, "Modular design of secure yet practical cryptographic protocols," *Ph. D. Thesis, CWI and University of Amsterdam*, 1996.

[42] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Proc. Conference on the Theory and Application of Cryptographic Techniques*, 1986, pp. 186–194.

[43] C. Rackoff and D. R. Simon, "Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack," in *Proc. International Cryptology Conference*, 1991, pp. 433–444.

[44] S. Agrawal, C. Ganesh, and P. Mohassel, "Non-interactive zero-knowledge proofs for composite statements," in *Annual International Cryptology Conference*, 2018, pp. 643–673.

[45] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *2013 IEEE Symposium on Security and Privacy*, 2013, pp. 238–252.

[46] J. Groth, "On the size of pairing-based non-interactive arguments," in *Annual international conference on the theory and applications of cryptographic techniques*, 2016, pp. 305–326.

[47] A. Vacca, A. Di Sorbo, C. A. Visaggio, and G. Canfora, "A systematic literature review of blockchain and smart contract development: Techniques, tools, and open challenges," *Journal of Systems and Software*, vol. 174, p. 110891, 2021.

[48] H. Wang, H. Qin, M. Zhao, X. Wei, H. Shen, and W. Susilo, "Blockchain-based fair payment smart contract for public cloud storage auditing," *Information Sciences*, vol. 519, pp. 348–362, 2020.

[49] E. W. Ngai, Y. Hu, Y. H. Wong, Y. Chen, and X. Sun, "The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature," *Decision Support Systems*, vol. 50, no. 3, pp. 559–569, 2011.

[50] B. van Gestel, "Mortgage fraud and facilitating circumstances," in *Situational Prevention of Organised Crimes*, 2013, pp. 129–147.

[51] S. Reno, S. S. R. A. Chowdhury, and I. Sadi, "Mitigating loan associated financial risk using blockchain based lending system," *Applied Computer Science*, vol. 17, no. 2, 2021.

[52] F. Schär, "Decentralized finance: On blockchain-and smart contract-based financial markets," *FRB of St. Louis Review*, 2021.

[53] "Aave protocol whitepaper," Website, https://whitepaper.io/document/533/aave-whitepaper.

[54] R. Leshner and G. Hayes, "Compound: The money market protocol," *White Paper*, 2019.

[55] A. Juliano, "dydx: A standard for decentralized margin trading and derivatives," *URl: https://whitepaper. dydx. exchange*, 2018.

[56] T. Hardjono and A. Pentland, "Verifiable anonymous identities and access control in permissioned blockchains," *arXiv preprint arXiv:1903.04584*, 2019.

[57] D. Bogatov, A. De Caro, K. Elkhiyaoui, and B. Tackmann, "Anonymous transactions with revocation and auditing in Hyperledger Fabric." *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 1097, 2019.

[58] D. Chaum and E. Van Heyst, "Group signatures," in *Workshop on the Theory and Application of Cryptographic Techniques*, 1991, pp. 257–265.

[59] H. Zheng, Q. Wu, B. Qin, L. Zhong, S. He, and J. Liu, "Linkable group signature for auditing anonymous communication," in *Australasian Conference on Information Security and Privacy*, 2018, pp. 304–321.

[60] M. Bellare, H. Shi, and C. Zhang, "Foundations of group signatures: The case of dynamic groups," in *Cryptographers' Track at the RSA Conference*, 2005, pp. 136–153.

[61] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Annual International Cryptology Conference*, 2004, pp. 41–55.

[62] R. L. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret," in *International Conference on the Theory and Application of Cryptology and Information Security*, 2001, pp. 552–565.

[63] W. A. A. Torres, R. Steinfeld, A. Sakzad, J. K. Liu, V. Kuchta, N. Bhattacharjee, M. H. Au, and J. Cheng, "Post-quantum one-time linkable ring signature and application to ring confidential transactions in blockchain (lattice ringct v1. 0)," in *Australasian Conference on Information Security and Privacy*, 2018, pp. 558–576.

[64] M. H. Au, J. K. Liu, T. H. Yuen, and D. S. Wong, "Id-based ring signature scheme secure in the standard model," in *International Workshop on Security*, 2006, pp. 1–16.

[65] D. Chaum, "Security without identification: Transaction systems to make big brother obsolete," *Communications of the ACM*, vol. 28, no. 10, pp. 1030–1044, 1985.

[66] M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham, "Randomizable proofs and delegatable anonymous credentials," in *Annual International Cryptology Conference*, 2009, pp. 108–125.

[67] J. Ni, X. Lin, and X. Shen, "Toward privacy-preserving valet parking in autonomous driving era," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2893–2905, 2019.

[68] J. Camenisch and A. Lysyanskaya, "An efficient system for non-transferable anonymous credentials with optional anonymity revocation," in *International conference on the theory and applications of cryptographic techniques*, 2001, pp. 93–118.

[69] R. Yang, M. H. Au, Q. Xu, and Z. Yu, "Decentralized blacklistable anonymous credentials with reputation," *Computers & Security*, vol. 85, pp. 353–371, 2019.

[70] R. Cleve, "Limits on the security of coin flips when half the processors are faulty," in *Proc. The Eighteenth Annual ACM symposium on Theory of computing*, 1986, pp. 364–369.

[71] G. Maxwell, "The first successful zero-knowledge contingent payment," Website, 2016, https://bitcoincore.org/en/2016/02/26/zero-knowledge-contingent-payments-announcement/.

[72] M. Petkus, "Why and how zk-SNARK works," *arXiv preprint arXiv:1906.07221*, 2019.

[73] Y. Li, C. Ye, Y. Hu, I. Morpheus, Y. Guo, C. Zhang, Y. Zhang, Z. Sun, Y. Lu, and H. Wang, "ZKCPlus: Optimized fair-exchange protocol supporting practical and flexible data exchange," in *Proc. 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 3002–3021.

[74] Y. Zhao, Y. Yu, Y. Li, G. Han, and X. Du, "Machine learning based privacy-preserving fair data trading in big data market," *Information Sciences*, vol. 478, pp. 449–460, 2019.

[75] Y. Li, L. Li, Y. Zhao, N. Guizani, Y. Yu, and X. Du, "Toward decentralized fair data trading based on blockchain," *IEEE Network*, vol. 35, no. 1, pp. 304–310, 2020.

[76] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)," *International journal of information security*, vol. 1, no. 1, pp. 36–63, 2001.

[77] D. Liu, C. Huang, J. Ni, X. Lin, and X. Shen, "Blockchain-cloud transparent data marketing: Consortium management and fairness," *IEEE Transactions on Computers*, 2022.

[78] W. Dai, C. Dai, K.-K. R. Choo, C. Cui, D. Zou, and H. Jin, "SDTE: A secure blockchain-based data trading ecosystem," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 725–737, 2019.

[79] D. Liu, J. Ni, C. Huang, X. Lin, and X. Shen, “Secure and efficient distributed network provenance for iot: A blockchain-based approach,” *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7564–7574, 2020.

[80] N. Van Saberhagen, “Cryptonote v 2.0,” 2013.

[81] G. Maxwell, “Coinjoin: Bitcoin privacy for the real world,” in *Post on Bitcoin forum*, 2013.

[82] S.-F. Sun, M. H. Au, J. K. Liu, and T. H. Yuen, “Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero,” in *European Symposium on Research in Computer Security*, 2017, pp. 456–474.

[83] I. Miers, C. Garman, M. Green, and A. D. Rubin, “Zerocoin: Anonymous distributed e-cash from bitcoin,” in *2013 IEEE Symposium on Security and Privacy*, 2013, pp. 397–411.

[84] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, “Zerocash: Decentralized anonymous payments from bitcoin,” in *2014 IEEE Symposium on Security and Privacy*, 2014, pp. 459–474.

[85] I. Damgård and E. Fujisaki, “A statistically-hiding integer commitment scheme based on groups with hidden order,” in *International Conference on the Theory and Application of Cryptology and Information Security*, 2002, pp. 125–142.

[86] C. Garman, M. Green, and I. Miers, “Accountable privacy for decentralized anonymous payments,” in *International Conference on Financial Cryptography and Data Security*, 2016, pp. 81–98.

[87] Y. Wu, H. Fan, X. Wang, and G. Zou, “A regulated digital currency,” *Science China Information Sciences*, vol. 62, no. 3, p. 32109, 2019.

[88] T. Ma, H. Xu, and P. Li, “Skyeye: A traceable scheme for blockchain.” *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 34, 2020.

[89] K. Wüst, K. Kostiainen, V. Čapkun, and S. Čapkun, “Prcash: Fast, private and regulated transactions for digital currencies,” in *International Conference on Financial Cryptography and Data Security*, 2019, pp. 158–178.

[90] C. Lin, D. He, X. Huang, M. K. Khan, and K.-K. R. Choo, “Dcap: a secure and efficient decentralized conditional anonymous payment system based on blockchain,”

IEEE Transactions on Information Forensics and Security, vol. 15, pp. 2440–2452, 2020.

[91] P. Chatzigiannis, F. Baldimtsi, and K. Chalkias, “Sok: Auditability and accountability in distributed payment systems,” in *International Conference on Applied Cryptography and Network Security*, 2021, pp. 311–337.

[92] K. Chalkias, K. Lewi, P. Mohassel, and V. Nikolaenko, “Distributed auditing proofs of liabilities,” *Cryptology ePrint Archive*, 2020.

[93] Y. Li, G. Yang, W. Susilo, Y. Yu, M. H. Au, and D. Liu, “Traceable monero: Anonymous cryptocurrency with enhanced accountability,” *IEEE Transactions on Dependable and Secure Computing*, 2019.

[94] E. Androulaki, J. Camenisch, A. D. Caro, M. Dubovitskaya, K. Elkhiyaoui, and B. Tackmann, “Privacy-preserving auditable token payments in a permissioned blockchain system,” in *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, 2020, pp. 255–267.

[95] M. B. Aalbers, “The financialization of home and the mortgage market crisis,” in *The Financialization of Housing*, 2016, pp. 40–63.

[96] Q. Yang and H. Wang, “Toward trustworthy vehicular social networks,” *IEEE Communications Magazine*, vol. 53, no. 8, pp. 42–47, 2015.

[97] J. Mendling, I. Weber, W. V. D. Aalst, J. V. Brocke, C. Cabanillas, F. Daniel, S. Debois, C. D. Ciccio, M. Dumas, S. Dustdar *et al.*, “Blockchains for business process management-challenges and opportunities,” *ACM Transactions on Management Information Systems (TMIS)*, vol. 9, no. 1, pp. 1–16, 2018.

[98] J. Ni, K. Zhang, Y. Yu, X. Lin, and X. S. Shen, “Providing task allocation and secure deduplication for mobile crowdsensing via fog computing,” *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 3, pp. 581–594, 2018.

[99] A. Tapscott and D. Tapscott, “How blockchain is changing finance,” *Harvard Business Review*, vol. 1, no. 9, pp. 2–5, 2017.

[100] K. Korpela, J. Hallikas, and T. Dahlberg, “Digital supply chain transformation toward blockchain integration,” in *Proc. the 50th Hawaii International Conference on System Sciences*, 2017.

[101] D. Liu, A. Alahmadi, J. Ni, X. Lin, and X. Shen, “Anonymous reputation system for IIoT-enabled retail marketing atop PoS blockchain,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3527–3537, 2019.

[102] J. Kang, R. Yu, X. Huang, S. Maharjan, Y. Zhang, and E. Hossain, “Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium blockchains,” *IEEE Transactions on Industrial Informatics*, vol. 13, no. 6, pp. 3154–3164, 2017.

[103] D. Zabelin, T. Yuyama, and T. Nakanishi, “The world is drowning in data. why don’t we trade it like on a stock exchange,” Website, 2022, https://www.weforum.org/agenda/2022/01/data-trading-stock-exchange/.

[104] “Dawex,” Website, https://www.dawex.com/en/.

[105] F. Liang, W. Yu, D. An, Q. Yang, X. Fu, and W. Zhao, “A survey on big data market: Pricing, trading and protection,” *IEEE Access*, vol. 6, pp. 15 132–15 154, 2018.

[106] “Xignite market data cloud platform on AWS,” Website, https://aws.amazon.com/financial-services/partner-solutions/xignite-market-data-cloud-platform/.

[107] L. Xue, D. Liu, J. Ni, X. Lin, and X. Shen, “Enabling regulatory compliance and enforcement in decentralized anonymous payment,” *IEEE Transactions on Dependable and Secure Computing*, 2022.

[108] M. Kayastha, S. Karim, R. Sandu, and E. Gide, “Ethereum blockchain and interplanetary file system (IPFS) based application model to record and share patient health information: An exemplary case study for e-health education in Nepal,” in *Proc. 2021 19th International Conference on Information Technology Based Higher Education and Training (ITHET)*, 2021, pp. 1–7.

[109] S. Dziembowski, L. Eckey, and S. Faust, “Fairswap: How to fairly exchange digital goods,” in *Proc. 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 967–984.

[110] M. Campanelli, R. Gennaro, S. Goldfeder, and L. Nizzardo, “Zero-knowledge contingent payments revisited: Attacks and payments for services,” in *Proc. 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 229–243.

[111] J. Bobolz, F. Eidens, S. Krenn, S. Ramacher, and K. Samelin, "Issuer-hiding attribute-based credentials," in *Proc. International Conference on Cryptology and Network Security*, 2021, pp. 158–178.

[112] O. Sanders, "Efficient redactable signature and application to anonymous credentials," in *Proc. IACR International Conference on Public-Key Cryptography*, 2020, pp. 628–656.

[113] W. Zou, D. Lo, P. S. Kochhar, X.-B. D. Le, X. Xia, Y. Feng, Z. Chen, and B. Xu, "Smart contract development: Challenges and opportunities," *IEEE Transactions on Software Engineering*, vol. 47, no. 10, pp. 2084–2106, 2019.

[114] "Diabetes data set," Website, https://archive.ics.uci.edu/ml/datasets/diabetes.

[115] C. Liu, R. Ranjan, C. Yang, X. Zhang, L. Wang, and J. Chen, "MuR-DPA: Top-down levelled multi-replica merkle hash tree based secure public auditing for dynamic big data storage on cloud," *IEEE Transactions on Computers*, vol. 64, no. 9, pp. 2609–2622, 2014.

[116] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proc. International Cryptology Conference*, 1991, pp. 129–140.

[117] "Miracl Library," Website, https://github.com/miracl/MIRACL.

[118] G. C. Pereira, M. A. Simplício Jr, M. Naehrig, and P. S. Barreto, "A family of implementation-friendly BN elliptic curves," *Journal of Systems and Software*, vol. 84, no. 8, pp. 1319–1326, 2011.

[119] K. Iyer and C. Dannen, *Building Games with Ethereum Smart Contracts*, 2018.

[120] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Website, https://bitcoin.org/bitcoin.pdf.

[121] H. Wang, Q. Wang, and D. He, "Blockchain-based private provable data possession," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[122] D. E. Kouicem, Y. Imine, A. Bouabdallah, and H. Lakhlef, "A decentralized blockchain-based trust management protocol for the internet of things," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[123] Website, https://coinmarketcap.com/.

[124] L. Xue, D. Liu, J. Ni, X. Lin, and X. Shen, "Balancing privacy and accountability for industrial mortgage management," *IEEE TII*, vol. 16, no. 6, pp. 4260–4269, 2019.

[125] S. Mabunda, "Cryptocurrency: The new face of cyber money laundering," in *2018 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)*, 2018, pp. 1–6.

[126] Y. Lindell, "An efficient transform from sigma protocols to nizk with a crs and non-programmable random oracle," in *Theory of Cryptography Conference*, 2015, pp. 93–109.

[127] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin (extended version)," in *Cryptology eprint Archive*, 2014.

[128] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *2018 IEEE S&P*, 2018, pp. 315–334.

[129] Website, https://github.com/StarLI-Trapdoor/libsnark_sample.

[130] K. Chalkias, P. Chatzigiannis, and Y. Ji, "Broken proofs of solvency in blockchain custodial wallets and exchanges," *Cryptology ePrint Archive*, 2022.

[131] H. Wang, D. He, and Y. Ji, "Designated-verifier proof of assets for bitcoin exchange using elliptic curve cryptography," *Future Generation Computer Systems*, vol. 107, pp. 854–862, 2020.

[132] N. Narula, W. Vasquez, and M. Virza, "{zkLedger}:{Privacy-Preserving} auditing for distributed ledgers," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, 2018, pp. 65–80.

[133] L. Du, Y. Tao, T. Chen, Q. Wang, and H. Lv, "An advanced pbft-based consensus algorithm for a bidding consortium blockchain," in *Proc. 2021 The 3rd International Conference on Blockchain Technology*, 2021, pp. 176–182.

[134] M. Aazam, S. Zeadally, and K. A. Harras, "Deploying fog computing in industrial internet of things and industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4674–4682, 2018.

[135] Y. Zhang, C. Xu, H. Li, K. Yang, J. Zhou, and X. Lin, "HealthDep: An efficient and secure deduplication scheme for cloud-assisted ehealth systems," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 4101–4112, 2018.

[136] S. Wu, J. B. Rendall, M. J. Smith, S. Zhu, J. Xu, H. Wang, Q. Yang, and P. Qin, "Survey on prediction algorithms in smart homes," *IEEE Internet of Things Journal*, vol. 4, no. 3, pp. 636–644, 2017.

[137] H. Yang, Q. Zhou, M. Yao, R. Lu, H. Li, and X. Zhang, "A practical and compatible cryptographic solution to ADS-B security," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3322–3334, 2018.

[138] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal on Computing*, vol. 18, no. 1, pp. 186–208, 1989.

[139] J. Camenisch and M. Stadler, "Efficient group signature schemes for large groups," in *Proc. International Cryptology Conference*, 1997, pp. 410–424.

[140] D. Chaum and E. v. Heyst, "Group signatures," in *Proc. Workshop on the Theory and Application of Cryptographic Techniques*, 1991, pp. 257–265.

[141] M. Bellare, H. Shi, and C. Zhang, "Foundations of group signatures: The case of dynamic groups," in *Proc. Cryptographers' Track at the RSA Conference*, 2005, pp. 136–153.

[142] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Proc. International Cryptology Conference*, 2004, pp. 41–55.

[143] H. Zheng, Q. Wu, B. Qin, L. Zhong, S. He, and J. Liu, "Linkable group signature for auditing anonymous communication," in *Proc. Australasian Conference on Information Security and Privacy*, 2018, pp. 304–321.

[144] R. L. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret," in *Proc. International Conference on the Theory and Application of Cryptology and Information Security*, 2001, pp. 552–565.

[145] W. A. Alberto Torres, R. Steinfeld, A. Sakzad, J. K. Liu, V. Kuchta, N. Bhattacharjee, M. H. Au, and J. Cheng, "Post-quantum one-time linkable ring signature and application to ring confidential transactions in blockchain (lattice RingCT v1. 0)," in *Proc. Australasian Conference on Information Security and Privacy*, 2018, pp. 558–576.

[146] D. Chaum, "Security without identification: Transaction systems to make big brother obsolete," *Communications of the ACM*, vol. 28, no. 10, pp. 1030–1044, 1985.

[147] J. Ni, X. Lin, and X. Shen, "Toward privacy-preserving valet parking in autonomous driving era," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2893–2905, 2019.

[148] R. Yang, M. H. Au, Q. Xu, and Z. Yu, "Decentralized blacklistable anonymous credentials with reputation," *Computers & Security*, vol. 85, pp. 353–371, 2019.

[149] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Proc. Conference on the Theory and Application of Cryptographic Techniques*, 1986, pp. 186–194.

[150] D. Boneh, "The decision diffie-hellman problem," in *Proc. International Algorithmic Number Theory Symposium*, 1998, pp. 48–63.

[151] R. Zhang, R. Xue, and L. Liu, "Security and privacy on blockchain," *ACM Computing Surveys (CSUR)*, vol. 52, no. 3, pp. 1–34, 2019.

[152] D. Pointcheval and O. Sanders, "Short randomizable signatures," in *Cryptographers' Track at the RSA Conference*, 2016, pp. 111–126.

[153] S. Gazi, "Regulatory responses to cryptoderivatives in the uk and the eu: The future of cryptoderivatives in the us," *Available at SSRN 3737947*, 2019.

[154] "Q4 2019 cryptocurrency anti-money laundering report," Website, 2020, https://ciphertrace.com/q4-2019-cryptocurrency-anti-money-laundering-report/.

[155] J. Camenisch, "Group signature schemes and payment systems based on the discrete logarithm problem," Ph.D. dissertation, ETH Zurich, 1998.

[156] Website, https://github.com/zkcrypto/bellman.

[157] Website, https://github.com/scipr-lab/libsnark.

Appendices

Appendix A

Zero-knowledge Proof for the Counter in RDAP

The relation $R = \{(r_p, \theta, \mu, \phi, cID) : C_1 = Com_q(cID_x) \wedge C_2 = Com_q(cID_y) \wedge cID = cID_{sn}h^{\phi} \wedge cID_{sn} = g_1^{r_p}g_2^{V}g_3^{\theta}g_4^{\mu}g_5^{\xi} \wedge C_{r_p} = Com_p(r_p)\}$ indicates that the counter in the transaction belongs to $\mathcal{U}$. $Com_p(r_p) = \bar{g}^{r_p}h^{t_{r_p}}$. Note that since cID, r_p, θ, μ, ξ, and ϕ are not public, to prove that $cID = g_1^{r_p}g_2^{V}g_3^{\theta}g_4^{\mu}g_5^{\xi}h^{\phi}$, double discrete logarithm proof is involved [44]. The details of the proof for R is shown as follows.

- Given the hash function $H_1 : \{0,1\}^* \rightarrow \{0,1\}^k$ and a security parameter λ with $\lambda \leq k$, for $cID = g_1^{r_p}g_2^{V}g_3^{\theta}g_4^{\mu}g_5^{\xi}h^{\phi}$, $\mathcal{U}$ generates the random numbers $\alpha_1, ..., \alpha_\lambda$, $\beta_1, ..., \beta_\lambda, \gamma_1, ..., \gamma_\lambda, \delta_1, ..., \delta_\lambda, \zeta_1, ..., \zeta_\lambda, \epsilon_1, ..., \epsilon_\lambda, \in Z_p$.
- For $1 \leq i \leq \lambda$, $\mathcal{U}$ computes $a_{i,1} = Com_p(\alpha_i) = \bar{g}^{\alpha_i}h^{t_{\alpha_i}}$, $a_{i,2} = Com_p(\beta_i) = \bar{g}^{\beta_i}h^{t_{\beta_i}}$, $a_{i,3} = Com_p(\gamma_i) = \bar{g}^{\gamma_i}h^{t_{\gamma_i}}$, $a_{i,4} = Com_p(\delta_i) = \bar{g}^{\delta_i}h^{t_{\delta_i}}$, $a_{i,5} = Com_p(\zeta_i) = \bar{g}^{\zeta_i}h^{t_{\zeta_i}}$, and $a_{i,6} = Com_p(\epsilon_i) = {g_1}^{\epsilon_i}h^{t_{\epsilon_i}}$. Then, $\mathcal{U}$ computes $A_i = g_1^{\alpha_i}g_2^{\beta_i}g_3^{\gamma_i}g_4^{\delta_i}g_5^{\zeta_i}h^{\epsilon_i}$. Let the coordinate values of A_i be $(A_{i,x}, A_{i,y})$. $\mathcal{U}$ computes the commitment of $A_{i,x}$ as $a_{i,7} = Com_q(A_{i,x}) = \hat{g}^{A_{i,x}}\hat{h}^{t_{A_{i,x}}}$, and the commitment of $A_{i,y}$ as $a_{i,8} = Com_q(A_{i,y}) = \hat{g}^{A_{i,y}}\hat{h}^{t_{A_{i,y}}}$.
- $\mathcal{U}$ computes:

$$\begin{aligned} c =& H_1(g_1||g_2||g_3||g_4||g_5||h||\bar{g}||\hat{g}||\hat{h}||a_{1,1}||a_{1,2}||\ldots,||a_{1,8} \\ & ||\ldots,||a_{i,1}||a_{i,2}||\ldots,||a_{i,8}). \end{aligned}$$

- For $1 \leq i \leq \lambda$, in case that $c[i] = 0$, $\mathcal{U}$ computes $z_{i,1} = \alpha_i, z_{i,2} = \beta_i, z_{i,3} = \gamma_i, z_{i,4} = \delta_i, z_{i,5} = \zeta_i, z_{i,6} = \epsilon_i, z_{i,7} = t_{\alpha_i}, z_{i,8} = t_{\beta_i}, z_{i,9} = t_{\gamma_i}, z_{i,10} = t_{\delta_i}, z_{i,11} = t_{\zeta_i}, z_{i,12} =$

$t_{\epsilon_i}, z_{i,13} = t_{A_{i,x}}, z_{i,14} = t_{A_{i,y}}$. If $c[i] = 1$, $\mathcal{U}$ computes $z_{i,1} = \alpha_i - r_p$, $z_{i,2} = \beta_i - V$,$z_{i,3} = \gamma_i - \theta$,$z_{i,4} = \delta_i - \mu$, $z_{i,5} = \zeta_i - \xi$, $z_{i,6} = \epsilon_i - \phi$, and $z_{i,7} = t_{\alpha_i} - t_{r_p}$. Let $T_i = g_1^{z_{i,1}} g_2^{z_{i,2}} g_3^{z_{i,3}} g_4^{z_{i,4}} g_5^{z_{i,5}} h^{z_{i,6}}$. Let the coordinate values of T_i be $(T_{i,x}, T_{i,y})$. $\mathcal{U}$ proves that $PK\{(cID_x, cID_y, A_{i,x}, A_{i,y}) : T_i = A_i/cID\}$ [44], and the resulted proof is denoted by π_i. After that, $\mathcal{U}$ adds $\{z_{i,1}, \ldots, z_{i,14}\}$ to the transaction if $c[i] = 0$. Otherwise, it adds $\{z_{i,1}, \ldots, z_{i,6}, \pi_i\}$ to the transaction. c and $\{a_{i,1}||a_{i,2}|| \ldots, ||a_{i,8}\}_{1 \le i \le \lambda}$ are also included in the transaction.

- For the verification, if $c[i] = 0$, where $1 \le i \le \lambda$, validators compute

$$T'_i = g_1^{z_{i,1}} g_2^{z_{i,2}} g_3^{z_{i,3}} g_4^{z_{i,4}} g_5^{z_{i,5}} h^{z_{i,6}} = (T'_{i,x}, T'_{i,y}),$$

and check whether $a_{i,1} = \bar{g}^{z_{i,1}} h^{z_{i,7}}$, $a_{i,2} = \bar{g}^{z_{i,2}} h^{z_{i,8}}$, $a_{i,3} = \bar{g}^{z_{i,3}} h^{z_{i,9}}$, $a_{i,4} = \bar{g}^{z_{i,4}} h^{z_{i,10}}$, $a_{i,5} = \bar{g}^{z_{i,5}} h^{z_{i,11}}$, $a_{i,6} = \bar{g}^{z_{i,6}} h^{z_{i,12}}$, $a_{i,7} = \hat{g}^{T'_{i,x}} \hat{h}^{z_{i,13}}$, and $a_{i,8} = \hat{g}^{T'_{i,y}} \hat{h}^{z_{i,14}}$. If $c[i] = 1$, validators compute $T'_i = g_1^{z_{i,1}} g_2^{z_{i,2}} g_3^{z_{i,3}} g_4^{z_{i,4}} g_5^{z_{i,5}} h^{z_{i,6}} = (T'_{i,x}, T'_{i,y})$, check whether $a_{i,1} = \bar{g}^{z_{i,1}} h^{z_{i,7}} C_{r_p}$, and verify π_i. Finally, validators compute c' and check whether $c' = c$.

- For the $c[i]$, where $1 \le i \le \lambda$, validators also verifies $cID_{sn} = g_1^{r_p} g_2^{V} g_3^{\theta} g_4^{\mu} g_5^{\xi}$ by using the same random numbers $\alpha_1, ..., \alpha_\lambda$, $\beta_1, ..., \beta_\lambda, \gamma_1, ..., \gamma_\lambda, \delta_1, ..., \delta_\lambda, \zeta_1, ..., \zeta_\lambda$. If the verification passes, $cID = cID_{sn} h^{\phi}$ holds, and the relation R is confirmed.