DOCTOR OF PHILOSOPHY THESIS

Application & Security of Smart Contract in Financial Industries

ZHIYU XU
Supervisor: **Prof. Yang Xiang**
Co-Supervisor: **Dr. Sheng Wen**

A dissertation submitted in fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in the
DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

December, 2022

Contents

Acknowledgements

First of all, I would like to thank my supervisors, Prof. Yang Xiang, Dr Sheng Wen, and Dr Donghai Liu, for all the efforts they have made for me in the past three years. At the early stage of my research, their wise suggestions and diligent training got me through the confusion stage and into the decentralized system security as early as possible. When I was worried about the results of not repeating the literature, it was Sheng, Yang and Donghai who taught me how to find the problems and correct them in time. When I was thinking hard about my research topic and couldn't find a solution, they made my mind clear. When I was writing my first paper, their word-by-word revision and patient guidance taught me how to write academic papers. These precious practical experiences in research trained me and inspired me, from which I learned to work independently.

In addition, I had the honour to work on blockchain-based data sharing on financial services with Dr Shiping Chen, Tengyun Jiao, and Minfeng Qi during my PhD study. I would like to express my sincere gratitude to these collaborators. I also received help and guidance from many other teachers at the Swinburne University of Technology, and I would like to express my appreciation. Finally, I would like to express my sincere gratitude to my family, including my father, mother, and brother (Weijie Wang). Thanks to their support from the beginning to the end, I completed my PhD studies successfully.

ZHIYU XU
Hawthorn, VIC 3122, Australia
December, 2022

Candidate Declaration

The thesis contains no material that has been accepted for the award to the candidate of any other degree or diploma. To the best of my knowledge, the thesis contains no material previously published or written by another person except where due refer- ence is made in the text of the examinable outcome, and with permission received to republish the work in the thesis.

ZHIYU XU
Hawthorn, VIC 3122, Australia
May, 2022

Abstract

Application & Security of Smart Contract in Financial Industries

Blockchain is an emerging technology that has the potential to revolutionize the global industry and create a trusted relationship in a multi-party business network. While there are several practical use cases where Blockchain has been applied, one specific area is the Art industry, where it is a natural fit in the way art forensics and transactions are conducted, tracked, and recorded. Estimating the worldwide annual art trade at $200 billion, about 3% ($6 billion) is attributed to illicit cultural property, ranging from counterfeit and forgeries to smuggled goods and outright theft. With the help of Blockchain, there is a high possibility to neutralize the threats in lost revenue globally. As a result, I propose several models on high-value commodities to solve privacy-related issues based on provenance information.

In addition, there is a trend in Open APIs and data sharing to provide better and innovative financial services. Data regulations aim to improve consumers' privacy and control over their data. However, it still confronts the challenges of properly managing and protecting users' privacy in an open data sharing paradigm. I propose a provenance-provided data sharing model and an automated consent management model via Blockchain to meet open banking requirements. Employing the Proof of Authority consensus mechanism to balance the business flexibility and efficiency with client privacy control. In my future work, I plan to focus my research on detecting vulnerabilities in smart contracts. Firstly, I will conclude all smart contract vulnerabilities. And try to find common malicious patterns in smart contracts to detect vulnerabilities.

Chapter 1

Introduction

1.1 Ethereum and Smart Contract

Blockchain, with the properties of unforgeability, tampering resistance, transparency and verifiability, perfectly meet the requirements of an open banking system. Originate from Bitcoin [1] and after a long-time development, blockchain has evolved into a smart-contract supported system [2].The smart-contract platform enables the programmability of predefined laws and rules, which has a significant impact on the financial sector. The design of blockchain ensures that no single entity can modify, delete, or even append a record to the distributed ledger without the consensus of other network participants, thereby providing data that is unforgeable and resistant to tampering. Due to the publicly readable ledgers, the decentralised system ensures transparency and verifiability.

Most DApps use Ethereum as the underlying structure [3–6]. Typically, DApp providers use smart contracts to implement business logic and employ their own cryptocurrencies as the exchange medium. Smart contracts are executable codes that comply with predetermined conditions. Smart contracts are a set of blockchain-based computer codes designed to automatically facilitate, verify, or enforce predefined rules and principles. A contract's negotiation or agreement is dependent on the underlying consensus mechanism in order to maintain the consistency of orders and states. Smart contracts based on the blockchain are intrinsically traceable and irreversible without compromising their viability. In addition, it can perform complex functions such as signature, agreement management, and application data storage. Ethereum, which was introduced in July 2015, is the largest and most established underlying development platform that enables smart contracts to be constructed and executed on the blockchain. Additionally, Ethereum is regarded as the most popular DApp development platform (Decentralized Application). Ethereum has on the following characteristics:

- Ethereum is a "Turing-complete" system capable of solving all theoretical computational problems. Real-world application scenarios with the support of smart contracts can implement more intricate logic.

- Ethereum performs well while building consortium blockchain. Last year, the world's most extensive consortium system Hyperledger[7], announced that EEA (Enterprise Ethereum Alliance) and Hyperledger would collaborate to meet global demand for enterprise blockchain.
- Ethereum system is flexible and scalable and Ethereum platform is still evolving. Ethereum developer community proposed Ethereum Improvement Proposal (EIP)[8] as the standards of new features, protocols, contracts or APIs.

There are two categories of privacy issues in decentralized applications: application privacy and blockchain privacy. Machine learning analysis can figure out the money flow on decentralized platforms and infer users' real identities. There are some existed methodologies that solve the application privacy, such as access control [9], and homomorphic encryption [10]. The 51% attack [11] and Sybil attack [12] are two representative attacks for blockchain privacy. Some methodologies solve blockchain privacy, such as ring signature [10] and zero-knowledge proof (ZKP) [10]. Although homomorphic encryption provided sound privacy-preserving on data sharing, homomorphic encryption is time-consuming. In contrast, response time and privacy-preserving in financial service platforms are significant features to evaluate a system.

1.2 Problem Statement

The financial service provides consumers with emerging financial products and services that enable them to exercise control over their data and grant permission to authorised data recipients. The sensitive and valuable nature of bank data presents both opportunities and challenges for consent management, authentication services, and client registration. There is an increasing number of companies that are willing to integrate blockchain into their system frameworks, such as the health-care [13][14], internet of things (IoT) [15][16] and high-value commodities [17][18] industries. The European Union proposed general data protection regulation [19] in 2016, which defines certain rules on customer data sharing.

The European Union proposed a general data protection regulation [19] in 2016, which defines specific rules on customer data sharing. Similarly, according to the legislation, financial services provide data read access to the ADRs in Australia [20]. Although the open banking service provides 'write' functions [21], it still have challenges and issues. First, financial data about clients will be shared among banking institutions. During data transmission, malicious data modification, spoofing, and data interception are all possible issues. These malicious actions undermine confidence and safety. Second, consumer experience research typically indicates that users will grant authorized institutions access to their bank information in exchange for an improved experience. Additional validation processes and regulation are potentially unnecessary expenses for banks. It will have an effect on the effectiveness of consent management. Thirdly, managing the duration of consent can be difficult. Clients are required to reauthorize consents that expire over time. Otherwise, ADRs may allow unauthorised access to client data after the end of service. Lastly, if historical access logs are not transparent, clients are unable to regulate and audit their data, which may result in financial loss. The inherent characteristics of Blockchain technology may provide various features that can address record transparency and user identity pseudo-anonymity. Users can audit and review records quickly because the blockchain ledger is append-only. Smart contracts [22] are automated scripts that can preprogram business requirements, which can improve the efficiency of the transaction processing.

1.3 Report Organization

This section illustrates the structure of this report. The content of the rest sections is as followed:

- **Section 2** presents the related work and background of data sharing applications in blockchain area and vulnerabilities in smart contracts.
- **Section 3** illustrates the ArtChain, a blockchain-based art trading system, which has been piloted and in operation as a working product in practice. It is expected to provide a total solution to these issues by creating a new ecosystem for art. ArtChain consists of the underlying architecture of blockchain to support a business system centered around art assets trading platforms.
- **Section 4** presents a provenance-provided data sharing model for open banking via blockchain. The model employs the programmable smart contracts as the middle witness between users and third-party services, and provides the modifications on data layer (data content, transaction structure), smart contract layer (ACL, logic), and application layer (customized APIs). Based on that, our PPM model possesses the properties of transparent authentication, privacy-provided control, and auditable provenance.
- **Section 5** presents a blockchain-based hierarchical data access model for financial services. We implement granular data access control by scoring authorised data recipients (ADRs). By accessing the corresponding blockchain service logs, the credits of ADRs will be dynamically updated. The algorithm for credit evaluation is responsible for calculating ADRs' credit based on their completion rate, business ethics rate, and feedback positive rate. In addition, by employing smart contracts, the efficiency of consent management can be enhanced and privacy policies can be managed flexibly.
- **Section 6** presents an automated consent management model for blockchain financial services platform. We employ the Proof of Authority consensus mechanism to balance the business flexibility and efficiency with client privacy control. Our model enables automated, transparent, and accountable management for consumer consents in financial services. We implement a proof of concept prototype to demonstrate the feasibility and applicability of our model.

- **Section 7** presents an image-based privacy-preserving blockchain model for financial services. We develop an image-based privacy-preserving scheme to store users' financial service data as images. Transferring the numerical service data into images ensures that users can comprehend the content and prevent data recognized by the machine.
- **Section 8** illustrates the main contributions of the thesis and the future work.
- **Section 9** lists all published papers during the PhD study.

Chapter 2

Literature Review

2.1 Blockchain-Based Data-Sharing Application

In the financial system, authentication for actions and privacy of users attracts lots of concerns since the leakage of sensitive data leads to dramatic financial loss and property damage. To address the critical problems, several types of research about data sharing focus on transparent authentication [23–25], privacy provided [26–28], and provenance [29–32] has been proposed in recent years. These frameworks utilize the immutability, autonomy, and anonymity properties of blockchain to solve the issues in the traditional areas. The financial services usually provide various financial products for clients. In order to have a better user experience, clients need to share sensitive data with ADRs. Consequently, relevant and well-considered security controls are required. In an integrated and reliable open banking system, the access controls and resource authentication are necessary to guarantee and enhance security protection. So far, there are many existing researches that are designed for certificate management and authentication issues. To make the financial data more useful, integrating financial data, service providers, and consumers into a platform for valuable data interconnections. In a data-sharing system, access control [33] is usually the most concerning security feature. If data access controls are not carefully designed, users' sensitive data can easily leak and cause financial damage.

With the development of blockchain, many distributed data sharing systems have emerged. Sabyasachi's work [34] proposed a theoretical blockchain-based credit analysis framework for financial systems. It explains the importance of the scoring system in the financial system and how to evaluate the credit based on the uploaded transaction data. However, the system is still a theoretical model and has not been proved experimentally. Moreover, credits are generated based on user-uploaded transaction history data, which may contain fraud data. Zhang [35] established a data-sharing system based on Hyperledger Fabric, which provides data access control. They design a behavior chain for recording operators' actions in an AI-powered network and utilizing these records to realize the data access control. However, the system does not specify how the user's behavior is evaluated to implement data access control. Hong [36] proposed a hierarchical attribute-based encryption algorithm, which uses a hierarchical attribute structure and multi-level authorization center. This system realizes the fine-grained access control by sending different user data to various authorization centers. Although this model has a complete theoretical analysis and experiment, hierarchical access to data based on encryption does not suit our model.

When it comes to the user privacy of the data-sharing application, we are primarily concerned with two aspects: application privacy and blockchain privacy.

- **Application Privacy:** The privacy of an application encompasses both the personal information of its users and data pertaining to the platform itself. When discussing the privacy of applications, the confidential information of the user is brought up first. When it comes to managing data, a centralized platform presents many challenges. For instance, setting access rights to user data incorrectly, using attack scripts to override server access, and attacking servers themselves. These kinds of security risks can potentially result in the theft or misuse of user information.

- **Blockchain Privacy:** Blockchain pioneers a new computing paradigm and collaboration model for building trust at low cost in an untrustworthy competitive environment, and its consensus algorithm mechanism provides at least two important supports for privacy protection: trusted third parties and resistance to cheating, both of which make blockchain essential for privacy protection. Traditional privacy protection without the use of blockchain can be achieved by constructing a private network (which is pricey) or by transferring data via HTTPS; however, this requires HTTPS certificate services. This requires a self-built certificate server in the case of an enterprise with only an intranet, and a cross-domain certificate server when reliable transfers are made between multiple enterprises with only an intranet. The security risks associated with relying on a centralised authority to manage the issuance and revocation of certificates are increased. Blockchain, as a trusted third party, can establish a "trusted third party" among multiple peers at a very low cost through consensus, with nodes participating in consensus jointly performing certificate issuance, authorisation, signature verification, and zero-knowledge proof verification via blockchain.

2.1.1 Certificate Management:

Traditional certificate authorities (CA) may have fraudulent certificates signature[37][38] and certificates revocation[39][40] issues. These problems put the system at risk. CertLedger[41] is the framework with a transparent certificate public key infrastructure(PKI) that based on blockchain. Through utilizing the immutability and autonomy of blockchain, it stores certificate records on the public chain to provide a transparent CA management. Guilherme[42] proposed a blockchain-based PKI system for internet of things (IoT) information. They connected the key-based platform and blockchain-based

PKI to realize a protocol that makes users accounts pseudo-anonymous in IoT devices. However, CertLedger does not support the automation on their CA verification. Moreover, Guilherme's work does not support the IoT devices' certificates revocation, and the adopted the proof of work (PoW) consensus algorithm can be attacked by others with enormous computational power.

2.1.2 Authentication Process:

Authentication services are an indispensable element in distributed platforms. They contain access controls and resource allocation management. Dongxing[43] proposed a blockchain-based authentication mechanism for IoT. This system employed hyperledger fabric to record device IDs on the blockchain in order to verify device identity. Besides, they built the permission chain to implement access control. Nisha's[44] work focused on the blockchain-based identity authentication and revocation models for vehicular networks. In the system, vehicles obtained a pseudo-ID from CA and stored the corresponding certificates on the blockchain. Moreover, any operations on their identities would be updated on a shared ledger. It aims to mitigate the reliance on ADRs for authentication. Unfortunately, Nisha's work is not entirely decentralized and still requires trusted authorities to participate in the authentication process.

Authentication service is a necessary component of a distributed system. Authentication to access websites, like the front door, provides a quick and secure connection between users and service providers. A powerful third party manages authentication, typically in the form of Certificate Authority (CA), resulting in single-point failure and inevitable mistrust. Consequently, a blockchain-based PKI system emerges to address the aforementioned issues. Bo [45] proposed a distributed certificate scheme to distribute the centralized CA. The public data is recorded in the nodes, and decentralized Certificate libraries are realize through the modified Merkle trees. Matsumoto proposed the Instant Karma PKI (IKP) model [39] to record the traditional CA's behaviour on the blockchain. The recording method can monitor CA and detect malicious actions made by the powerful authority. Gan's approach [46] builds a private blockchain to store and manage IoT devices' latest public keys. Users can update or modify public keys through sending transactions. Other proposed models, such as [47][48][49][50], also solve authentication issues to some extent.

2.1.3 Auditable Provenance:

Data provenance provides users or institutions with the ability to audit and trace the resource. Provenance record the history of the data request, which is necessary for users and institutions. ProvChain [30] is a provenance recording system based on blockchain. It treated blockchain as an online distributed database to store data. Those data is validated by provenance auditor offline. Due to the immutability of blockchain, data cannot be erased, which guarantees the integrity of data. DataProv [51] combine provenance data and smart contract together to provide secure and reliable service for sensitive cloud data. When users send a data request to nodes, the smart contract will automatically send changes to blockchain network. Finally, modifications to data will be approved or rejected by nodes. Many applications employ the provenance-provided blockchain to address the real-life problem [52][53].

2.1.4 Data Regulation:

Several issues still exist in traditional data privacy services, such as hiding users' identity [54], sealing transaction amounts [55], data control [56] [14], and access control list maintenance [57]. Our PPM focuses on the customizing consumers' Access Control List (ACL) to realize fine-grained management. The customizing the ACL enables users to control and share personal data with other parties with their own willingness. Zyskind [56] proposed a new system which combines and off-chain storage though defining T_{access} and T_{data} in the system. T_{access} focuses on users' permission allocation and data management. T_{data} is designed for data storage and sharing. This model achieves access control by sending different transactions. Xia proposed BBDS model [58] to enhance the privacy aspect for medical records in the remote servers. It modifies the transaction and block header to meet requirements for medical records. Moreover, BBDS implements access control by using identity-based authentication.

Financial, political, and other industries have utilised data regulation for quite some time. In recent years, data regulation has become a hot topic due to the proliferation of Fintech. The benefits of data sharing include more accurate forecasts and analyses, as well as increased productivity. In addition to providing convenience, it introduces a number of unavoidable problems. Treleaven's work [59] proposed using smart contracts to regulate behaviours of financial institutions. Through automated reporting and monitoring of social media to achieve the automated regulation. Although the proposed idea is innovative, the model is in a theoretical level, and the adaptability of applications is unknown. Asaph's work [60] focuses on data regulation about electronic medical data. The proposed model illustrates the management of medical data among various medical stakeholders. The stakeholders and patients are treated as provider and patient nodes, respectively. Managing medical data using blockchain technology is an innovative concept. However, the paper does not contain any practical evaluations of the model. Additionally, the blockchain platform is undefined.

Research on the privacy protection of blockchain is crucial to the practical application of blockchain technology. This section examines and categorises the privacy threats that blockchain's network and transaction layers face. In addition, we summarise the existing blockchain privacy protection schemes and compare them to the image-based encryption scheme we propose.

The primary privacy threat at the network layer emanates from the attacker's monitoring of blockchain communication data and the analysis of the topological relationship between nodes. Taking Bitcoin with the most significant number of participants as an example, an attacker can easily access the Bitcoin network and obtain the IP addresses and transaction data by deploying probe nodes. Based on the above information, the authors [61, 62] roughly infer the geographical distribution and scale of Bitcoin nodes, the propagation path of network layer information, and even associate user transaction information with their IP address.

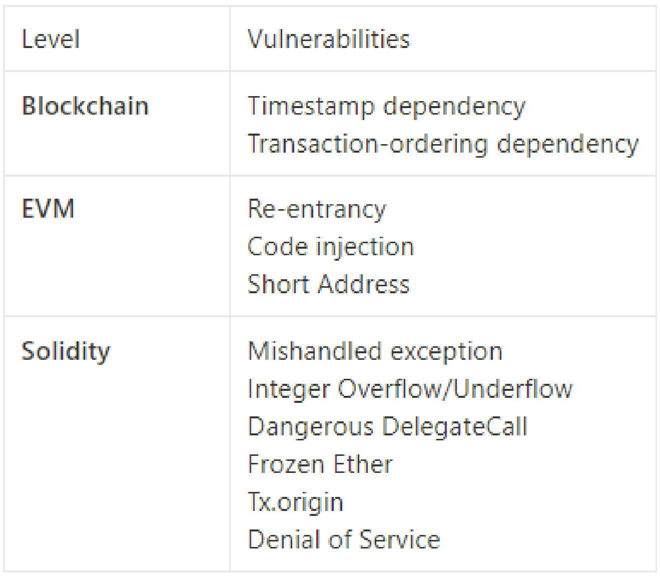
For the transaction layer, the risk of user privacy leakage lies in two aspects. One aspect is that the attacker can analyze a specific account's transaction details, fund flow, and balance through public transaction records. Another aspect is the identification of user identity privacy. Through techniques such as cluster analysis [63, 64], authors obtain a large number of associated addresses and different user sets. In addition, through the abstraction of transaction behaviors [65], and deep neural network [66], authors can realize the association of users' bitcoin addresses and real identities.

Similarly, to address the above privacy threats, we summarize the current privacy protection mechanisms for blockchain from two perspectives, namely network and transaction layers. At the network layer, the focus of countering attackers is to enhance the detection of malicious nodes and hide the IP address of normal nodes. Huang et al. [67] propose a method to quickly detect and locate malicious nodes based on behavioral pattern clustering. Ouaddah et al. [9] design an access control network model, which can perform strict authorization control on blockchain nodes, thereby increasing The difficulty for the attacker to access the network and deploy probe nodes. Additionally, running on TOR network [68] (recommended by Bitcoin [69]) or Invisible Internet Project (I2P) [70] (adopted by Monero [71]) are able to provide better privacy protection for blockchain. Technically speaking, this approach can hide node IP by running blockchain on an anonymous network, preventing attackers from finding and tracking IP.

The key to reducing privacy leakage at the transaction layer is to display as little and specific transaction information as possible to attackers. Mix service [72] is a popular solution to mitigate blockchian privacy attacks by obfuscating transaction relationships between transaction senders and receivers. Specifically, it developed into centralized (Dash [73]) and decentralized (Coinjoin [74]) directions depending on whether it needs a third party. Besides that, there are some encryption-based schemes that have proven suitable for blockchain, such as Ring signature (CryptoNote [75] and Monero [71]), Non-interactive zero-knowledge proof (Zerocash [76]), and Homomorphic encryption (Monero [71]). Another approach to protecting privacy is to allocate privacy-sensitive data from on-chain to off-chain. Lightning Network [77] consisting of an off-chain transmission network provides a fast and privacy-protected transaction environment. Only the process of establishing a two-way payment channel involves on-chain transactions, while other transactions that occur in the channel are all off-chain.

2.2 Smart Contract Security

Ethereum [2] is the first platform that support smart contracts as the runnable script to execute a series of logical operations. Some platforms, such as bitcoin [1] and Hyperledger Fabric [7], have compatibility about deploying smart contracts on platforms. Smart contracts are implemented on



Level	Vulnerabilities
Blockchain	Timestamp dependency Transaction-ordering dependency
EVM	Re-entrancy Code injection Short Address
Solidity	Mishandled exception Integer Overflow/Underflow Dangerous DelegateCall Frozen Ether Tx.origin Denial of Service

Figure 2.1: Ethereum Smart Contract Vulnerabilities

Ethereum and based on a script language called *Solidity*.

The smart contract is the core of DApps logic control, therefore its security is vital. The security of smart contracts can be broken down into two components: logic security and code security. The flash loan vulnerability [78], for instance, is an example of the contract's logical security. Due to the developer's singular pricing in the coin swap section, the price measurement contains logical flaws. The reentrancy vulnerability [79] is the result of inadequate code security, which enables an attacking contract to repeatedly call the same function. Before designing a complete system for distributed data sharing, we should comprehend the security issues in the smart contract and be able to prevent their occurrence.

In a decentralized data-sharing application, the smart contract controls the product logic. According to Purathani's work [80], there are three categories of vulnerabilities in smart contracts, which are blockchain vulnerabilities, software security issues and Ethereum and solidity vulnerabilities. In this section, we will introduce several common vulnerabilities of smart contracts. We will analyze the causes of the vulnerabilities, how the attackers exploit the vulnerabilities, and introduce relevant preventive measures. As shown in figure2.1, we classify these vulnerabilities into three layers, including the blockchain layer, virtual machine layer and solidity layer.

2.2.1 Timestamp Dependency

Timestamps are an essential component of blockchain networks. It can be considered an anti-fake tag that helps miners determine whether a new block is legitimate. While packaging the transactions into a new block, the miner will include a timestamp. Typically, the timestamp is based on the system time of the mining device. Although the block timestamp appears to be random, the Ethereum blockchain allows a timestamp tolerance of 900 seconds so that miners can set the timestamp within a short interval while remaining legal. Thus, if the timestamp is used as the conditions in a smart contract or to generate random numbers for important operations (such as the transfer of Ether), a malicious miner can circumvent the validation and profit by slightly modifying the timestamp. In a nutshell, the timestamp dependency vulnerability is a potential security issue brought by using the *block.timestamp* as a condition to determine the execution of an operation in a smart contract. The prevention techniques of timestamp vulnerabilities include avoiding using *block.timestamp* to generate random numbers, or avoiding it being a key decision condition.

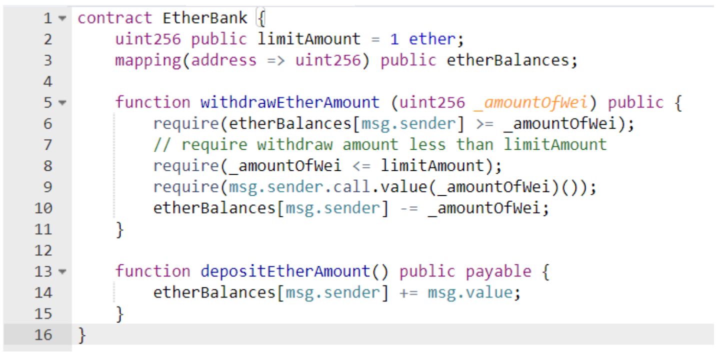
2.2.2 Transaction-Ordering Dependency

Transaction-ordering dependency is also referred to as race condition and front running. Front running refers to obtaining large transactions that may affect asset prices from non-public information in advance, in order to buy and sell first for profit, which has been a problem in traditional financial markets for a long time. In the blockchain system, miners are responsible for selecting and encapsulating transactions into blocks, and since the block state is dependent on the transactions, miners determine the block state. The miners order and include transactions into new blocks according to rewards, the higher the *gasPrice* is, the faster the transaction will be processed. A contract may be vulnerable to TOD attack if the decision condition is based solely on the transaction ordering.

For instance, a puzzle contract may award 100 Ether to the first account to discover the solution. An adversary is able to observe all participants in the trading pool and verify whether their answers are correct. An adversary can observe everyone who submits answers to the transaction pool and validate their validity. If the attacker finds a transaction with the correct answer, he can then submit another transaction including the correct answer with a higher *gasPrice.*According to Ethereum's gas mechanism, the attacker will receive the 100 Ether reward, while the first solver will receive nothing. The block packaging and gas pricing mechanism introduces a transaction ordering dependency vulnerability. It can be avoided by some best development practices, such as setting the upper bound of the *gasPrice*, or hiding transaction information.

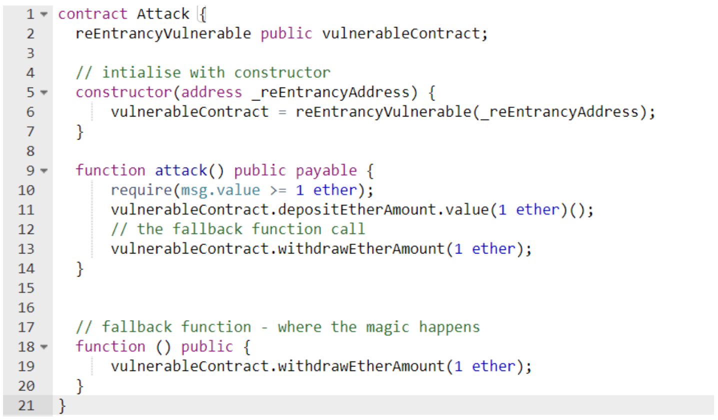
2.2.3 Reentrancy

Re-entrancy is a typical Ethereum smart contract vulnerability. Certain operations cannot be completed in a single smart contract on the blockchain. Currently, calls between smart contracts are necessary. External call is a feature of smart contracts on Ethereum. Externally calling and utilising smart contracts is possible for users, and these calls typically involve Ether sending operations. The smart contract contains a fallback function with no parameters or return value. Whenever the contract accepts the user's transfer and receives Ether, this fallback function will be activated. If the fallback function contains code that calls itself, the user will be able to re-enter the contract. These external calls can be exploited by adversaries to transfer funds by executing the withdrawal function



```
1  contract EtherBank {
2      uint256 public limitAmount = 1 ether;
3      mapping(address => uint256) public etherBalances;
4
5      function withdrawEtherAmount (uint256 _amountOfWei) public {
6          require(etherBalances[msg.sender] >= _amountOfWei);
7          // require withdraw amount less than limitAmount
8          require(_amountOfWei <= limitAmount);
9          require(msg.sender.call.value(_amountOfWei)());
10         etherBalances[msg.sender] -= _amountOfWei;
11     }
12
13     function depositEtherAmount() public payable {
14         etherBalances[msg.sender] += msg.value;
15     }
16 }
```

Figure 2.2: Re-entrancy Vulnerable Contract



```
1  contract Attack {
2    reEntrancyVulnerable public vulnerableContract;
3
4    // intialise with constructor
5    constructor(address _reEntrancyAddress) {
6        vulnerableContract = reEntrancyVulnerable(_reEntrancyAddress);
7    }
8
9    function attack() public payable {
10       require(msg.value >= 1 ether);
11       vulnerableContract.depositEtherAmount.value(1 ether)();
12       // the fallback function call
13       vulnerableContract.withdrawEtherAmount(1 ether);
14   }
15
16
17   // fallback function - where the magic happens
18   function () public {
19       vulnerableContract.withdrawEtherAmount(1 ether);
20   }
21 }
```

Figure 2.3: Re-entrancy Attack Contract

in the fallback function repeatedly. Several attackers exploit this flaw to steal Ethernet from smart contracts. The DAO attack [81] is one of the most representative examples, which eventually leads to the hard fork of Ethereum. The following example briefly illustrates how re-entrancy attack happens.

As shown in Figure 2.2, there is a contract called *EtherBank*, which allows the depositor to withdraw up to 1 ether. There is a public function *withdrawEtherAmount* to allow the sender to specify the Ether amount of Ether to withdraw. When the sender's withdrawal amount passes a series of validations, the code in the ninth line will send the Ether to the user, and the re-entrancy vulnerability occurs here.

Figure 2.3 is an example of an attack contract. This contract calls the transfer function of the target contract and then re-enters the target contract itself through the fallback function for recursive calls. An attacker can instantiate a contract object by providing an address of *EtherBank* contract. The attacker will call *attack()* function to store certain amount of Ether. Once the function is called, the following happens:

- attack.sol - line 11: The function *depositEtherAmount()* will be called. Currently, the attacker deposit one Ether into the *EtherBank* contract.
- attack.sol - line 13: Here, the attacker try to withdraw one Ether from the *EtherBank* contract.

Theoretically, the *EtherBank* contract will return one Ether to the attacker's account.

- attack.sol - line 18: Due to the fallback function, the *withdrawEtherAmount()* function will be continuously called until the victim's contract balance come to zero.

The final result is that the attacker only deposits one Ether but can withdraw all Ethers in the account. To sum up, the main conditions for re-entrancy vulnerabilities include 1) Ether transfer happens before the modification of storage state variable; 2) The use of CALL instruction for Ether transfer; 3) the transfer target contract has a malicious fallback function. The re-entrancy vulnerability can be avoided by changing the order of Ether transfer and storage state modification. The official solidity security guide also recommends to use *send()* and avoid *call.values* for Ether transfer.

2.2.4 Short Address

Short address attack signifies that the attacker makes a contract call by constructing an address with zero at the end, and then rounds off the zero at the end of the address in the call parameters in order to use the virtual machine's automatic data completion mechanism to shift and amplify the second parameter. The exchange is commonly the target of short address attacks. If the user initiates the transfer of the contract in the exchange while using the maliciously constructed short address as the target. Due to the short address vulnerability, the actual transfer amount will be multiplied multiple times if the exchange does not verify the length of the user input, resulting in a significant capital loss. The primary cause of the short address vulnerability is that, when reading the contract call input, the virtual machine automatically fills the end of the field whose length does not meet the requirements, leading to data ambiguity and parameter shift expansion.

2.2.5 Mishandled Exceptions

This vulnerability is also known as exception disorder or unchecked call return value. Many operations in Ethereum smart contracts usually require interaction between contracts. The interaction between contracts can be carried out through two methods: 1) directly calling the function of the callee contract; 2) using external message calls (through *call*, *delegatecall*, *send* instructions). If the first method is used to directly reference the function of the callee contract, the exception can be handled correctly, and the contract will be terminated and reset to the state at the beginning of the call. However, when using an external message call, the exception message is not propagated back to the caller contract. If the exception is not handled correctly, the contract may continue to be executed and cause capital loss. Multiple conditions can trigger exceptions, and this vulnerability can be classified into sub vulnerabilities:

Gasless send: When using *send* to transfer Ether, the fallback method of the receiver has a gas limit of 2300. If the fallback function of the receiver contains many instructions with high gas consumption, an out-of-gas exception will be triggered that there's not enough gas to complete the operation. If there is no exception handling mechanism, the contract may still believe that the send operation is successful and the balance will be updated, but the Ether is not actually received.

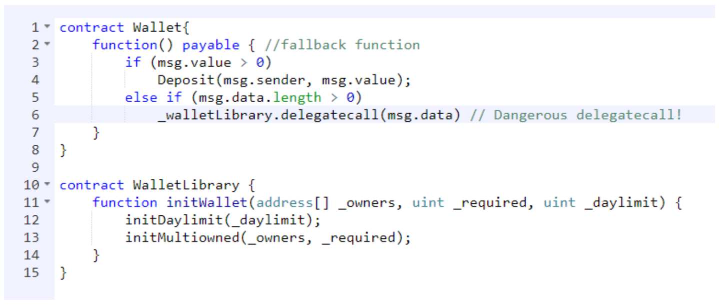
Call stack depth limit: Note that this problem has been resolved after the implementation of EIP-150 [82]. In the original EVM structure design, the maximum stack size was set to 1024 [2]. When one contract calls another, the depth of the call stack increases by 1. If the stack depth reaches the maximum limit, an exception will be thrown. According to Luu's research [83], the call stack depth of normal contract invocations will not exceed 50. However, an attacker can deliberately make the stack depth exceed the limit by filling the call stack in advance through multiple contract calls, transactions, resulting in contract call failure. If the exception is not handled correctly, the victim contract will continue to be executed, and the attacker can benefit from it. After Ethereum Tangerine Whistle hard fork, the gas calculation method of external call has been modified, and the call depth will never exceed 1024 frames.

In conclusion, the mishandled exception vulnerabilities refer to a series of vulnerabilities caused by the failure to explicitly check the return value (i.e., whether the call operation was successful) when employing certain low-level call methods for intercontract invocations. To avoid the state confusion caused by inconsistent exception propagation, preventive techniques include the code that developers add to the caller contract to check the operation's result. Although this exception propagation mechanism and its subclass call stack depth limit are EVM-related characteristics, since the root cause of the vulnerability is improper programming, we classify it as a solidity-level vulnerability.

2.2.6 Integer Overflow/Underflow

Overflow and underflow are examples of common integer errors in computer programming languages. Computer integer variables have upper and lower bounds. If arithmetic operations encounter an out-of-bounds condition, two types of integer errors will occur. If the maximum range of the integer type is exceeded, the number changes from its maximum value to its minimum value or returns directly to zero. This is referred to as "overflow"; if the minimum range of the integer type is exceeded, the number changes from a maximum to a minimum value. Underflow occurs when a small number or zero becomes the maximum value. In short, integer overflow indicates that the data stored in the memory unit storing integers exceeds the maximum value that memory unit can store, resulting in overflow. If the developer of a smart contract does not perform a safe overflow detection on the operation's result, an integer overflow error will occur. Errors in the operation of the smart contract programme will be caused by integer overflow errors, allowing attackers to bypass checks and perform operations such as balance tampering, which can result in significant losses.

In the Solidity language, the step size gap of each integer type is 8 bits, which supports from uint8 to uint256, and int8 to int256. The largest unsigned integer type is uint256, and the maximum supported value is 2^{256} -1. When a uint256 variable reaches this value, adding 1 will make it back to zero. Integer overflow is a very common type of security vulnerability in smart contracts, especially



```
1  contract Wallet{
2      function() payable { //fallback function
3          if (msg.value > 0)
4              Deposit(msg.sender, msg.value);
5          else if (msg.data.length > 0)
6              _walletLibrary.delegatecall(msg.data) // Dangerous delegatecall!
7      }
8  }
9
10 contract WalletLibrary {
11     function initWallet(address[] _owners, uint _required, uint _daylimit) {
12         initDaylimit(_daylimit);
13         initMultiowned(_owners, _required);
14     }
15 }
```

Figure 2.4: Parity Wallet code example

ERC20 token contracts. Attackers can use this vulnerability to pass in some specific values to skip some conditional verifications to make illegal and malicious requests (such as transferring a large amount of token). Some real-world cases include the BEC smart contract attack of Beauty Chain [84], and the SMT project attack in 2018. At present, in order to prevent the occurrence of integer overflow, on the one hand, verification can be performed before and after the arithmetic logic. On the other hand, SafeMath [85] library (including four safe methods, add, sub, mul and div) which is one of a set of smart contract libraries maintained by OpenZeppelin can be used.

2.2.7 Dangerous DelegateCall

In the Solidity language, the call function is similar to delegatecall. The difference is that delegatecall only uses the code of a target address but executes in the contect of the calling contract. The purpose of delegatecall is to use library code stored in another contract. Through delegatecall, the function of multiple contracts calling the same library contract can be realized, avoiding repeated deployment to save gas consumed during the deployment process. However, delegatecall is a dangerous function, by passing in specific parameters, the attacker can fully manipulate the current contract state (such as balance, storage variable). Figure 2.4 shows a simple example. In the Wallet contract, line 6 calls delegatecall and passes the parameter msg.data, which allows the attacker to call any public function in walletLibrary. Therefore, the attacker can call initWallet in line 10 to become the owner of the wallet contract, and then can transfer Ether from the wallet contract to his own address. In July 2017, this vulnerability was revealed in the Parity Multisig wallet contract [86]. The attacker stole over 150,000 Ether from three high-security multi-signature contracts, which were valued at approximately $30 million at the time. This vulnerability not only allows attackers to steal user assets, but if an attacker executes the kill command to remove a library contract's code from the blockchain, other contracts utilising the library contract will no longer be able to function, resulting in frozen funds that cannot be transferred out. In November of the same year, Github user devops199 claimed to accidentally kill the Parity multi-signature library contract [87], resulting in the freezing of 500,000 ETH at a price of approximately US$150 million.

2.2.8 Frozen Ether

This vulnerability is also known as greedy contract, a term that is simpler to comprehend. Due to the inability to transfer Ether out or the failure of the dependent external transfer function, the contract becomes greedy and results in the loss of user funds. Smart contracts play a crucial role in the management of digital assets on blockchain platforms. Typically, Ethereum Smart Contracts use Ether to manage digital assets. Numerous contracts implement high-value digital asset usage scenarios through Ether transactions with users involving digital assets. Unlike standard Ethereum accounts, smart contract accounts do not have their own private keys and can only manage their Ether through contract codes (contract balance). Certain contracts are utilised to accept Ether and transfer funds to other addresses.

If the developer only codes for the function of receiving ETH when creating the smart contract, but does not provide a function that allows ETH to be transferred out, or if the transfer out function cannot be triggered, the ETH assets stored in the contract will be permanently frozen and cannot be transferred. In addition, similar to 2.2.7, if the contract itself does not implement a transfer function by itself, it uses delegatecall to call some transfer functions in other contracts to realize the Ether transfer out function. Once these contracts that provide the transfer function perform suicide or self-destruct operations, then the contract that calls the transfer function through delegatecall may be frozen.

2.2.9 Tx.origin

The tx.origin global variable in Solidity traverses the entire call stack and returns the address of the account that initiated the call (or transaction). Using this variable for authentication in a smart contract exposes the contract to attacks similar to phishing. The practicable defensive measure is to use tx.origin as little as possible. Not that tx.origin variables are never usable. There are valid applications for this variable. For instance, if we want to restrict the ability of external contracts to invoke this contract, we can use require verification statements to prevent intermediate contracts from invoking this contract and restrict access to this contract to common codeless addresses only.

2.2.10 Denial of Service

Denial of service refers to the intentional destruction of normal services, which prevents users from accessing or utilising them. It is a common security flaw in the conventional realm. The access denial vulnerability in the smart contract will prevent the contract from performing the corresponding functions as intended. Due to certain characteristics of smart contracts (such as their inability to be repaired via update), smart contracts are more susceptible to access denial vulnerabilities. Once some contracts are subjected to a denial of service attack, they will be permanently paralysed and unable to be called, which may result in some ether becoming permanently trapped in the contracts. There are numerous reasons for access denial to smart contracts, including external call failure, gas reaching the block limit, unanticipated self-destruction instructions, etc. Typically, these reasons can be avoided with secure code specification.

2.2.11 Uninitialized Pointer

The EVM stores data either as storage or as memory, and it is possible to generate insecure contracts by initialising variables improperly. Depending on their type, local variables within functions default to storage or memory. Uninitialized local storage variables may point to other unexpected storage variables in the contract, resulting in intentional (i.e., the developer places them there on purpose to exploit them later) or unintentional vulnerabilities. The Solidity compiler issues warnings for uninitialized storage variables; therefore, developers constructing smart contracts should pay close attention to these warnings. When dealing with complex types, developers should explicitly use the memory or storage keywords to ensure that they behave as expected.

Chapter 3

ArtChain: Blockchain-Enabled Platform for Art Marketplace

Blockchain is an emerging technology with the potential to revolutionise the global industry and create a network of trusted business relationships between multiple parties. While blockchain has been applied to a variety of practical use cases, one industry in which it is a natural fit is the art industry, due to the way art forensics and transactions are conducted, tracked, and recorded. About 3% ($6 billion) of the estimated $200 billion annual art trade is attributed to illicit cultural property, which includes counterfeits, forgeries, smuggled goods, and outright theft. With the aid of Blockchain, there is a strong possibility of neutralising global revenue loss threats. In light of this, we have developed the ArtChain platform to aid the Art Industry as a whole. It is the first blockchain-enabled art trading platform to be implemented in Australia. It offers a transparent, privacy-preserving, and tamper-resistant transaction history for authentication, tracing, and provenance of art assets.

3.0.1 Blockchain Business Network For Art

In this section, we start with the rationale behind the use of blockchain for an art marketplace, and then discuss the benefits of this blockchain-enabled platform.

Rationale behind Using Blockchain

The major participants in this blockchain business net- work are described in the following.

Artist Established artists along with new generation artists all have equal opportunities for professional evaluation and to publish their artworks. Published items will be available for trade.

Art gallery After artwork is registered on the blockchain system, it can be tracked and located in real-time giving an additional level of security to galleries around the world.

Auction house Data stored on blockchain can be synchronized with auction houses, opening up new channels for a greater global audience participation. More traffic equals more opportunities for all involved.

Collector Online synchronization of collectors offline art assets. This proves ownership and sets the provenance of the pieces for future generations ensuring the value is preserved. Access to a global database with extensive filtering capabilities.

Benefits of a Blockchain-enabled Art Marketplace

- Artwork authenticity and traceable data: When it comes to art collecting, provenance is crucial. Therefore, a distributed ledger can be used to trace the transfer of ownership over a period of time and serve as a decentralised database securing provenance data and other vital information related to artworks. The absence of a record of the ownership history of a masterpiece often raises suspicions that it is stolen or a forgery. This facilitates the swift and indisputable transfer of ownership in trading.
- Royalty payment from secondary market for artists: a 5% royalty will be payable to visual artists on certain commercial sales of their work. This entitlement is created by the Resale Royalty Right for Visual Artists Act 2009. However, due to the difficulty in tracking the resales in the current art market, often artists do not necessary get the royalty.
- Blockchain audit trail: Helps in detecting tax evasion and money laundering. Add-on analytic or AI services can predict the current value of an artwork based on shared transparent data. This helps primary market valuation, which is more difficult and more speculative than secondary market due to a lack of market history.

Furthermore, ArtChain is an open and expandable blockchain infrastructure orientated towards the art industry. This means that participants will have the opportunity to develop an extensive range of art-related applications for specific scenarios based on the foundation of ArtChain.

3.0.2 System Overview

In this section, we first present the foundational principles the architecture is based on, the high-level architecture and its main components. Then, we present basic data model design and the trading process of the platform. In addition, we discuss the trust and security issues.

High-level Architecture Design

We evaluate multiple blockchain platforms before deciding which platform to implement. Based on business requirements and a technical evaluation, we've decided to use the Ethereum1 private blockchain and Proof of Authority (PoA) as the consensus algorithm. Initial consideration was given to utilising Hyperledger Fabric to implement our system due to its capability, popularity, and maturity.

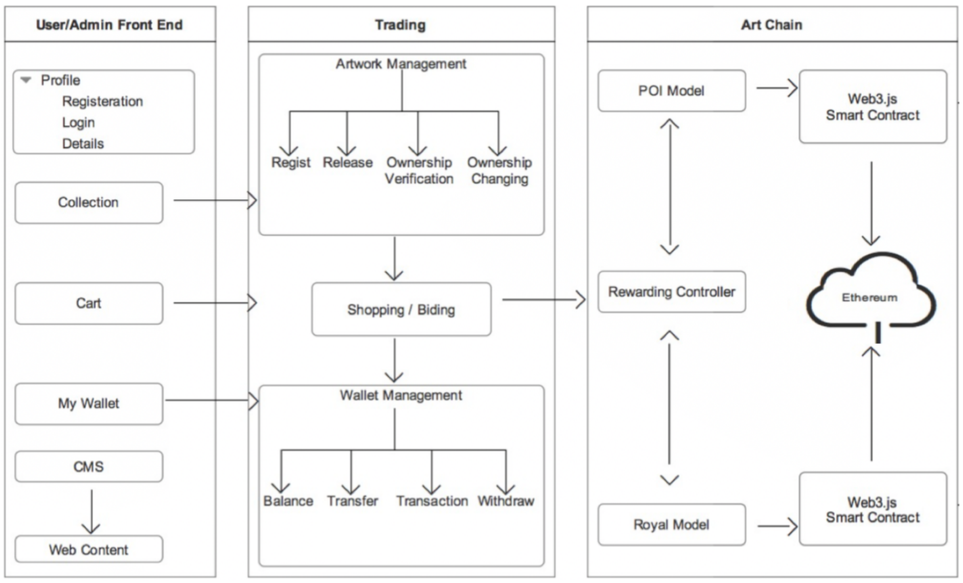


Figure 3.1: System Architecture

However, it lacks support for native tokens, a key business requirement in our design as the art trading platform seeks to integrate the payment and ownership transfer processes. To accomplish this business requirement, we design and implement an ACGT utility token. We adopted micro-services architecture for the following benefits:

- Allows quick parallel development of various components in the application landscape.
- Reduces discussion time between various groups developing various components.
- When done properly, provides clean reusable interfaces.
- When done properly, reduces handshaking in interfaces.
- Reduces the risk and time of integration/chain testing.

The architecture design is shown in Figure 3.1. There are three layers in the system: the user front end, the trading back end, and the ArtChain blockchain layer.

User Front End includes the following functions: man- aging Profile for user registration, login and user details; displaying art Collection; shoping Cart; user Wallet; and CMS (Content Management System) to create and manage web content.

Trading Back End consists of Artwork Management, Shopping/biding, and Wallet Management. Artwork Management includes artwork registration, ownership verification and ownership transfer. Artists or collectors conduct the registration of their artworks through the assessment system of professional institutions within ArtChain. Their works of art will then be eligible for trading and participating in the ecosystem.

ArtChain Blockchain including the following components: Royalty model: responsible for artists royalty payment in the resale of their artworks. POI model: manage Proof of Interaction (POI) agreements, which are used as incentives to grow the ecosystem of applications.

Basic Data Flow

There are three major groups of data objects stored in the distributed ledger as illustrated in Figure 3.2: User contains all information related to a user's profile, login, wallet, and auction events attended. An artist is also a user, with additional information and verification. Artwork consists details, tag, history of ownership trans- fer, and order details. Trading combines an order with the artwork and the buyer's basic information and the shipping address. The trading process is shown in Figure 3.3. When the user's trading request is received, Profile Services and Trading Services are triggered to retrieve the customer info and trade info. After checking the trading conditions, Payment Services are responsible for handling payment. Then Reward Services are called to request and receive the reward information. In the end, Shipping Services handle the shipping information.

Trust and Security

ArtChain collaborates with specialised or prominent partners in the primary or secondary markets of the art industry, such as museums, art galleries, and auction houses, to establish the original ledger nodes and provide core functions including validating, ordering, and generating blocks of transactions. Together, these ledger nodes and other agent or routeing nodes safeguard the blockchain network.

We use Proof-of-Authority (PoA) as the trust model of ArtChain network. PoA is well-suited to regulated industries where entities are responsible for maintaining the network (known as authorities), rather than remain anonymous as in mining-based chains.

For our business purpose, well-known museums and art galleries are acting as authorities in ArtChain network to conduct authenticity and price assessment for an artwork. They are called SuperNode. Supernodes perform validating, block generation and publishing.

When a node exhibits anomalous behaviour, other nodes in the network will detect it immediately if it is attempting to engage in malicious behaviour or is under attack (e.g., sending illegal transactions, traffic attacks, and data tampering). The network will immediately isolate and warn about the offending node. ArtChain deploys ledger nodes throughout the primary and secondary art markets, including internet companies, cloud service providers, and a large number of collectors of works of art and artists, which, from a probability standpoint, eliminates the likelihood that the majority of nodes fall victim to an attack or collude to engage in malicious activities.

In the beginning, we set up 100 nodes to provide sufficient redundancy and defend against a 51% attack. Currently, they are deployed on AWS and Ali cloud, but they are not activated simultaneously. We monitor the behaviour of the nodes and replace them dynamically if they fail. We intend to increase the deployment's decentralisation on other clouds. The trade-off between decentralisation

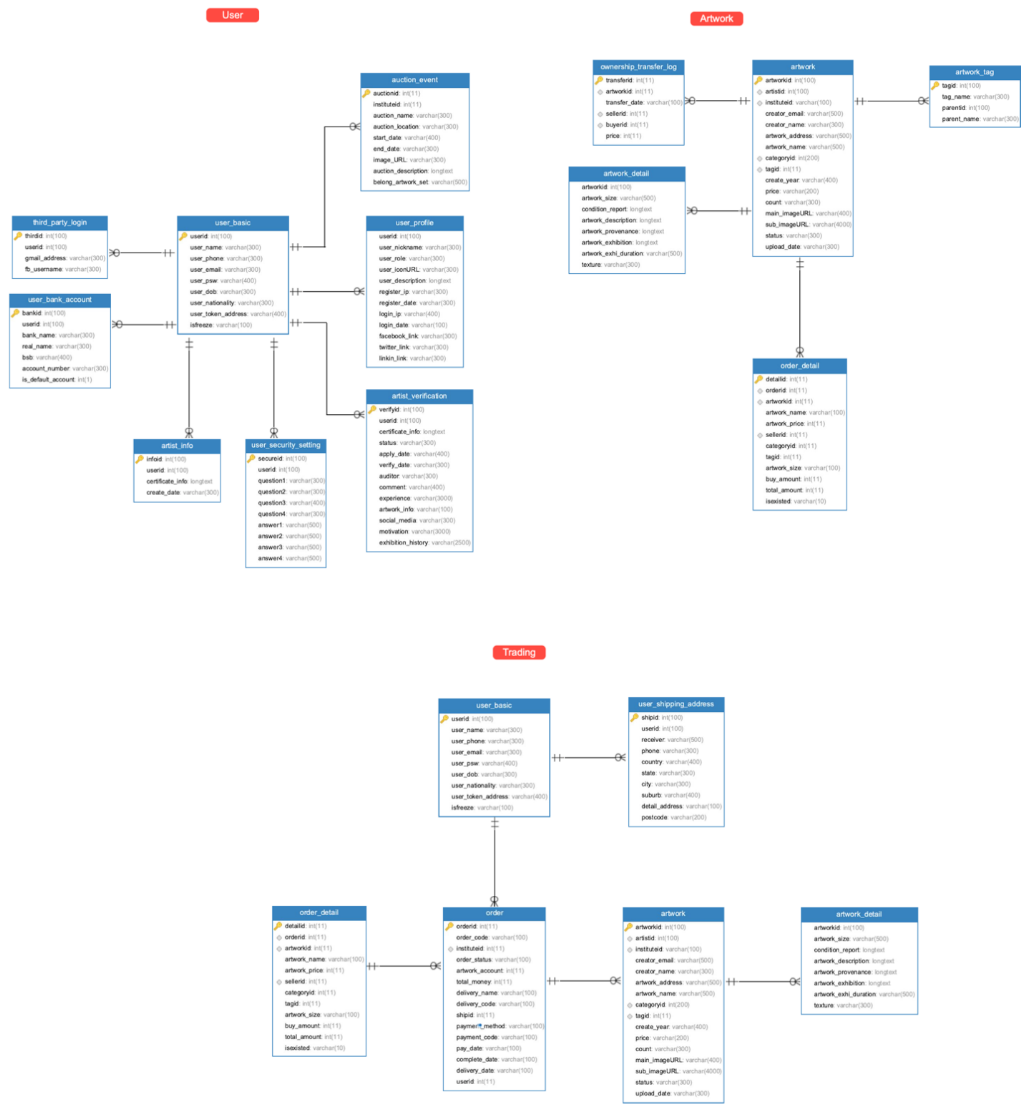


Figure 3.2: Data Model

and performance is an important factor to consider in this regard.

Transaction security ArtChain network assures the security of users' accounts and funds by using blockchain consensus, digital signatures and end-users encrypted wallets. The artwork trading platform provides security services that are likened to those offered by financial institutions. It integrates data, applications and transactions in blockchain clouds through the efficient integration of data storage, net- work and other resources, so as to create a secure transaction environment.

Financial Management ArtChain maintains high standards of integrity and ethical business conduct and is in compliance with relevant laws and regulations, as well as self-regulatory principles of the industry. We also implement a component to conduct the regulatory duty of Know-You-Customer (KYC).

3.0.3 Scalable Blockchain Implementation

In this section, we present the implementation of ArtChain network. We first introduce tokenization and how it works in our system. Then we describe the design and implementation of upgradable smart contract for the purpose of improving function and fixing bugs. Finally, we discuss how to preserve privacy and confidentiality as required by regulations and business needs.

Tokenizaiton

Tokenization is the process of converting an asset into a digital token on the blockchain so that ownership of the asset can be transferred through smart contracts. Smart contracts include functions for automatic transactions, formulas for calculating asset prices, and other specialised characteristics. Tokenization is not just the act of creating a token. Instead, the focus is on the design of the entire system, which includes an awareness of the various rights and issues.

ArtChain contains two types of tokens: security token ACG and utility token ACGT. ACG token possesses the essential technical characteristics of digital currencies, such as a steady issue curve, free trading, immunity to double-spending attacks, and a transaction history that can be traced. These characteristics are protected by the distributed ledger architecture and smart contracts. We develop E-wallets for corporate and institutional users that incorporate all the necessary features for interacting with ArtChain applications. ACG token provides incentives for the maintenance of the ArtChain network and application ecosystem.

Network Maintenance the consistency of ArtChain network is jointly assured by ledger nodes. Ledger nodes will have the opportunity to be awarded with ACG as block rewards and transaction fees, to encourage them to contribute to the security and stability of the ArtChain network.

Ecosystem of Applications ArtChain will reward users with newly added ACG in positive correlation within a certain cycle based on a number of indicators, including their frequency of interaction with the ArtChain ecosystem, contribution levels, influence, and the number of ACG coins they hold. All ecosystem incentive indicators are quantifiable and verifiable, and ledger nodes collect and calculate them. The allocation of incentives will be governed by Proof of Interaction (POI) agreements.

The ratio of the incentives for ArtChain network maintenance and the incentives for the ecosystem of ArtChain applications will be dynamically adjusted by using a negative feedback mechanism to maintain the balance and stability of the ArtChain network and the ecosystem of applications. The specific indicators and algorithms will be published before any main relevant applications go online, and will be implemented and operated through open rules of contracts. Relevant institutional users of the ecosystem (art galleries, museums, auction houses and artists) will be consulted.

The utility token ACGT is used only internally to facilitate art trading payments. It is a type of stable coins, which are intended to have a stable price or value over time and are, as a result, less volatile. On the one hand, these coins aim to mimic the relative price stability of fiat currencies, while on the other, they retain the core values of cryptocurrencies, such as decentralisation and security. Each ACGT token is backed by 1 AUD in fiat currency held by a central custodian. Holders are guaranteed the ability to exchange their tokens at any time for a stable value denominated in fiat currency.

Upgradable Smart Contracts

Smart contract, once deployed into the block chain, is immutable literally. In consideration of bug fix and function improvement, lots of work has been done to propose an upgrdeable design pattern of the smart contract. The typical methods include:

- Separate logic and data
- Partially upgradable smart contracts system
- Separate logic and data in key value pairs
- Eternal storage with proxy contract

Among these methods, the proxy mechanism is most flexible and guarantees a 100% upgradable mechanism i.e., the logic could be completely modified while remaining the existing data state. In our system, we refer to Zeppelin's proxy patterns (Figure 3.4) and implement the so-called Unstructured Storage Pattern. The contract structure is shown below:

By using this pattern, the system achieves following features:

- General user is unaware about upgrade of the contract.
- Implementation of logic contract is 100% upgradable.
- Data is stored in proxy contract. New data fields could be added by the upgraded logic contract, without touching existing data structure.

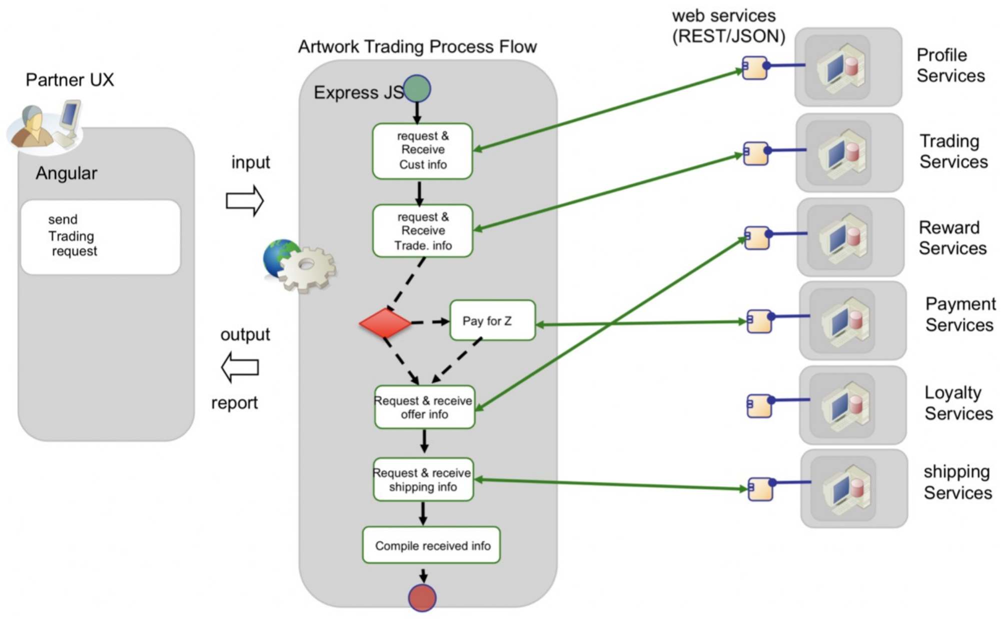


Figure 3.3: Trading process

This design has been selected for the ZeppelinOS smart contract system and has passed a comprehensive security audit. In our implementation, we also address the problem of initialisation. This is a long-standing issue with Ethereum upgradability solutions. Our objective is to develop an upgradeable logic contract, and we deploy a proxy contract that delegate to a previously deployed logic contract on the blockchain. Therefore, the proxy contract has no chance of establishing the initialising steps in the function Object() [native code] of the logic contract. To correctly initialise the proxy contract, we must take special measures. Utilizing an initializer function instead of the function Object() [native code] and ensuring that it is only executed once for the required initialisation process is our workaround. Other proposal could be found as Initializer Contracts [10].

Privacy and Confidentiality

ArtChain makes public all ledger nodes and their state in the network in real time. The transaction history (block content) and state information in ArtChain blockchain and ledger is publicly visible. However, in case of any privacy requirements for some transactions, such privacy information will be processed. In the data model design, we carefully decide what data to be stored on-chain and what off-chain. The design has been evolved along business needs and regulatory needs. Currently, the on-chain data store contains information on artist, the hash of ownership, price, and history. The hash of ownership protects the privacy for owners who do not want to be known to the public, as well as in compliance with privacy regulations, such as the General Data Protection Regulation (GDPR).

3.0.4 Performance Evaluation

We conduct extensive performance test of the system. We identify that the performance bottleneck of the system is the low-level I/O efficiency of the Ethereum client, i.e. Geth in our system.

ArtChain private chain, based on POA consensus and 5-second block interval and deployed on 6 cloud nodes (8x2.5GHz CPUs, 32G memory), supports up to 1500 TPS, i.e., 1500 raw transactions on the chain, far more superior to Ethereum main-net (about 15 TPS nowadays). Integrated user actions, like post new artwork or top up tokens, are usually comprise of a series of transactions/queries on the chain. ArtChain on average processes about 40-70 user actions per second.

Environment Setup

The private chain is composed of 3 Ali cloud servers (8x 2.5GHz CPUs, 32GB memory, 64GB hard disc), and 3 AWS cloud servers (8x 2.5GHz CPUs, 32GB memory, 8GB hard disc). Geth version 1.8.17, startup parameter is tuned as:

- –targetgaslimit 4294967295: increase gas limit to 0xFFFFFFFF to seal as many transaction as in one block. Note this need coordinate with the gasLimit in the genesis.json file when creating the chain.
- –txpool.lifetime 24h –txpool.accountslots 65536 – txpool.globalslots 65536 –txpool.accountqueue 64 – txpool.globalqueue 65536: increase txpool so that it stores as many transactions both account specifically and globally as we submitted.

We use web3.js to communicate with the chain, and to monitor its performance we use Wireshark to capture packet for analysis.

Throughput Analysis

As for our private chain, we tried following steps to tune up the system performance:

- Speed up the block generation by changing the block interval when generating the genesis.json. To summary, chain with 2 second interval shows best performance, but 5 second is also acceptable.
- Improve the gas limit of the chain. It does not shows significant improvement on the performance, because gas limit is not the bottleneck of the system.

As our chain is only maintained by 6 cloud servers, we can ignore the effect of network scalability mentioned above. As long as the transactions are sealed, the nodes always have adequate time to keep sync. On the contrary, it is the node's hardware configuration that determines the system performance. We observe frequently crash of Geth client on the node with only 8GB memory originally. Using the node with 32GB memory, the performance is significantly improved, but crash still occurs in certain scenarios.023302+

Geth is thought as a memory monster whose design follows a “I use up what I have” idea, and will use up all available memory on the server. By default our node servers disable the swap and will kill Geth process if it tries to use up the memory. Unfortunately this always occurs. We observed it used up 8Gb memory when trying to create 70 new accounts. A suggestion is to enable the swap on the node, in terms of the sacrifice of the performance. Note the AWS cloud server has only 8GB disc space, and so the swap space is restricted on AWS servers.

Test Results

Raw transaction test:

- Chain is configured with 5 second block interval
- Transaction carries data of 50 bytes, it is a typical value for general transactions.
- Establishes 2000 transactions in about 6-8 seconds per node.
- Establishes 20000 general transaction queries in 2-5 seconds per node.

API based test: APIs such as check_user(), check_transaction() and check_artwork() only query information from the chain and do not include any practical transactions. So they show as high throughput as general queries. It is only depends on the processor and network performance. APIs such as buy_tokens(), post_new_artworks() and freeze_tokens() combine a series of queries and transactions, and these operations usually depend on the result of the precedent, so those APIs have bad parallel performance. For example, post_new_artwork includes 16 low-level operations:

- 2x eth.sendTransaction
- 1x personal.unlockAccount
- 1x eth.estimatesGas
- 2x eth.gasPrice
- 6x eth.getTransactionReceipt
- 2x eth.subscribe
- 2x eth.unsubscribe

Some test results are listed below (Chain is configured with 5 second block interval):

- Establish 116 times API post_new_artworks() within about 18-20 seconds per node.
- Establish 58 times API buy_tokens() within about 5-6 seconds per node.

API add_new_user() contains a low-level operation of personal.newAccount, which uses significant memory and CPU cycles. A typical result is listed below (Chain is configured with 5 second block interval):

- Establish 64 times API add_new_user() within about 20 seconds per node.

Test on different block interval: As all APIs are called simultaneously during the test, we observe that all transactions are actually sealed in a single block. Our supply chain is fully capable of ensuring this. Aside from the block interval difference, these calling procedures require nearly the same amount of network and process processing time. Different block intervals practically result in different throughput for this reason.

Test on node crash: Geth client crashes under cer- tain scenarios. What we notice is that transactions like personal.newAccount(included in API add_new_user()) make Geth consume lots of memory. Got in tests:

- On the node configured with 32GB memory, Geth could support up to 70 concurrent add_new_user() calls.

- Geth crash when submitting more than 70 add_new_user() calls. It is killed by OS after using up all 8GB memory.

We then try to enable 32GB swap on the server, and find that Geth succeeds processing 96 concurrent add_new_user() calls. During the process it used up 32GB physical memory, and then 9.1GB swap memory. As a result, it uses as long as 914 seconds to establish all the calls. As a comparison, it needs only about 20 second for 64 calls without using swap memory. Performance degradation is obvious.

Bottleneck of the System

Based on our performance test, we find out that: 1. The performance bottleneck of our system is at the Geth IO execution. 2. The way to improve the system performance is to improve node hardware configuration. Ethereum uses LevelDB9 as the database to store key/value. The key to accessing database is irregular on account of the discreteness of hash. The LevelDB has an excellent performance in reading/writing continuously, while bad for random key. Therefore, the time t for accessing LevelDB would be longer as the amount of data storage increases. In fact, the test results show that if n is large enough, the value of t will increase and the eciency will degrade largely for some data which not hit LevelDB cache at times. Geth consumes huge memory on certain transactions, e.g., personal.newAccount, and will crash when receiving multiple concurrent memory-consuming transactions. A suggestion is to enable swap memory on the node to improve system stability, at the sacrifice of the performance (See performance degradation when memory is swapped). We suggest 4GB of swap space on the nodes, based on the performance test result. This improves the system stability and does not degrade the performance significantly.

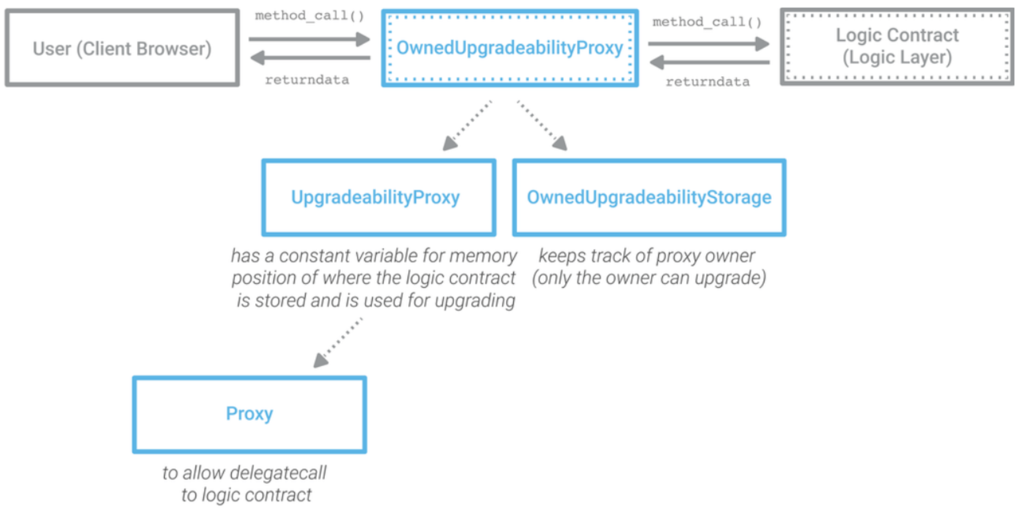

User (Client Browser)
method_call()
returndata
OwnedUpgradeabilityProxy
method_call()
returndata
Logic Contract
(Logic Layer)
UpgradeabilityProxy
has a constant variable for memory position of where the logic contract is stored and is used for upgrading
OwnedUpgradeabilityStorage
keeps track of proxy owner (only the owner can upgrade)
Proxy
to allow delegatecall to logic contract

Figure 3.4: Zepplin proxy architecture pattern

3.0.5 Conclusion

In this chapter, we described how we design, implement and deploy the blockchain platform in operation as a working product in practice. There are several features we plan to further develop in the next versions of the product: 1) anti-counterfeiting for the original works of art by integrating with the smart modules of IoT, and activating relevant smart hardware and other functionality (e.g. positioning/location tracking) as required by artists or collectors; and 2) improving the efficiency of high-value art assets insurance by using smart contracts to automate the process of claim and periodic renewal.

Chapter 4

PPM: A Provenance-Provided Data Sharing Model for OpenBanking via Blockchain

In recent years, open banking has become more and more prevalent in Australia. It aims to facilitate the secure transfer and exchange of users' personal financial data between banks. The financial data's sensitivity necessitates enhanced participant authentication and provenance. To meet the requirements of open banking, we propose a provenance-provided data sharing model (PPM) via blockchain in this chapter. The model employs programmable smart contracts as the intermediary witness between users and third-party services, and provides modifications to the data layer (data content, transaction structure), smart contract layer (access control list, logic), and application layer (customized APIs). Based on this, our PPM model features transparent authentication, privacy-preserving control, and auditable provenance. In the face of open banking, the analyses and discussion demonstrate that our model is a secure and realisable system.

Blockchain, with the properties of unforgeability, tampering resistance, transparency and verifiability, perfectly meet the requirements of an open banking system. Originate from Bitcoin [1] and after a long-time development, blockchain has evolved into a smart-contract supported system [2]. The smart-contract platform enables the predefined laws and rules programmable, which brings significant influence in the finance area. The design of blockchain ensures that no single entity can modify, delete, or even append any record to the ledger without consensus from other network participants, which provides the unforgeability and tampering resistance of data. The distributed system also guarantees transparency and verifiability due to the public readable ledgers.

Based on the requirements of open banking and properties of blockchain [88] [89], we design a Provenance-Provided Data Sharing Model (PPM) for open banking system via blockchain technology.

By factoring the processes of data sharing, decoupling the architecture of blockchain and redesigning the modules and functions, we construct the PPM model compatible with completeness and robustness. Our PPM model achieves the following properties:

- *Transparent authentication:* The authentication of our model is transparent and public on a distributed ledger. The transparency means users and third-party services can freely obtain the records on who authenticates, whom the actions authenticate to, what kind of authenticated actions, etc.

- *Privacy-Provided Control:* The customized access control enables users to control and share personal data with other parties with their willingness. The public key encryption system protects the authentication of actions' control with the users' identities.

- *Auditable Provenance:* The authentication is publicly recorded on the chain, where both user and the third party can get the whole history of their actions. Whenever the powerful authority does evil, the activity will be traced with accountability.

Our PPM model is presented in detail at the following sections, and the comprehensive analyses and discussions are also provided. PPM model, designed for the open banking system, will also give insight for other applications in different scenarios.

4.1 System Illustration

The Facebook-Cambridge Analytica scandal of 2018 heightens public awareness of data privacy. People have given their data considerable consideration. They are concerned that their personal information may have been collected without their authorisation. Currently, two advanced methods are used to resolve data-sharing issues in various contexts. There are some methodologies which adopts different techniques to reach data sharing in different fields. Here we introduce two representative works as follow.

In Gan's paper [46], they proposed a structure about using PKI specified in the Internet of Things(IoT). This system is an authentication system based on key management. Traditionally, some third-party institutions maintain CA, which may exist trust issues. Gan's approach builds a private blockchain to store and manage all nodes' latest public keys. Device Manufacturer Validator(DVMs) is the entity that connects CCA and IoT devices. The IoT Manufacturers maintain DVMs, and all IoT devices are connected. In the beginning, a new DVM requests the CCA (centralized certificate authority) to authenticate and joins the blockchain network. When passing the authentication, DVM's public key, hash address, and CCA's signature will be packaged and broadcasted to the blockchain network. Similarly, new IoT devices can register into the network by creating transactions in the same way. Moreover, DVMs can update or invoke IoT devices' public keys by sending transactions.

Zyskind [56] proposed a new system which combines and off-chain storage. This data management service has a good performance on privacy-preserving. There are three roles in this system, which are mobile users, service providers and blockchain nodes. Users are keen on using the service or obtaining applications. Service providers act as middlemen to request personal data and processing data according to users requirements. Blockchain nodes are the institutions which maintain the blockchain network and store private key-value data locally. There are two kinds of transactions in this model, which are T_{access} and T_{data}. T_{access} focuses on users' permission allocation and data management. T_{data} is designed for data storage and sharing. Users and service providers can overwrite a new set of permissions with the correct signature in T_{access}. The T_{data} provide encrypted data for those authorized users and service providers. Hence, unauthorized users have no chance to decrypt the data package. Zyskind's methods provide new ideas on data privacy. Through applying blockchain on the traditional data-sharing model, the privacy issues can be solved to some extent.

Both of the aforementioned works focus on distinct aspects. Gan's method describes how to improve the authentication and verification of IoT devices in the blockchain. The work of Zyskind introduces a privacy-focused model that uses two distinct transactions to accomplish permission allocation and data storage. Consequently, we propose the PPM model with provenance, data verification, and user privacy in open banking services. Initially, there are three points we need to discuss:

- **Business Model:** For the commercial banks, open banking service will break their traditional profit model. Data transparency enables bank data to public access. Although they can earn profit from the new model, it is hard to define the incentive regulation that meets all banks requirements.
- **Data Security:** Traditional bank system is a closed system, and users sensitive data circulate among banks internally. With the popularization of an open bank, third-party institutions can visit some users private data for commercial purpose. Only if the open banking service can promise the privacy and security of users data, it can be widely accepted.
- **Technical Risk:** Due to the data is sharing between various institutions, more entities will own the duplicate. In this case, sensitive information is more likely to be hacked or visited without authorization. Also, frequent data access will cause data interception.

4.1.1 System Model

Our model have been proposed to solve the sensitive data sharing in the open banking system. There are three entities in our framework: user, third-party service, and smart contract, as presented in Figure 4.1. The illustrations and the behavior of three components are stated as follow:

- **User:** A user is the entity who owns bank account and interest in sharing personal data with authenticated institutions. Users public keys are submitted to the blockchain ledger. Any

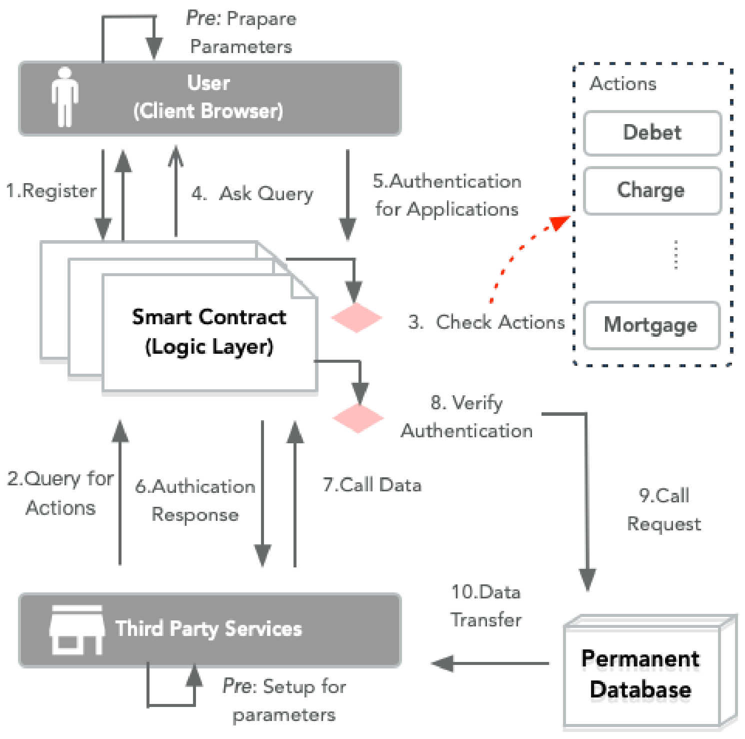


Figure 4.1: Framework of PPM

update of the public key is sending through transactions.

User end behavior:

Generate the public key and identity
Register identity into the contract for records
Authenticate the access of different actions

- **TPS end behavior:** Third-party service provider are the institutions who provide service and applications for bank users. They process and analyze users' bank data to achieve business purpose.

  TPS End:

  $TPSEnd$: Setup for the parameters
  Register for the Actions
  Call the data

- **Smart Contract** Smart Contract is the entity responsible for logic control. Users registration, data request, and provenance recording are all managed by the smart contract.

  Smart Contract logic:

  Check the registration of users
  Check the existed actors
  Check the authentications of TPS

We combine blockchain ledger and offline data storage together to achieve bank data sharing.

4.1.2 Design Goals

- **Membership Authentication and Registration:** Authenticated nodes are responsible for verifying all registration requests. And we implements asymmetric cryptography in the system to validate users' identity.
- **Secure Data Sharing:** Bank data is sensitive because it contains all users personal data. In PPM system, we will build the application programming interface (API) to normalize the data request. Different scenarios call the targeted APIs. In addition, our model restricts the specific nodes to call APIs, which implements the access control of bank data.
- **Data Request Provenance:** All request about personal data will be recorded on the blockchain through the smart contract. The provenance data can help users or banks to audit and track the history and behaviors.

4.2 PPM: A Provenance-Provided Data Sharing Model via Blockchain

4.2.1 Overview of PPM

In this chapter, we design a provenance-provided open banking data-sharing model based on blockchain architecture, as shown in Figure 4.2. There are four layers in the PPM system, which are the contract layer, consensus layer, network layer, and data layer.

- **Data Layer**: The Data Layer consists of service data, provenance data, and block data. These are the three fundamental data types in the PPM system. In particular, provenance data is responsible for recording the history of data transactions on the platform, allowing users to trace the origin of the data.
- **Consensus Layer**:Due to the unique characteristics of the open banking platform, we employ PoA as a consensus algorithm to facilitate the appropriate management of financial data.
- **Contract Layer**:Data from the data layer is placed in the block in the contract layer by implementing data interaction-related functions in the contract. Simultaneously, for some core functions, we verify the user's identity and permission to ensure data security.
- **Application Layer**:The Application Layer is primarily in charge of interacting with the contract Layer. The application layer receives data from the front end and filters it before passing it to the contract layer for in-chain interaction.

We customize the structure in the data layer to store provenance data. Therefore, manipulations on users' bank data can be recorded on the blockchain.

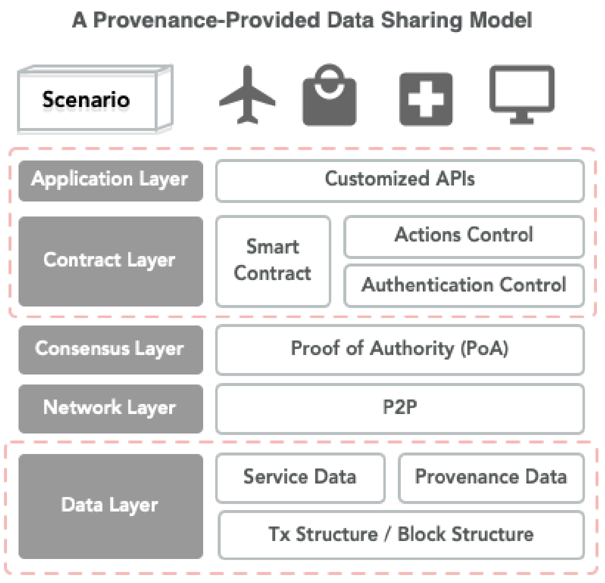


Figure 4.2: Layer-based Architecture

We present the application programming interface (API) in the application layer to control external data access. Only authorised third-party organisations can access bank data and obtain a copy. We use proof of authority (PoA) as the consensus algorithm in the consensus layer. Traditional banks will serve as supernodes, which are tasked with authenticating service providers and validating data requests.

4.2.2 Data Structure

Core to our system is the management of data. In the data-sharing banking system, the highest priority is placed on data security. Consequently, it is crucial to consider which categories of data must be updated and maintained in an open banking system. In the following paragraphs, we will discuss the type of sensitive data we include in our service as well as the format of provenance data.

Service Data: As a open banking data sharing system, consumer data regulation is significant in our model. We choose the data which can identify an individual and prove an individual has account in the bank system as the public-access information. The data includes:

- *Financial Product:* Financial product data includes bank products, bank rates, bonds and other services which reflects what the banks provided.
- *Client Information:* Client data contains users' personal information, which is address, phone number, ID number. etc. These data confirms the unique of user's identity.
- *Account Balance:* Account data includes the account balance and account number.
- *Transactions:* Transaction data includes all history about user's account and how much users spent monthly.

All categories we mentioned above will be store into framework's off-line database. When users send a request to grant authorised service providers, the duplicate of data will response after the verification.

Provenance Data: The provenance of data is required for a distributed system. It is able to report and monitor requests for users' data. Users of the PPM system can monitor the manipulation of sensitive data. For example, if users grant authorised institutions access to their data, they can audit the provenance data to determine if their data has been subject to malicious actions. When a new request is validated according to our model, a form will be generated to store vital information.

Algorithm 1: Registration

```
Input: Registration Query Message(requestForm, pk, userString)
Output: Results
% check the signature of request;
while validSignature(pk, requestForm, id) do
    read current;
    % check the duplication of userString;
    if checkUserString(userString) then
        return "Private String Already Exist";
    else
        addDatabase(requestForm, pk);
        addProvenance(requestForm, currentTime);
        return "Successfully Registered";
    end
end
```

We build the provenance data as the six-tuple, which includes *CREATED TIME*, *FROM*, *TO*, *DURATION*, *VERIFIER* and *TYPE*. The field *CREATED TIME* identifies the time a user creates the request. *FROM* and *TO* presents the hash identity of request initiator and recipient. *DURATION* and *VERIFIER* illustrate the valid time about a data duplicate and node who grant this request, respectively. There are four different values in field *TYPE*, which are "TRANSACTION", "PRODUCT", "ACCOUNT", and "PERSONAL". These values are correspondingly related to the data types as discussed before.

4.2.3 User Layer

The proposed framework bases authentication and encryption service on an identity-based protocol. Our system allows users to generate an asymmetric key based on their bank account and user-id string. In this section, we will demonstrate how to register for an account in our model.

A user who wishes to attend the model must initially set a private string by themselves. The string will be used to generate the pairs of asymmetric keys. The public key will be recorded on the blockchain. Then, when a user sends the registration request, the consensus nodes receive the package and retrieve the public key from the blockchain. In the first algorithm, we illustrate the user registration process. Nodes will initially validate the signature using the public key. The model will then examine the user-specified private string to determine if it has existed in our system. If the signature is valid and the private string is unique, a package containing the user's vital information will be included in the transaction and broadcast to the blockchain network.

The offline database will include all user's personal information. Besides, the registration and timestamp information will form the user's initial provenance, which stores into the blockchain after the process completes.

4.2.4 Third-Party Service

In our model, third-party institutions serve as service providers. They offer a variety of products and recommendations for property management, monthly budgeting, and vehicle registration. When users send their financial information to institutions, service providers can provide them with relevant recommendations. Due to the sensitive and valuable nature of banking data, our model includes access control.

As third-party companies, they must submit the authorisation request to supernodes in order to obtain permission. When supernodes validate the identities and qualifications of institutions, they will package and transmit related data to the blockchain network. Additionally, institutions wishing to obtain a duplicate must request data via remote process call (RPC). Our model includes a series of essential APIs, which will be discussed in the section on security analysis. The establishment of APIs provides a uniform standard for service providers to request financial data. They can extend their products legally to maximise their profits.

4.2.5 Logic Layer

The predefined laws and rules are written into logic and then developed/deployed into the smart contract. The contract holds the critical access control lists, and one is *Registration* list and the other is *Action* list. The lists record the permitted users and corresponding actions. Whenever a TPS intends to invoke/call data, the authentication is processed under the ACLs. Here is a logic to show how the smart contract proceeds.

As demonstrated in Figure 4.3, the interactions between both users and third-party service provider (TPS) are presented. There are three main stages in the workflow: *Prepare*, *connected with smart contract* and *invoke data*. Prepare stage represents the initial preparation of both users and TPS. The second stage provides the interactive steps to the smart contract. The smart contract embodies the logic of predefined rules, where the authentication and ACL are depended on. The third stage is to invoke data from the offline database.

Algorithm 2: Logic in Smart Contract

```
Input: Ask Query Messages
Output: Results
% check the Register List;
while RegisterList(pk, RegsterQuery) do
    read current registerList;
    % check the duplication of RegsterString;
    if checkRegister(registerString) then
        return "Register Already Exist";
    else
        addList(requestForm, pk,id);
        return "Successfully Registered";
    end
end
% check the Action list;
while actionList(pk,TID,id,ActionQuer) do
    read current actions;
    func(mortgage);
    func(debitt);
    func(charge);
    % check the duplication of Actions;
    if chkAct(actionString) then
        return "Action Already Exist";
    else
        Action(actionString,pk,TID);
        return "Successfully Added";
    end
end
% check the Authentication list;
while validSignature(pk, requestForm) do
    % verify the authentication of TPS;
    if checkTPSString(actionString,TID,id) then
        callData(actionString,TID,id);
        return "loadingData";
    else
        return "Not Authenticated";
    end
end
```

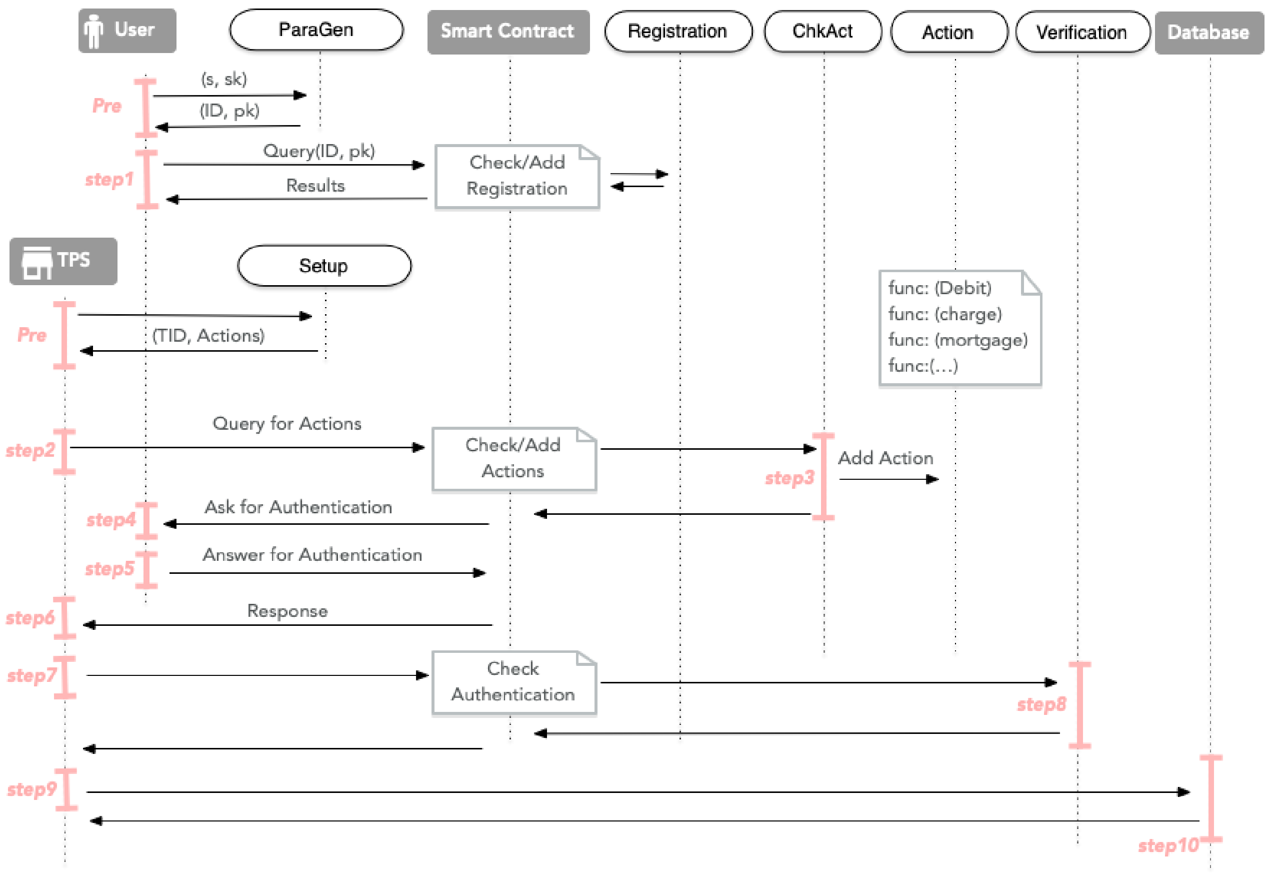


Figure 4.3: Concrete Workflow of PPM

- **Prepare:**
  - *Pre-User:* The preparation of user is to generate keys pair (sk, pk), identities id , and keep it at local.
  - *Pre-TPS:* The preparation of TPS is to generate an unique identity, and the required actions.
- **Connected with smart contract:**
  - *Step1:* The user sends a registered query to the smart contract. The contract is run inside blockchain oracle as a state transition. When received the query, it will first check whether the user existed in the list. If not, add it into the list. The function is denoted as `Registration`.
  - *Step2:* The TPS sends a query to the smart contract, which contains the TPS unique identity, and targeted actions for further authentication. When received the query, the contract will first check whether the actions are inside the action list, if not, add it into the list. The function is denoted as `ChkAct`.
  - *Step3:* The the function containing specific activity logic is denoted as `Action`. The function of action includes financial-related activities such as debit, mortgage, charge, payment, and so on.
  - *Step4:* The smart contract transfer the query to the user for the authentication on specific action. Each query can only represent one action.
  - *Step5:* The user reply to the query with Yes/No. The decision depends on the users personal wiliness. The authentication can be set as with three state: one-time authentication, certain time-period authentication, and forever authentication. The authentication reply is sent back to the smart contract, and being recorded into the list. *Step6:* The TPS get the response from the smart contract. The authentication of actions decides whether the TPS can proceed.
  - *Step7:* If the response is yes, the function is going to invoke and call the data. This step needs to firstly send a query to the smart contract for verification. When received the query, the contract will check whether the authentication inside the list. If yes, call the data, if not, deny the query. The function is denoted as `Verification`.
- **Invoke data:**
  - *Step9:* After the verification from the access control list on the smart contract, the TPS can call/invoke the data from an independently offline database. The offline database ensures the security of data and decreases the burden of a block. Also, the offline database can be flexibly mounted and layered.

– *Step10:* Received the query, and the database sends the sensitive data to the TPS. Since then, the whole process of the data sharing from user to TPS is achieved.

Table 4.1: Core Interfaces of PPM

FUNCTION	PARAMETER	RETURN	DESCRIPTION
_addAccount	account_address	1: address, 0: 0x0	Create a new account for users and institutions
_setProvenanceData	object{request details} user_address institution_address	Promise<address> 1: asset address, 0: ”0x0”	Record the request on the blockchain, including time and some detailed data
_verifyTransaction	object{request details} user_address	Promise<object>	Check and validate if the request is legal
_approveData	institution_address approve time, user_address	Promise<object>	Send user’s bank information to service providers
_lockAccount	account_address	Promise<boolean> 1: True, 0: False	Lock the user/institution account
_unlockAccount	account_address	Promise<boolean> 1: True, 0: False	Unlock the user/institution account

4.3 Analysis of PPM

Due to the particularity of financial data, we choose proof of authority(PoA) as the consensus algorithm. PoA is the improved algorithm on proof of stake(PoS) to some extent. In this section, we analyze the proposed model in business feasibility and attacks defending.

4.3.1 Business Feasibility

Our provenance-provided model aims at providing a data-sharing platform for traditional banks. Financial data is sensitive and of great value, which enables banks to choose not to open their service data. For the bank industry, keeping financial data private is a kind of temporary protection. The closed database makes banks lose the opportunity to take advantage of sharing data and advanced technology. Nowadays, customers have more critical demands on banking services and more technology giants participate in finance. The closed banking system will lose the chance to build financial ecosphere.

Our model is an open platform that provides a secure protocol for the bank industry to share partial service data. It provides small innovative financial firms opportunities to compete fairly with traditional banks. The benefits for customers are 1) they can manage all bank accounts in a simple interface. 2) they can choose bank products according to personal requirements. 3) they can manage their property and deposit in an efficient way.

4.3.2 Security Analysis

Public Key Management: Key pairs are generated during the registration process based on the user's bank account and private string. Users are required to set a secret string when generating their private keys. Moreover, the blockchain ledger stores the associated public key. This methodology ensures the anonymity of user identities. If their bank account information is stolen and misused, the private key can prevent financial harm.

Additionally, if users wish to update their public key, they can do so by submitting a request. The consensus nodes will update the key via transaction broadcasts.

Audit Provenance Data: When a user submits a data-sharing request, a provenance form is generated. It contains the timestamp, user and institution addresses. Then, following the verifier's confirmation of the signature, the verifier's identity will be added to the provenance form. Lastly, the data type, whose possible values are 'full' or 'part,' will be added to the form following the distribution of duplicates to service providers.

The blockchain stores provenance data, which records all manipulations of users' financial data. If consumers have a problem with their data access, they can examine provenance records to determine if their sensitive data has been accessed maliciously. Therefore, provenance information can assist users in properly managing their sensitive data.

Access Control: Access control is a significant feature in a distributed model. We build a series of APIs to control external data accesses. As presented in Table 4.1, we design six critical functions in a smart contract and illustrate in the following paragraph.

- *addAccount:* When users or service providers successfully registered in our system, this function will be triggered, and their essential information will be recorded on the blockchain
- *setProvenanceData:* When the data request is granted, and institutions receive the duplicate, this process will generate a provenance form. The function will be called.
- *verifyTransaction:* This function will be invoked when a new request needs to check the signature.
- *lockAccount:* This function can lock users and institutions account with their address.
- *unlockAccount:* This function can unlock users and institutions account with their address.

Through designing these interfaces, all data accesses will be managed under the regulation. If unauthorized service providers try to obtain personal data, the model will reject the request. In this way, we can secure users' financial data.

4.4 Discussion

Table 4.2: Comparison between Related Models

Features	Gan[46]	Zys[56]	BBDS[58]	Xia[14]	DataProv[51]	ProvChain[30]	**PPM**
Blockchain-promised	Y	Y	Y	Y	Y	Y	Y
Identity Management	Y	Y	Y	Y	N	N	Y
Programmable Laws	Y	Y	Y	Y	N	N	Y
Access Control List(ACL)	Y	Y	N	Y	Y	N	Y
User Privacy	N	N	Y	N	Y	N	Y
Scalable Storage	N	N	N	N	N	N	Y
Data Accountability	Y	Y	Y	Y	Y	Y	Y

Table 4.2 presents the comparison between current related models. We set 7 metrics to evaluate our PPM model, and provide the comparison with 6 related works. *Blockchain-promised* means the basic prototype is based on the blockchain oracle which inherits the properties of irreversibility, transparency, verifibility and traceability. *Identity Management* means the identity can be bound with its corresponding rights. *Programmable Laws* means the predefined laws and rules can be adaptively written into logic and then developed and deployed on smart contract. *Access Control List* represents the permissions of users and third party services. *User Privacy* is the customized access control functions, which can be implemented as APIs. *Scalable Storage* represents the permanent storage. Usually the blockchain stores its data on chain, which is limited by the block size. Scalable storage means the storage can flexibly add and drop according to the requirements. *Data Accountability* means the data history can be traced, so that the malicious behaviors can be caught by the public.

According to the table 4.2, we find that the PPM possess the features where other projects may not. PPM achieves user privacy and ACL by customizing the authentication lists. The internal functions also be hidden into APIs. The scalable storage is realized by the independent offline database for large scale data. The independent offline storage can also be flexibly mounted. The identities is hold by user itself, and the key pair (sk, pk) is generated by themselves. The programmable laws and data accountability inherent from the fundamental properties of blockchain.

4.5 Conclusion

We propose a blockchain-based provenance-provided data sharing model (PPM) for open banking. Our PPM model is designed with transparent authentication, privacy-protected control, and auditable provenance to meet the open banking area's requirements precisely. Our model employs programmable smart contracts as the intermediary witness between users and third-party services in order to ensure the trustworthiness and reliability of communication. The predefined laws and rules are encoded as contract logics, which are automatically executed on the blockchain. On this basis, we decouple the blockchain into hierarchical layers and present our modifications separately at the data layer (data content, transaction structure), the smart contract layer (ACL, logic), and the application layer (customized APIs). Detailed model module designs and workflows are also provided. The analyses and discussion demonstrate that our model is a secure and realisable approach to open banking.

Chapter 5

BHDA - A Blockchain-Based Hierarchical Data Access Model for Financial Services

Financial data sharing services face both opportunities and challenges posed by blockchain technology. Data access control and privacy control are essential features of a distributed system with multiple parties. This article proposes a hierarchical data access model for financial services that includes consent management and dynamic credit management. We implement granular data access control by rating authorised data recipients (ADRs). Credits for ADRs will be dynamically updated by accessing the corresponding blockchain service logs. The credits evaluation algorithm is responsible for calculating the credits of ADRs based on their completion rate, business ethics rate, and positive feedback rate. In addition, by utilising smart contracts, the efficacy of consent management can be enhanced and privacy policies can be managed flexibly. Finally, we deploy smart contracts on the Ethereum Rinkeby testnet in order to assess the viability of the model. In addition, the theoretical analysis and experimental findings demonstrate that the prototype is secure and effective.

We propose a blockchain-based hierarchical data access model to realize data access and consent management. Because of the openness and tamper-proofing of the blockchain, records on the blockchain are transparent, and users can audit history service records. By preprogramming control logic in smart contracts [22] beforehand, consent management and multi-level financial data access can be performed automatically. In summary, our model achieves the following contributions:

- **Transparent registration & service records:** Based on consortium blockchain, the system's proposed model needs to register on the blockchain to record their certificate data. Before accepting financial services, clients can view the certificate and history services' information about ADRs.

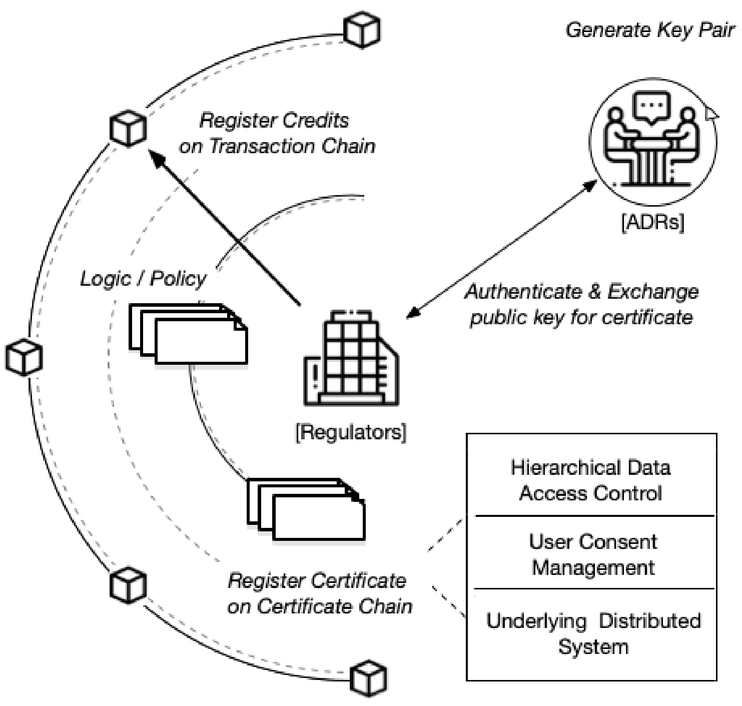

Generate Key Pair
Register Credits on Transaction Chain
[ADRs]
Logic / Policy
Authenticate & Exchange public key for certificate
[Regulators]
Hierarchical Data Access Control
User Consent Management
Underlying Distributed System
Register Certificate on Certificate Chain

Figure 5.1: Accredited Data Recipients Registration Model

- **Hierarchical data access control:** The hierarchical data access control is based on transactions completed by ADRs. As more and more blocks are created along the transaction chain, the ADR's score is dynamically updated. At the same time, different scores can access different levels of personal financial data.

- **Automated consent process:** In order to improve the approval efficiency between different parties, we write logic into smart contracts for automated management.

5.1 Underlying Construction

The underlying structure of the system ensures the accurate and efficient operations of the system. Our platform connects various financial data islands, providing a robust and reliable platform for open financial services. The BHDA system is based on the consortium chain [90] and uses proof of authority (PoA) consensus algorithm [91] to provide blockchain services. Regulators and ADRs maintain the PoA committee nodes corporately.Due to the sensitivity of financial data, our model cannot apply some openness consensus algorithms (i.e. Proof of Work) to the proposed framework. PoA ensures that all chain data is maintained by several authorized nodes and managed jointly. Smart contracts are responsible for executing policies and rules according to pre-programmed logic. The detailed descriptions for consortium chain, PoA and smart contract are as follows.

Consortium chain: Consortium blockchain mainly refers to a distributed ledger system where the

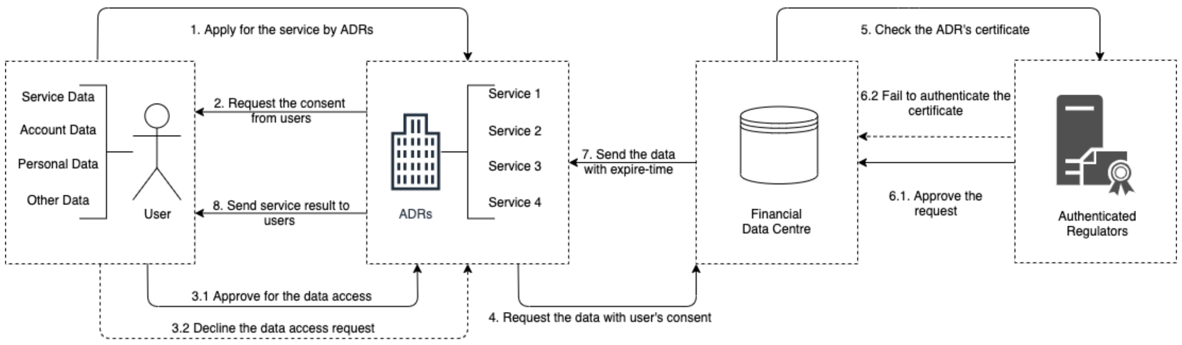

1. Apply for the service by ADRs
Service Data
Account Data
Personal Data
Other Data
User
2. Request the consent from users
8. Send service result to users
ADRs
Service 1
Service 2
Service 3
Service 4
7. Send the data with expire-time
Financial Data Centre
5. Check the ADR's certificate
6.2 Fail to authenticate the certificate
6.1. Approve the request
Authenticated Regulators
3.1 Approve for the data access
3.2 Decline the data access request
4. Request the data with user's consent

Figure 5.2: Consent Process

blocks' decision-makers are limited in a relatively small committee. The committee members corporately maintain the chain and share the incentives. The Consortium blockchain is inherently permissioned with higher efficiency. Many companies and their alliance corporate together to establish their consortium blockchain, including the Hyperledger [7] and Corda [92]. We build up the consortium chain based on the Ethereum platform with the PoA consensus mechanism. In this model, we provide 10-100 testing nodes to form the committee. The nodes in the committee are elected to be the leader, in turn, to package the transactions and broadcast the blocks.

PoA: PoA is a kind of consensus algorithm used for blockchain systems. It can be regarded as a modified version of Proof-of-Stake (PoS) [93] by using a validator's identity performs as the stake. By utilizing the PoA mechanism, the system enjoys the advantage of (1) *lower energy consumption*: The resource consumption of PoA is significantly lower than that of PoW. (2) *Higher Efficiency:* The transaction confirmation time of PoA is considerably faster than PoW or PoS. (3) *Better Scalability:* It is quite easy to build and maintain a decentralized application (DApp) using a PoA based network.

To ensure consistency, our model employs the PoA to establish the fundamental consensus mechanism. All transactions and blocks in PoA-based networks are validated and approved by authenticated nodes. The granted nodes are designated as validators, and they consist of ADRs with a stellar reputation. To be eligible for staking identity, a real identity and a reasonable deposit were required. Their credibility would be the guarantee of operating the blockchain system with integrity.

Smart Contract: Smart contracts are a set of blockchain-based computer codes designed to automatically facilitate, verify, or enforce predefined rules and principles. A contract's negotiation or agreement is dependent on the underlying consensus mechanism in order to maintain the consistency of orders and states. Smart contracts based on the blockchain are intrinsically traceable and irreversible without compromising their viability. Our model implements modular functions such as service registration, policy management, and data maintenance using smart contracts. Each modular design is presented in the following section.

5.2 Blockchain-based Consent Management Model

The model for consent management is an essential component of the BHDA model. It manages authorisation, registration, and consent for financial services. There are two phases to the flow of consent management. One is that when a user accepts an ADR's service, a consent identifier is generated that grants the ADR access to the user's private financial data. The alternative is for the ADR to submit the user's consent identifier, the user's data level, and the ADR certificate to the regulatory authority in order to obtain permission to obtain data from financial institutions.

In the BHDA system, data integrity and availability [94] are very important indicators, including data protection, accessibility, and authenticity. So we record all the consent information on the blockchain to ensure the information is open and transparent. In this section, we present the business model, registration process, and automated consent workflow.

5.2.1 Business Model

Due to the sensitivity of the data, the majority of Australian financial institutions only offer read access (data sharing) for ADRs at this time. The goal of the BHDA model is to connect disparate data islands in order to encourage ADRs to offer innovative and collaborative services. These services consist of customised financial data management and ADR credit evaluation. In the system, there are a total of four participating parties:

- **ADRs:** Authorized accredited data recipients acts as service providers in our model. After obtaining the certificate information by registering on the chain, ADRs can provide appropriate financial products with the consent of users.
- **Financial Institution:** A financial institution is responsible for managing all of a user's data.When an ADR requests data with the consent of the user and regulatory authority, the financial institution will open read access to the corresponding data according to the type of service and the ADR's qualifications.
- **Consumers:** Consumers are the entity that owns financial data and can initiate financial services provided by ADRs.
- **Regulators:** Regulators are responsible for generating and supervising the ADR's certificates. In financial service' processes, they are responsible for verifying the validity of company certificates.

These entities are essential components of our system. The system can automate the consent procedure by incorporating business logic into smart contracts. We store transaction records on the blockchain and user financial data in conventional distributed databases due to the sensitive nature of the data and the blockchain storage. In addition, transaction records on the blockchain create a transparent environment for others to audit data and monitor suspicious behaviour.

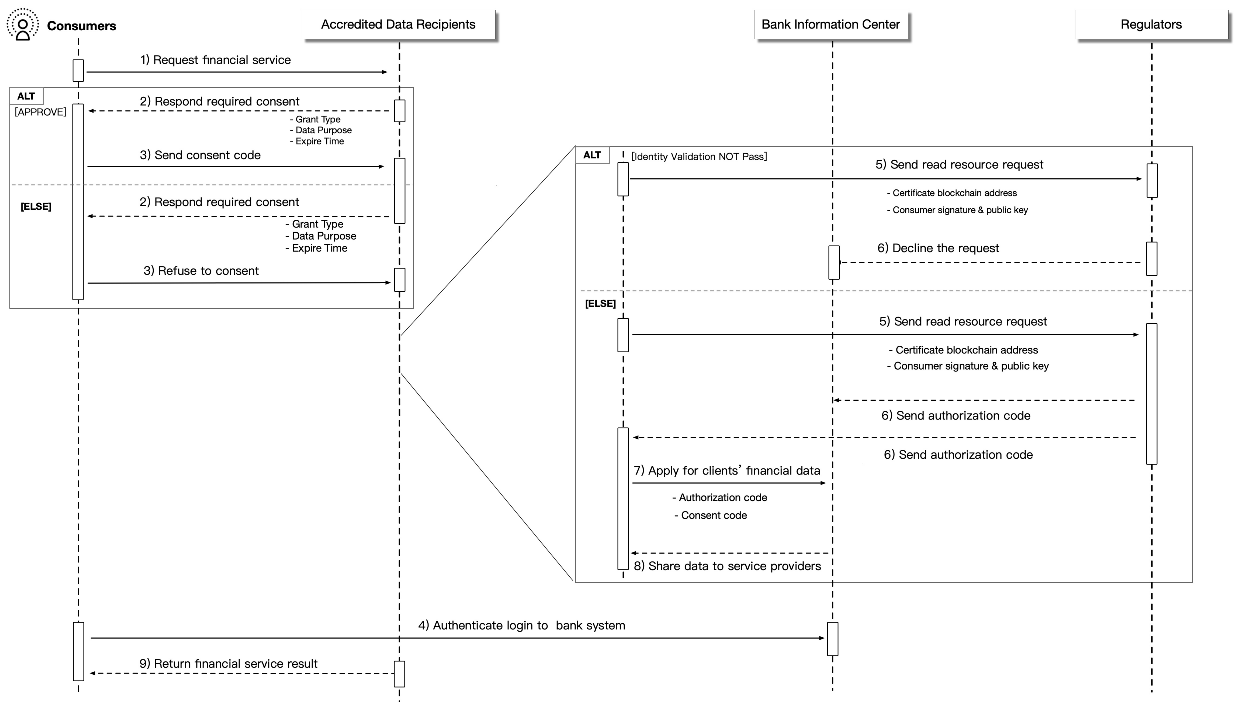

Consumers
Accredited Data Recipients
Bank Information Center
Regulators
1) Request financial service
ALT
[APPROVE]
2) Respond required consent
- Grant Type
- Data Purpose
- Expire Time
3) Send consent code
[ELSE]
2) Respond required consent
- Grant Type
- Data Purpose
- Expire Time
3) Refuse to consent
ALT
[Identity Validation NOT Pass]
5) Send read resource request
- Certificate blockchain address
- Consumer signature & public key
6) Decline the request
[ELSE]
5) Send read resource request
- Certificate blockchain address
- Consumer signature & public key
6) Send authorization code
6) Send authorization code
7) Apply for clients' financial data
- Authorization code
- Consent code
8) Share data to service providers
4) Authenticate login to bank system
9) Return financial service result

Figure 5.3: Automated Consent Workflow

5.2.2 ADRs Registration

In the conventional mechanism for managing certificates, all institutions' certificates are managed by authorised institutions. Such centralised management can easily result in a single point of failure and other potential dangers. Moreover, centralised certificate management obscures certificate information and may be perceived as tampering. To avoid the aforementioned issues, we record the ADR certificates in the blockchain. Due to the immutability of blockchain data, data integrity and traceability are assured. In this section, we will describe the entire registration service's workflow in detail.

In an open financial ecosystem, many third-party institutions compete by offering a variety of financial products for users to select. They are both competitors and collaborators, sharing customers' financial information with their customers' consent. Because of the importance and sensitivity of users' financial data, regulators are responsible for signing certificates to service providers. As shown in Figure 5.1, ADR registration is divided into two steps:

- **Register on certificate chain:** ADRs need to register a certificate on the certificate chain before providing financial services. Because of the sensitivity of financial data, regulators manually verify ADRs' qualifications. If approved, the ADRs' certificate information is stored on the block, including the beginning date, expire date, data access level, and information hash code. At the same time, regulators will send the block address to ADRs as proof of providing financial services in the future.
- **Register on transaction chain:** The transaction chain records detailed information about

financial services, including the service type and user feedback score. After completing services, details such as user identity number, completion time, and feedback score are recorded on the chain. Based on this information, the ADRs' scores will be dynamically adjusted according to the completion rate, business ethics rate, and feedback positive rate. The specific scoring mechanism will be explained in detail later.

Due to traditional certificate authorities controlling certificates privately, certificate transparency is a prerequisite in our model. By registering certificates on an append-only public chain, consumers can validate the ADR's identity before choosing their products and services. By appending ADRs' certificate information and financial services transaction information on the certificate chain and transaction chain respectively, we provide a secure, transparent, and auditable environment for users and ADRs. Through the smart contract, when the certificate information of ADRs is expired or needs to be updated, we can put the updated information on the blockchain to ensure the traceability of all certificate information.

5.2.3 Consent Control Flow

Consent management is indispensable in the BHDA framework. It allows users' financial data to be shared with qualified ADRs with authorization and consent. We divided the financial data of users into four levels: fundamental data, moderate data, analytics data, and restricted data. We would discuss these data categories in section 5.3.2. Reasonable consent management can prevent data from being illegally accessed and also ensure privacy and security. The following paragraph illustrates the consent process.

Consent Process: In the system, consent's flow has two phases: the users-ADRs and the ADRs-regulators consent phase. As shown in figure 5.2, in the users-ADRs phase, users can obtain financial services from qualified ADRs according to their own needs. ADRs need to request consent from users to access their financial data before providing services to them. If users reckon that ADRs have excessive access to their data, they can revoke their authorization to ADRs, and the financial services lifecycle will terminate immediately. In the ADRs-regulators consent phase, ADRs will initiate a request to the regulators to obtain a consent to access the users' financial data. Through reviewing users' consents and ADRs' certificate information, regulators will assign consent to ADRs. When ADRs obtain both user's consent and regulatory consent, they can obtain users' data from the financial institutions successfully. The entire process requires the participation of all four entities, and if either party revokes the consent, the entire service will be terminated immediately.

To get a more intuitive understanding of the consent management process, we have drawn a sequence diagram to support our consent management process. There are nine steps totally, as shown in Fig.6.3.

- *Step1:* Users send requests to qualified ADRs for services such as loan advice, mortgage comparison, and financial planning.

- *Step2:* After receiving the user's request, ADRs will send the required financial data categories, authorization time, and other information to the user to solicit the user's consent.

- *Step3:* If users reckon the data required by services is inappropriate, services will be terminated immediately, and the service information will not be recorded on the blockchain.

- *Step4:* Users agree to license the data to ADRs for a short period to provide financial services.

- *Step5:* ADRs sends the user's consent information, along with certificate information, to regulators for auditing.

- *Step6:* Regulators will determine whether to give consent based on the validity of the user's consent, the validity of ADRs certificate, and the data access authority of ADRs. If approved, the service information is added to the blockchain, and an authorization consent code is issued to the ADRs.

- *Step7:* ADRs send user's and regulator's consent code to financial institutions for the validation.

- *Step8:* When the validation is passed, the consumer data is transferred to the ADR accordingly.

- *Step9:* ADRs returns the result of the corresponding financial service to the user.

5.2.4 Automated Authentication

Smart contracts conduct the automated authentication service. According to business features in financial services, we pre-program the relevant authentication rules into the smart contracts. The authentication service includes two sections, which are *grant user consent* and *validate service provider*.

Grant User Consent: In the BHDA model, managing user consent is essential in financial service, a unique characteristic. When clients accept the service, they need to generate a consent identifier that contains an expiration date, purpose, and type of service information. Upon receipt of the consent identifier, the ADR verifies its validity and stores it as a transaction credential on the blockchain. Besides, if the user needs to audit previous service details, the consent identifiers on the blockchain can be provided as a historical record, which is transparent and reliable.

Validate ADRs: Each ADR needs to be registered and managed by a regulator. The request is sent to the regulators for authentication with the unique identifications of the blockchain address and the consumer's public key. Approval of request by the regulator permits ADRs to receive consumer data from the financial information center. The financial information center manages and stores the raw data of consumers. Note that this functional component connects the authorized ADRs, the financial information center, and the regulator.

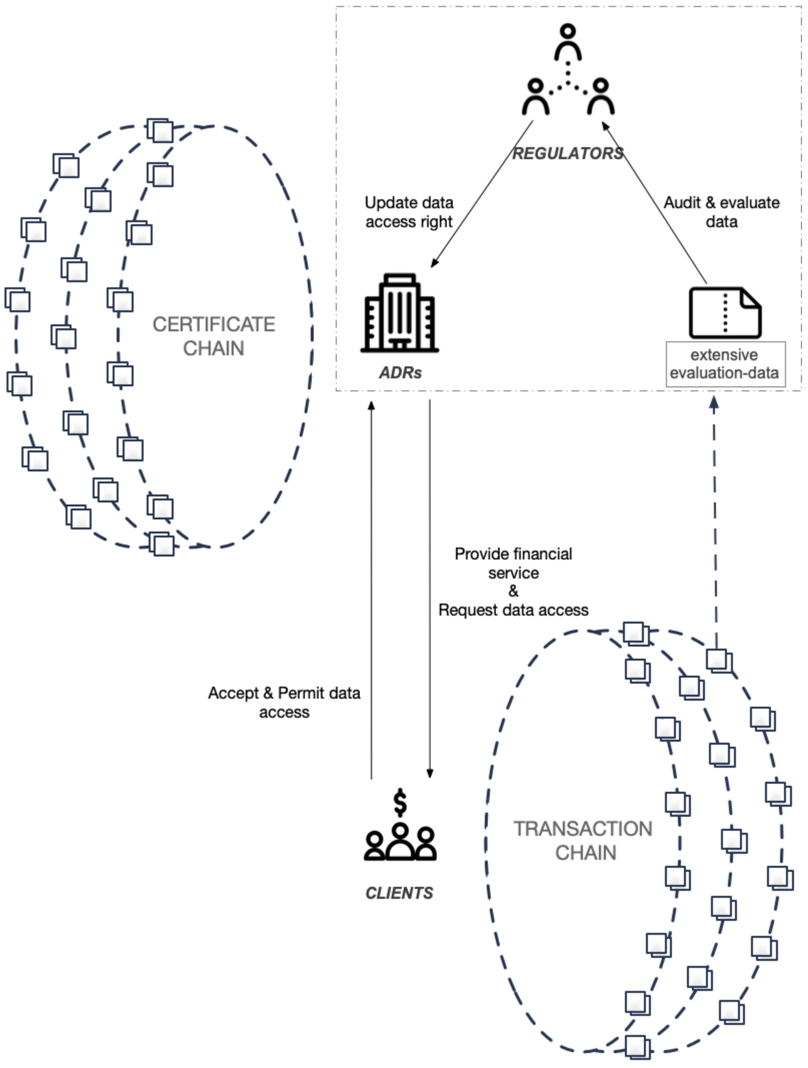


Figure 5.4: Overview of ADRs' Credits Evaluation

5.3 Dynamic Hierarchical Data Access Control System

The BHDA is designed to achieve hierarchical access to clients' financial data. In our proposed prototype, we develop two chains to store ADRs' certificate information and transaction information, respectively. The certificate chain is responsible for recording the ADR's credit information and certificate details. The transaction chain records each service order from ADRs, including order completion data, user feedback, and service type. When providing services, ADRs conduct data analysis by accessing users' private financial data. Therefore, it is necessary to reasonably control the access of ADRs to users' financial data.

5.3.1 Business Model:

In our model, We update ADRs' credits through multidimensional analysis of transaction data as shown in Figure 5.4. According to the score results, the system realizes the hierarchical access control

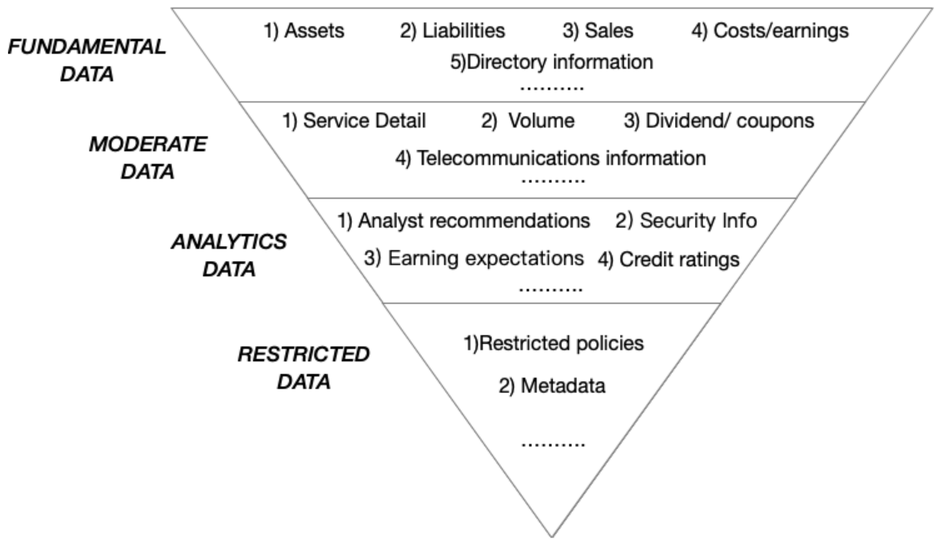

FUNDAMENTAL DATA
1) Assets 2) Liabilities 3) Sales 4) Costs/earnings
5)Directory information
....
MODERATE DATA
1) Service Detail 2) Volume 3) Dividend/ coupons
4) Telecommunications information
....
ANALYTICS DATA
1) Analyst recommendations 2) Security Info
3) Earning expectations 4) Credit ratings
....
RESTRICTED DATA
1)Restricted policies
2) Metadata
....

Figure 5.5: Financial Data Pyramid

of ADRs to users' financial data. A multi-level data access system can reduce the risk that users' financial data will be over accessed. By hierarchical classification of ADRs, ADRs can promote their credits to access more users' financial data. We score the overall social behaviors of the ADR by evaluating the positive feedback ratings, completion ratings, and business ethics ratings.

- **Completion Rate:** The transaction completion rate is calculated by calculating the percentage of successful service among all transactions. At the same time, the completion rate is also a manifestation of the ADR's performance ability.

- **Business Ethics Rate:** Business ethics Points are used to measure whether an ADR has a violation of a privacy standard. Violations are obtained through user feedback after each transaction is completed. Although blockchain can organize almost all unauthorized access to data, malicious actions can still occur.

- **Feedback Positive Rate:** Feedback positive rate is an index that side standard ADRs' services. It can reflect the quality and efficiency of financial services.

By reference to the FICO system, we drew up an algorithm to rate an ADR's credit. The transaction chain $Tc = \{Tc_1,\ Tc_2,\ Tc_3\ \ldots\ Tc_n\}$ contains each service record from ADRs, where $Tc_i = \{\lambda_{i,1},\ \lambda_{i,2},\ \lambda_{i,3}\ \ldots\ \lambda_{i,n_i}\}$, n_j is the j^{th} attribute in the transaction chain block. These attributes contains order completion data, user feedback and service type. The completion rate is denoted as $\overline{CR}$, where n is the total number of transactions done by the ADR. It can be retrieved from the transaction chain.

$$\overline{CR} = \frac{\sum\limits_{\sigma=1}^{n} CR_\sigma}{n}$$

The feedback positive rate is denoted as *FPR*. The *FPR* is based on a comprehensive assessment of user ratings. Mark point represents the Mk. Our model divides the rating into five different levels, represented as Mk, Mk2, Mk3, Mk4, and Mk5. Furthermore, n, n2, n3, n4, and n5 represents the corresponding number of clients.

$$FPR \ = \ \frac{n5 * Mk5 \ + \ n4 * Mk4}{n + n2 + n3 + n4 + n5} + \frac{n3 * Mk3 \ + \ n2 * Mk2 \ + \ n * Mk}{n + n2 + n3 + n4 + n5}$$

The business ethics rate is the overall rate of service completion without privacy disclosure. After obtaining these rates, we put different weights on these ratios: 45% on completion rate, 30% on feedback positive rate, and 25% on business ethics rate. After obtaining the final composite score, we would update ADRs' credits.

5.3.2 Financial Data Pyramid:

A data classification that can be understood on some level is required by the BHDA system. In accordance with the findings of the Oreilly report [95] we would like to organize the users' financial data into the following categories: fundamental data, moderate data, analytics data, and restricted data, as illustrated in the figure 5.5.

- **Fundamental Data:** The term "fundamental data" refers to the public financial information that is available to anyone. Users will face few challenges as a result of the disclosure of these data. ADRs with the appropriate authorisation have unrestricted access to the fundamental data.
- **Moderate Data:** The history of financial services, information regarding loans, and tax information are all examples of moderate data, all of which can refer to the identities of the users. Users might be exposed to some risk even if only a small amount of data is disclosed.
- **Analytics Data:** Analytics data can be analyst recommendations and earnings expectations, which is highly sensitive and confidential. The leakage of analytics data could bring significant risks to users.
- **Restricted Data:** Restricted data can be users' meat data and details of total investments, which need restricted access control. The leakage of restricted data could cause severe risks to users.

As a consequence of this, we will carry out user data access control on ADRs based on these four categories of financial data. In the beginning, ADRs only have restricted access to the fundamental data of their users, such as their bank statements, so they can only offer them limited services. The number of times the service is performed will directly correlate to an increase in the scores awarded

Algorithm 3: Multi-level Data Access

Input: Request of financial data

Output: consentCode

```
while AddList(pk,requestString) do
    consentCode=Hash(pk,requestString) if ckApprovalList(consentCode) then
        return "Service Already Approved";
    else
        if ckConditions(pk,requestString,Timestamp) then
            addList(consentCode,Timestamp);
        else
            return “Request Refused"
        end
    end
end
while ckConditions(requestString,Timestamp,adrScore) do
    if ckGrantType(requestString) then
        if validTime(Timestamp) then
            return “Time Valid”;
        else
            return “Time Expired Failed";
        end
        if ckDataLevel(adrScore) then
            return “Legal Permission”;
        else
            return “Illegal Permission";
        end
    else
        return “Service Failed";
    end
end
```

to ADRs, which will be based on the opinions and suggestions of service recipients. When the score reaches a certain level, the merchant can access more user data to provide better service. According to the Algorithm 3, we can see that when ADRs request a consent code from regulators to read data, the request will be rejected if the ADRs' credit is invalid.

As a result of the immutability of the blockchain, there will be no falsification of transaction data, and all transaction data will be able to objectively reflect the behaviour of ADRs. Consequently, there will be no need for a central ledger. In the same time, the control logic of ADRs' rating and user financial data access control is implemented by smart contracts, ensuring the whole process's integrity and safety in the process.

5.4 Evaluation

To evaluate the BHDA model, we conducted two performance testing on smart contract's function response time. Moreover, we analyzed the security and privacy feature theoretically.

5.4.1 Efficiency Performance

Testing Environment: We have two test environments, one for private chain and another for the public chain. For the private chain, we set up two local servers (8x 3.7 GHz CPUs, 16 GiB memory, 256 GiB hard disc) to deploy the private chain on this hardware and use Geth as the Ethereum client. The response time is acceptable because it supports up to 1200 transactions per second (TPS). Moreover, we set the block interval as 5 seconds when generating the *genusis.json* to boost the performance. We have tested that each transaction carries data of 50 bytes, a typical value for general transactions.

For the public chain, We adopt the Truffle [96] as the Ethereum framework, which is a scriptable, extensible deployment & migrations framework. Besides, we set up a test bench on the Ethereum Rinkeby [97] testnet to test the system performance under the real-world blockchain system. Rinkeby testnet is different from the Ethereum mainnet. It adopts PoA as the consensus algorithm. The system accesses the Ethereum testnetwork using APIs provided by Infura, in which way we do not need to set up a local Ethereum node and sync all the chain data before submitting the transactions. The smart contract is deployed and maintained by the Truffle toolkit.

Functions response time: For the public network test, we select some smart contract's functions to test the function response time, which are *addCertificate()*, *getCertificateInfo()*, *checkScore()*, and *updateCertificateInfo()*. Function *addCertificate* aims to create a certificate block for ADR; *update-CertificateInfo* is used to update certificate status on the chain; *getCertificateInfo* returns the ADR's info on the block; *checkScore* aims at checking if an ADR has the right to access users' data. As shown in Table 1, we test these functions four times, and the average response time of these core functions is considered acceptable for a class of specific applications. Usually, the typical TPS of Ethereum public

Table 5.1: Function Response Time Test

Time/ms	**Set 1**	**Time 2**	**Time 3**	**Time 4**
addCertificate()	106.192	93.626	98.658	86.273
getCertificateInfo()	54.355	41.243	32.558	38.884
checkScore()	37.511	32.186	23.753	30.883
updateCertificateInfo()	75.401	78.059	75.995	76.931

network is around 10 20. According to the result of the response time, we believe it is acceptable for services that BHDA provides.

5.4.2 Security and privacy analysis

Content of Transaction: Information related to financial services will be appended on the certificate chain and transaction chain to ensure the data's reliability and transparency. To enhance the security and privacy of the system, the BHDA model performs dynamic scoring according to the service behaviors of ADRs. Depending on the score of the ADRs, the ADRs can access different levels of user financial data. Once an ADR successfully registers on the certificate chain, the certificate data and expired time are recorded on the ledger permanently. When clients request a service from the ADR, the process details would be recorded on the transaction chain attentively. Our model's transaction content includes the clients' consent, ADRs' certificate, and the transaction record. Due to the immutability of the blockchain, the certificate and transaction data cannot be modified maliciously. Besides, clients' sensitive data is stored off-chain, which cannot be shared without corresponding consents or valid certificate data.

Data Storage: The public can view Transaction-related data in order to achieve data transparency. Due to the limited storage space and transaction speed on blockchain, we store the users' financial data on the local servers and store the hash summary of data on blocks. Once the user's local data is modified, the hash summary of the corresponding block changes. In the proposed BHDA model, there are two categories of data stored on the transaction chain and certificate chain. Once an ADR successfully registers on the certificate chain, the certificate data and expired time are recorded on the ledger permanently. When clients request a service from the ADR, the process details would be recorded on the transaction chain attentively. Our model's transaction content includes the clients' consent, ADRs' certificate, and the transaction record. Due to the immutability of the blockchain, the certificate and transaction data cannot be modified maliciously. Besides, clients' sensitive data is stored off-chain, which cannot be shared without corresponding consents or valid certificate data.

Anonymous Users' Identities: A complete and secure system is quite significant and necessary to protect users' personal information. In many real-life applications, the user's true identity can be inferred by tracking and analyzing transaction records, even if the user's identity has changed to a serialized number. In our system, by combining with the blockchain, all users' financial service records are anonymously recorded on the blockchain. Besides, when the transaction history of ADRs needs to be audited, the user's identification number is recorded anonymously on the block. Because of blockchain's anonymity, users' personally identifiable information will not be disclosed. As a result, users' identities can be adequately protected in our model.

5.5 Conclusion

In this chapter, we propose a hierarchical data access model based on blockchain to realize the data access and consent management in open financial service. We employ smart contracts to customize the policy and conditions to flexibly meet various financial requirements and the PoA-based underlying distributed system to manage the consortium chain efficiently. The model provides capabilities on automation, transparency, and accountability. Our model is expected to benefit the financial infrastructure in practice.

Chapter 6

AC2M: An Automated Consent Management Model for Blockchain Financial Services Platform

There is a trend in Open APIs and data sharing to provide better and innovative financial services. Data regulations aim to improve consumers' privacy and control over their data. However, it still confronts the challenges of properly managing and protecting users' privacy in such an open data sharing paradigm. This paper proposes an automated consent management model based on blockchain technology for various coming financial services. We employ the Proof of Authority consensus mechanism to balance the business flexibility and efficiency with client privacy control. Our model enables automated, transparent, and accountable management for consumer consents in financial services. We implement a proof of concept prototype to demonstrate the feasibility and applicability of our model. Performance testing is also conducted to demonstrate the feasibility and efficiency from the performance viewpoint.

In this chapter, we propose a blockchain-based automated consent management model (AC2M) for the open financial services, which provides functions of the ADRs' secure registration, financial data sharing, and the update of automated consents. Our model employs the blockchain as the underlying foundation to support the system establishment. We utilize the Proof of Authority (PoA) [98] mechanism as the consensus algorithm to improve the efficiency and smart contract as a state transition machine to execute and enforce the policies automatically. Since the request records are transparent on the append-only ledger, all users, regulators, and other authorities can audit and review them. Moreover, unauthorized people cannot manipulate and even view extra security control records because of blockchain's tamper-proof characteristics. Our model applies broader financial services and other sectors involving consumer data sharing. In summary, our model achieves the following

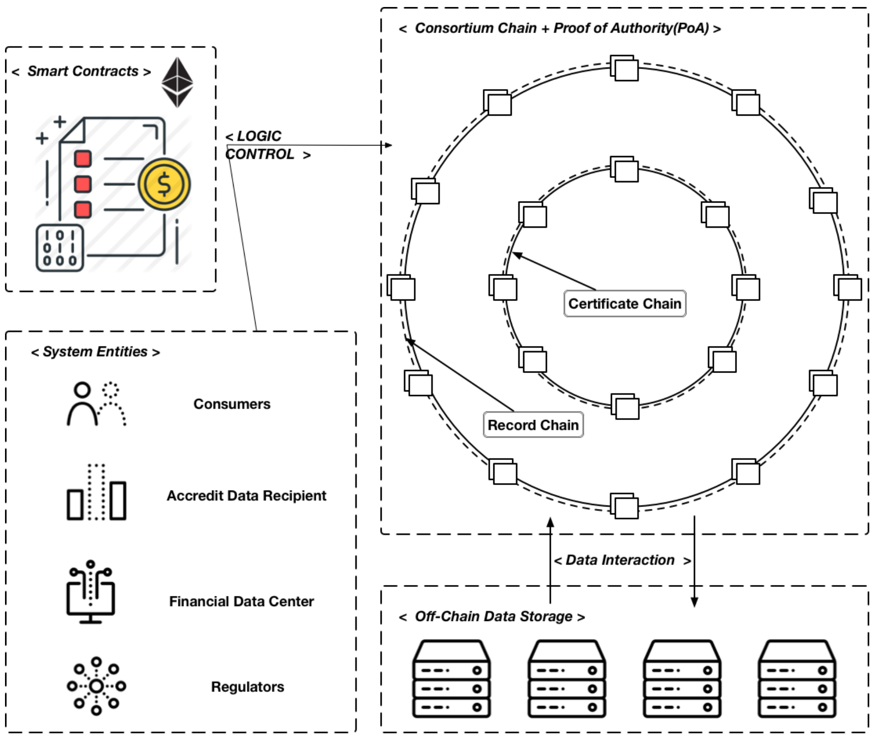


Figure 6.1: AC2M Structure

contributions:

- **Reliable user data sharing:** Our model proposed a novel consent management workflow, and the user, regulators, and ADRs jointly and automatically manage the data sharing. Data sharing between different parties need to verify certificates and request are controlled by smart contracts around different parties.

- **Transparent registration:** Due to the ADRs being responsible for providing various financial services, we record ADRs' registration data on the public ledger. Participants can examine and ensure the validity of companies' identities before granting them to provide services.

- **Time-limited automated authentication:** Our model can automatically check ADRs' certificates and users' consent by integrating specific rules into smart contracts. Once identities are valid, ADRs will receive data consent with a set expiration time.

6.1 System Overview

Our model intends to develop an automated user consent management system utilising the consortium chain and the PoA consensus algorithm. This section will provide a brief overview of the business model and system capabilities.

6.1.1 Business Model

The majority of open financial data-sharing services in Australia currently offer read access (data sharing) to ADRs. This data-sharing initiative seeks to foster more business collaborations and innovations, such as innovative credit assessment and personalised financial management from financial data centres. System dependability and data regulation are required due to the proliferation of FinTech. As shown in Figure 6.1, there are four parties in our AC2M model:

- **ADRs:** Authorized accredited data recipients, also known as service providers, offer users financial products. They obtain explicit consent from users in order to offer reasonable and suitable financial products by gaining access to user data from financial institutions. Due to the sensitivity of financial data, they must obtain permission from regulators prior to offering services to customers.

- **Financial Institution:** Financial institutions prioritise the management of users' financial and personal information. They are accountable for granting access to ADRs if users approve the request. If an authorised data request is received, they will grant data read permission with a validity period.

- **Clients:** Financial clients are entities that hold financial accounts and have financing needs. They may send a request to ADRs authorising them to conduct an analysis based on their financial data.

- **Regulators:** Regulators are institutions that oversee the sharing of financial data. They are accountable for authorising service provider certificates and managing regulatory processes. They will validate the identities of ADRs included in data requests.

These four entities are fundamental to our platform. Using well-defined, transparent smart contracts, consumer requests will be automatically processed by pre-defined smart contract rules. To establish a transparent context for certificate management, our model provides on-chain ADR registration records. Moreover, every consent request that is fully processed is also recorded in the chain. It can be audited by authorities to monitor illegal behaviour.

In the proposed model, on-chain and off-chain storage are combined. We choose to store data locally due to the enormous size of financial data and the limited storage space on the blockchain. Among these data are the account balance, bank statement, and transaction history of the user, in addition to analytical data. We intend to store these data off-chain, as their size exceeds the block's storage capacity and their sensitivity warrants it. As a result, we design two blockchains, certificate chain and record chain, to record transactions and ADRs activities, as will be demonstrated in the following section.

6.1.2 Threat Model

In conventional centralised or distributed data sharing models, there are a number of pain points, including multiple data authentication levels, privatisation of data management, and data transparency. In the data-sharing system, data requests must be approved by various entities, users, institutions, and financial hubs. This pattern of multi-layer data validation causes data sharing latency, which reduces service efficiency and degrades the user experience. Certain institutions maintain service data and historical records, and if the company does not improve security, it is likely that the data will be manipulated maliciously. Users are typically concerned about the transparency of data management in conventional models. When authorising others to provide financial services, they attempt to prevent the misuse of personal data. It will result in the disclosure of private information and financial loss.

As a result, a trustworthy and open system for sharing financial data is crucial. To ensure system integrity in the proposed AC2M model, we employ the consortium chain and proof of authority(PoA) consensus algorithm as the underlying structure. As an open and transparent online repository, the blockchain guarantees the data transparency of our system. Moreover, the use of smart contracts simplifies the data authorisation process. Smart contracts execute data requests automatically by writing business logics into contracts.

6.1.3 Transparent Registration

Certificate authorities are responsible for managing the certificates of institutions according to the conventional certificate generation schema, including issuing, updating, and revoking certificates. This centralised administration may result in a single point of failure and other security issues. Similarly, centralised financial institutions are responsible for the unreliable maintenance of certificate management in data sharing.

In the AC2M model, we design two blockchains: certificate chain and record chain, to manage ADRs' certificate and service transactions in a transparent manner. The certificate chain is accountable for documenting ADR qualifications. ADRs are agents who provide clients with financial services. Their qualifications as service providers are determined by the veracity and dependability of their identities. For example, services provided by an uncertified ADR may steal users' information without any privacy or security protections. The record chain contains the ADR behaviours of the model's financial services. Both the record chain and the certificate chain are maintained by regulatory authorities. Periodically, they would audit the chain data to confirm that the ADRs' certificates are valid and that their behaviours are benign.

- **Register on certificate chain:** ADRs need initially generate the critical pair and transmit it to regulators. Then, regulators verify the identity of ADRs. If the identity is valid, the regulator will sign the certificate and add certificate-related information to the certificate chain, including the signer's identity and the certificate's expiration date. After the block is generated on the blockchain, regulators will transmit the block's hash address to ADRs. In our model, prior to a user requesting financial services from an ADR, the ADR's data on the certificate chain would be validated and authorised.

- **Register on record chain:** The block on the record chain will be generated automatically upon completion of a financial service. ADR's initial credits, ADR's certificate address, ADR's identity number, request timestamp, request type, and expiration time would be pushed onto the blockchain. The record chain is responsible for collecting ADR's consent request behaviours. The ADR's credits represent their own dependability. Due to the fact that regulators maintain the record chain, once fraudulent activities were detected, regulators would reduce ADR's credits.

In our model, certificate and behaviour transparency are prerequisites. By registering certificates on an append-only public chain, users can verify the identities of ADRs before selecting their products or services. In addition, the smart contract would monitor certificate validity. When the certificate of an ADR has expired, the script will deactivate it until it is renewed. Consequently, operations such as certificate update, revocation, and registration are all recorded on the certificate chain with the aid of the smart contract, and the record chain would store the behaviours of ADRs in financial services for purposes of supervision.

6.1.4 Automated Consent Model

Consent management is a crucial component of the open financial service architecture. It includes the agreement between ADRs and financial data centres. With the help of smart contracts, the automated consent validation can reduce latency in financial services and modularise the consent flow. Included in consumer data are personally identifiable information, account balances, and transaction records. As a result, we consider data management seriously and with great intensity. In addition, we define the data structure of consumer authorisation on the blockchain. In the following paragraphs, we will discuss these two data structures and consent workflows.

Consent Process: In the proposed prototype, we have two stages of the consent process, as in Figure.6.2. One is the consent between users and ADRs. Consumers hold various data, including financial service data, account data, personal data, and other data, share with ADRs. ADRs are responsible for providing various financial services for clients. For instance, clients apply for specific services from ADRs, and ADRs will request the data from users for analysis. In this process, if clients reject the data access, the data sharing will be terminated. Conversely, users will allocate consents to ADRs and allow them to access their financial data. Another one is the consent that authenticated regulators allocate to the financial data center. For example, clients allocate consent for the ADR to visit their data, the financial data center will check the ADR certificate first. Authenticated regulators will confirm ADRs' identity by corresponding information store on the certificate chain. If it is valid, the data center will allocate the request data and expire time to the ADR.

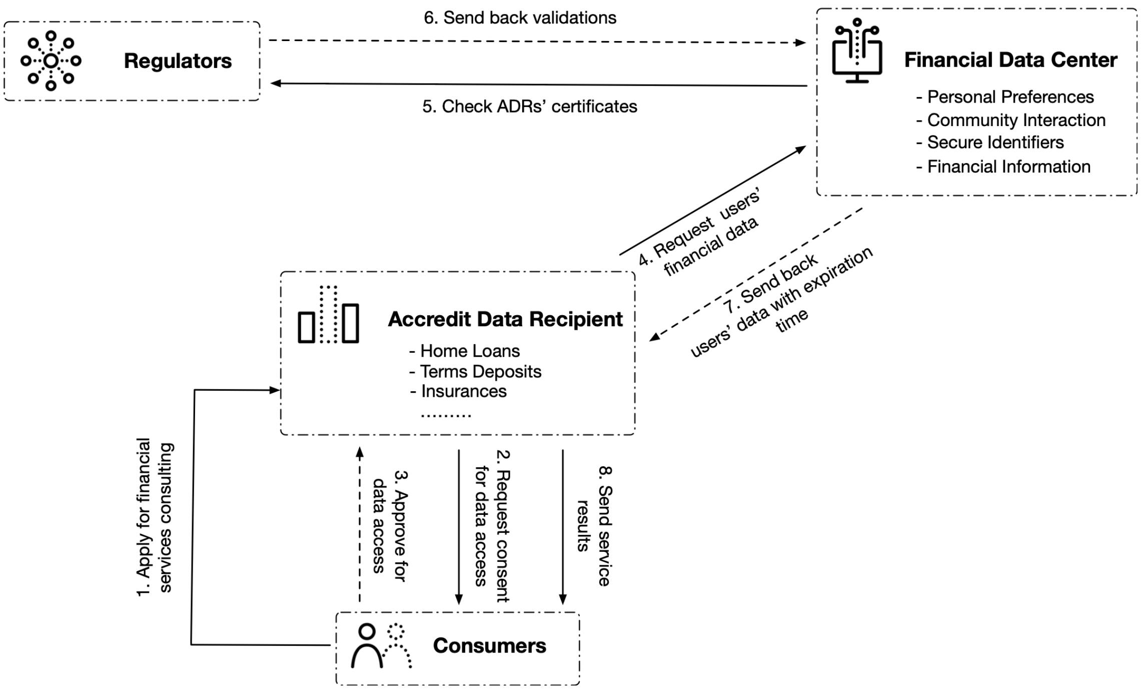


Figure 6.2: Explicit Consent Process

Consent Record: In the AC2M model, consent information is recorded during financial services by the record chain. When ADRs request permission to access a user's financial data, the logic control is managed by smart contracts. Every legal and unlawful record is automatically added to the chain.

The consent record includes 1) ADR's certificate address, 2) User identity number, 3) request timestamp, 4) request type, and 5) expire time. Regulators can audit these five-tuple data periodically. If an ADR violates a specific rule, the corresponding record chain will mark down the confidence level.

6.2 System Design And Implementation

As the underlying structure of the AC2M system, we employ the consortium chain and PoA consensus algorithm. Regulators function as full nodes and store complete chain data. With the assistance of the consortium chain, our financial data-sharing model provides transparent and tamper-proof records on the certificate and record chain. Consent management ensures users' privacy. In our model, there are two consent allocation processes: users-ADRs management and ADR-financial data centre management. Our model's operational mechanisms for ADR registration, certificate management, and automated consent management are described in the following section.

6.2.1 Underlying Structure

Our model constructs a distributed consortium chain-based Ethereum project with the PoA consensus protocol, which guarantees the correctness and efficacy of theF system's execution. The essential PoA committee nodes in the certificate chain and record chain are maintained by regulators and ADRs, respectively.

Consortium chain: Consortium blockchain mainly refers to a distributed ledger system where the blocks' decision-makers are limited in a relatively small committee. The Consortium blockchain is inherently permission with higher efficiency. Many companies and their alliance cooperate to establish their consortium blockchains, including the Hyperledger [7] and Corda [92]. The Consortium chain is an appropriate choice for decentralized financial applications. Due to financial data's sensitivity and privacy, committee members corporately maintain the chain and share the incentives in a small group. We build up the consortium chain based on the Ethereum platform with the PoA consensus mechanism. In this model, we provide 10-100 testing nodes to form the committee. The committee members are elected to be the leader, in turn, to package the transactions and broadcast the blocks.

PoA: PoA is a kind of consensus algorithm used for blockchain systems. It can be regarded as a modified version of Proof-of-Stake (PoS) by using a validator's identity performs as the stake. By utilizing the PoA mechanism, the system enjoys the advantage of (1) *lower energy consumption*: The resource consumption of PoA is significantly lower than that of PoW. (2) *Higher Efficiency:* The transaction confirmation time of PoA is considerably faster than PoW or PoS. (3) *Better Scalability:* It is pretty easy to build and maintain a decentralized application (DApp) using a PoA based network.

To ensure consistency, our model employs the PoA to establish the fundamental consensus mechanism. All transactions and blocks in PoA-based networks are validated and approved by authenticated nodes. The granted nodes are denoted as validators, and they consist of highly regarded ADRs and regulators. Their credibility would be the guarantee of operating the blockchain system with integrity.

Smart Contract: Smart contracts are codes based on Ethereum virtual machines that automatically facilitate, verify, or enforce predefined rules and principles. It performs as an agent to connect user interfaces and business protocols. Moreover, smart contracts are predefined with solidity codes, and nodes in the network will copy and execute these codes. Based on the blockchain, smart contracts are inherently trackable and irreversible without compromising their feasibility. Our model employs smart contracts to realize modular functions, including service registration, policy management, and data maintenance. By adopting smart contracts, our model implements the automated consent processed in financial data sharing. Users can verify ADRs certificate information by requesting data with smart contracts interfaces.

Network Model: Our scheme relies on a partially synchronous network. Other honest nodes will finally receive messages sent from an honest node within a fixed bounded delay. Meanwhile, data transmitted at the network layer lays the foundation of our system. Classic blockchain systems rely on a public P2P network for message dissemination between participating peers. Nodes perform as servers and clients and can retrieve the historical transactions when they participate in the network. Our system employs the same strategy as the implicit dissemination model. In our certificate chain, it is maintained by limited numbers of regulators $Re = \{Re_1,\ Re_2,\ Re_3\ \dots\ Re_n\}$. For example,

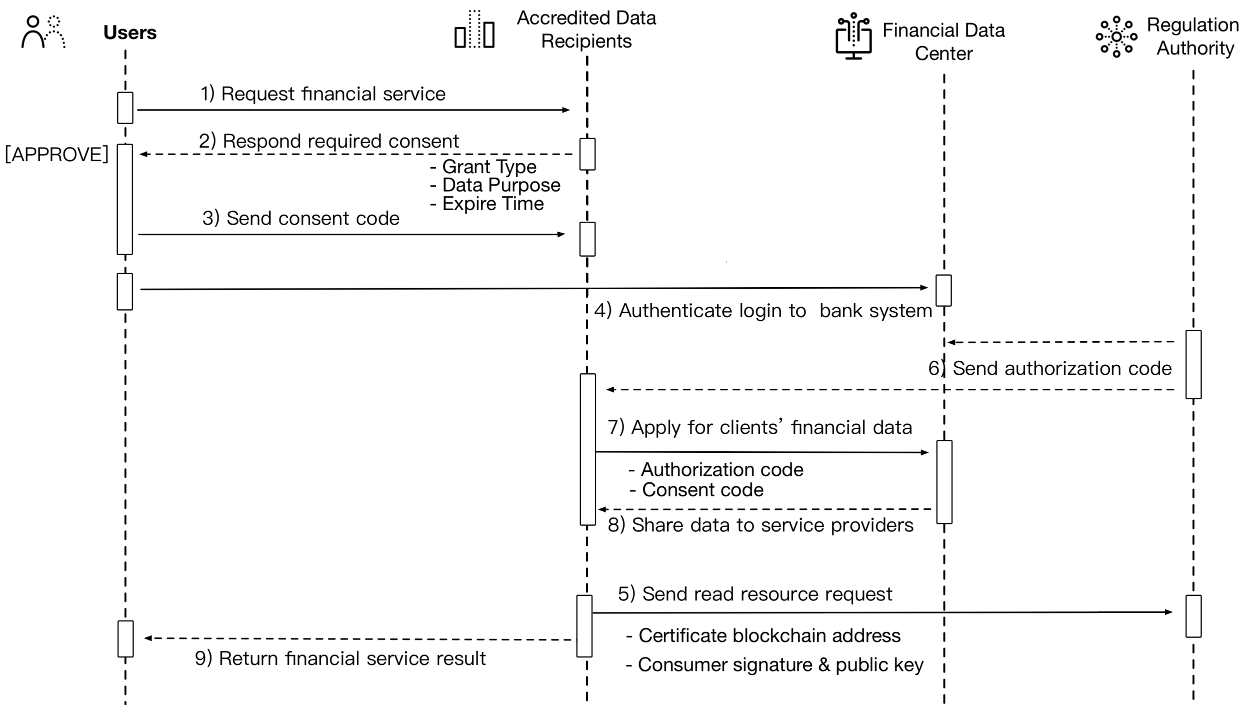

Users
Accredited Data Recipients
Financial Data Center
Regulation Authority
1) Request financial service
[APPROVE]
2) Respond required consent
- Grant Type
- Data Purpose
- Expire Time
3) Send consent code
4) Authenticate login to bank system
6) Send authorization code
7) Apply for clients' financial data
- Authorization code
- Consent code
8) Share data to service providers
5) Send read resource request
- Certificate blockchain address
- Consumer signature & public key
9) Return financial service result

Figure 6.3: AC2M Consent Workflow

when certificate requests enter the network, a validator Re_n will verify requests. After passing the validation, requests would be transfer into a block. Our platform adopts the PoA consensus algorithm, and the new block would be added into the network if it passes with the other two regulators: $Re_n - 1$ and $Re_n - 2$. Relevant messages could be widely spread to nodes, providing the public verifiability of distributed records.

6.2.2 ADR Management

ADR management includes two processes: grant user consent and validates ADRs. Grant user consent exists between users and ADRs when users try to initiate a financial service. It focuses on ADR requests sending to users. Validating ADRs exists between ADRs and regulators, concentrating on identifying if ADRs have a valid certificate on the certificate chain. Smart contracts with predefined rules control both processes.

Grant User Consent: The grant user consent process exists between users and ADRs. Financial services are marked with unique identifications in the form of consent codes. If consumers try to apply for financial services, they need to generate the consent data and send it to corresponding ADRs. The consent data includes service type, purpose, and expiration time. When authorized ADRs receive consent code, they will verify the code and integrate code into further service requests. Note that this functional component connects the individual users and ADRs. Detailed logic is provided in Algorithm 4. There are two functions in Algorithm 1: *AddList()* and *checkConditions()*. The function

Algorithm 4: Grant User Consent

```
Input: Request of financial services
Output: consentCode
while AddList(pk,requestString) do
    consentCode=Hash(pk,requestString) if checkApprovalList(consentCode) then
        return "Service Already Approved";
    else
        if checkConditions(pk,requestString,Timestamp) then
            addList(consentCode,Timestamp);
        else
            return “Request Refused"
        end
    end
end
while checkConditions(pk,requestString,Timestamp) do
    if checkGrantType(requestString) then
        if checkServicePurpose(purposeString) then
            if validTime(Timestamp) then
                return “Time Valid”;
            else
                return “Time Expired Failed";
            end
        else
            return “Purpose Failed";
        end
    else
        return “Service Failed";
    end
end
```

AddList() is responsible for adding encrypted consent code into the approved service list. And the function *checkConditions()* is responsible for check the consent type and expiration time.

Validate ADRs: Each ADR needs to be registered and managed by a regulator. The request is sent to the regulators for authentication with the unique identifications of the blockchain address and the consumer's public key. Approval of request by the regulator permits ADRs to receive consumer data from the financial information center. The financial information center manages and stores the raw data of consumers. Note that this functional component connects the authorized ADRs, the financial information center, and the regulator. Detailed logic is provided in Algorithm 5.

6.2.3 Consent Management

To have a clear view of the automated consent workflow in our prototype, we provide the consent workflow to demonstrate the steps as follows, and the detailed chart is shown in Figure.6.3. We build nine different stages to support the workflow to operate automatically and securely.

- *Step1:* The consumer submits a request for the authorised ADRs' financial services, which includes debt management, mortgage comparison, and foreign exchange.
- *Step2:* ADR responds to the consumer. If the list has already recorded the service, it returns the results. The successful response is returned in the form of a consent code to identify the corresponding service.
- *Step3:* If the request of service fails, it returns a denied message. Here, if the service has not been provided, it may need to register first. The registration contains the conditions of the request type, data purpose, expiration time, and date.
- *Step4:* A consumer receives successful responses and authenticates them to the financial information center.
- *Step5:* For the registration, ADR sends a request to the regulator for reading and calling the resources. The request contains the certificates in the form of the public key and blockchain address.
- *Step6:* The regulator will check the request. If the request is successfully approved, the new service request will be added to the list, and the financial information center and the ADR receives the response. If the request fails, the regulator declines the request.
- *Step7:* The authorized ADR applies to the client's financial data, and the financial information center checks the validation of the consent code and authentication process.
- *Step8:* When the validation is passed, the consumer data is transferred to the ADR accordingly.
- *Step9:* ADR sends requested services to consumers.

Algorithm 5: Validate ADRs

```
Input: Service Request
Output: Results
while addService(pk, serviceString) do
    if validAuthenticationCode(consentCode) then
        return “Authentication Approved";
        if validConsentCode(pk, requestForm) then
            return “Consent Approved";
            checkServiceList(serviceString)
        else
            return “Consent Failed";
        end
    else
        return “Authentication Failed";
    end
end
while validAuthenticationCode(pk, requestForm) do
    if checkConsentCode(authentcationCode,pk) then
        callData(authentcationCode,pk);
        return "loadingData";
    else
        return "Not Approved";
    end
end
while validConsentCode(pk, requestForm) do
    if checkConsentCode(consentCode,pk) then
        callData(consentCode,pk);
        return "loadingData";
    else
        return "Not Approved";
    end
end
if checkServiceList(serviceString) then
    load service;
    func(transfer), func(mortgage), func(debit);
else
    service(serviceString,pk);
    map(serviceList)==>consentIndex(Num) return "Successfully Added";
end
```

Table 6.1: Function Response Time Test

Time/ms	Set 1	Time 2	Time 3	Time 4
addCertificate()	3245.60	3265.12	3411.71	3223.19
updateCertificate()	1332.62	1265.96	1328.91	1275.42
addCredits()	3209.81	3398.27	3371.18	3412.80
tranferData()	8284.91	8382.7	8582.64	8190.06
revokeCertificate()	1187.73	1264.27	1386.73	1423.12

6.3 Evaluation

In this section, we introduce the testing environment of our prototype. Besides, we evaluate our model based on consensus algorithm throughput and smart contract functions response time.

6.3.1 Experiment Evaluation

Testing Environment: We use the proof of authorities consensus algorithm in our prototype. We deployed the blockchain on six cloud servers, including three Ali cloud servers (8x 2.3 GHz CPUs, 16 GiB memory, 64 GiB hard disc) and three AWS cloud servers (4x 2.3 GHz CPUs, 16 GiB memory, 32 GiB hard disc, Geth version 1.8.17). Our certificate chain supports a maximum of 1000 transactions per second and the block interval is set to 5 seconds (TPS). In addition, web3.js is used to interact with the blockchain.

PoA Consensus Algorithm: Usually, the throughput of a blockchain is decided by the block generation time, transaction confirmation time, and transactions in a block. Taking Ethereum as an example, if the Ethereum network is congested, nodes will spend more time on data synchronization. As a result, in an extensive network, the network speed will also influence transaction broadcasting.

To improve our model efficiency, we modified the block generation interval to five-second when creating *genesis.json*. We deployed nodes on six services and in two groups to avoid the network scalability issue. In our experiment, we install the Geth, the client-side application of Ethereum, on six servers. Initially, Geth broke down always with 4GB of memory storage. With the improvement of adjusting the server configuration to 16GB memory, the crash time dramatically decrease. However, in some specific conditions, the Geth application will still crash. As a result, we recommend that the setup memory space need to be a minimum of 64GB to ensure the platform can run well.

Geth's memory utilisation logic uses all available memory. Our six servers disable swap space and terminate the Geth process if memory space is exhausted. Through testing Geth's memory usage, we've determined that the creation of 76 new accounts requires approximately 9GB of memory. Due to the total amount of available memory, we've set it to 16GB. This outcome is satisfactory for our prototype.

In addition, we conducted several experiments. Each transaction consists of 50 bytes of data, which is a reasonable size for general Ethereum transactions. Each node requires 8-11 seconds to generate 3000 transactions, and the TPS reaches 1000. This condition passed our testing requirements.

Functions response time: We test four functions, which are *addCertificates()*, *addCredits()*, *updateCertificate()*, and *tranferData()*. Function *addCertificate* aims to create a certificate block for ADR; *updateCertificate* is used to update certificate status on the chain; *addCredits* targets to create a new credits block for ADRs; *tranferData* is adopted to allocate resource read access to ADR; *revokeCertificate* is used to set the status of certificate to freeze". As shown in Table 6.1, the response time of *transferData()* is longer than other functions, because it contains a certain amount of operations, such as *eth.sendTransaction*, *eth.getBlockNumber*, and *eth.sendSignedTransaction*. As a whole, the average response time of these core functions is considered acceptable for a class of specific applications.

6.3.2 Security and Privacy Analysis

Content of Transaction: In order to achieve data transparency, Transaction-related data are accessible to the public. Although some security precautions have been taken, the transaction's content may still be susceptible to tampering. In the proposed consent management model, the record chain and certificate chain store two categories of data. Once an ADR is successfully registered on the certificate chain, the certificate information and expiration date are permanently recorded on the ledger. When clients request an ADR service, the process details will be meticulously recorded on the record chain. The transaction content of our model comprises the clients' approval, the ADRs' certificate, and the transaction record. Due to the blockchain's immutability, the certificate and transaction data cannot be altered maliciously. In addition, sensitive client data is stored off-chain and cannot be shared without appropriate permissions or valid certificate data.

Time-limited Data Authorization: Protecting users' sensitive data is significant in any integrated and secured system. In some real-life applications, data spoofing and anonymous access are hard to prevent. It is possible to obtain and analyze consumers' data by malicious scripts and SSL-based attacks. In our model, through creating expired-time and access tokens for ADRs, users' data will only be accessed by authorized ADRs at a specific time. When access time is expired, even authorized ADRs cannot request users data.

Moreover, each data request will be recorded on the record chain. If ADRs have a certain amount of invalid data requests, regulators will freeze their accounts to determine suspicious activity. As a result, expiration time and access token can protect users' privacy and system reliability.

6.3.3 Limitations and Future Work

We analyse the PoA consensus algorithm and function response time to evaluate the performance of our model. In the discussion, we demonstrate that, if necessary, Geth would consume all available memory space, causing it to crash. We propose that expanding memory space can help alleviate this issue. The hard disc storage space in our prototype is 32 GB, and the swap function is disabled on servers. In our future work, we will enable the swap function on servers instead of increasing memory space to meet our needs. The swap function decreased Geth's performance, but it ensures system stability.

In addition, zero-knowledge proof (ZKP) will be incorporated into our model. Currently, our prototype uses smart contracts to implement an automated consent management process. When consumer data is transferred between entities, the open data sharing system places a premium on keeping sensitive data private. We will examine ZKP technology to determine how we can incorporate it into the model.

6.4 Conclusion

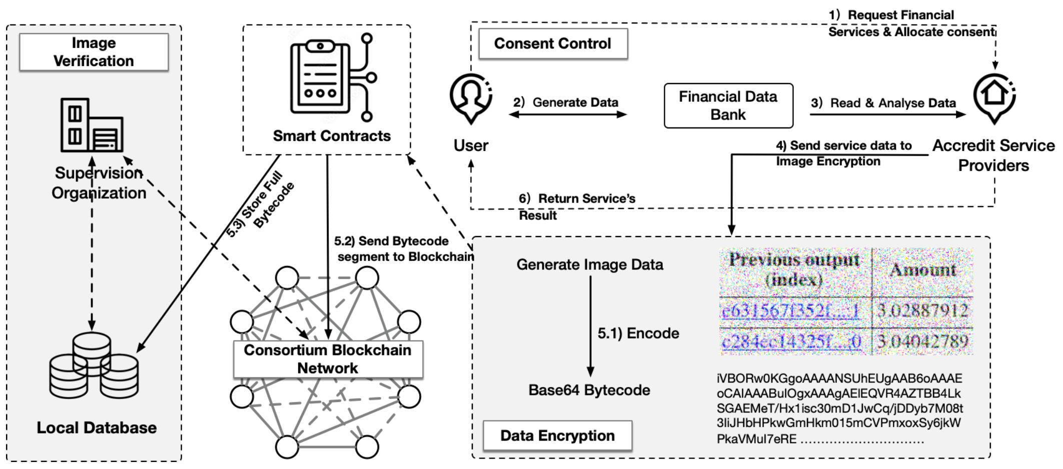
In this section, we proposes an automated consent management model based on blockchain to realize the service and consent management in open financial service. We employ smart contracts to customize the policy and conditions to flexibly meet various financial requirements and the PoA-based underlying distributed system to manage the consortium chain efficiently. The model provides capabilities on automation, transparency, and accountability. Our model is expected to benefit the financial infrastructure in practice.

Chapter 7

IB2P: An image-based privacy-preserving blockchain model for financial services

In recent years, decentralised finance has expanded dramatically to include various cryptocurrency or blockchain-based financial applications. People's enthusiasm and confidence in decentralised financial services have been bolstered by blockchain technology. Due to the sharing of personal information with other institutions, it will be difficult to ensure users' privacy and data security. In addition, numerous encryption techniques, such as homomorphic encryption, are designed to safeguard the data stored on a blockchain, but few improve client-side data security. Motivated by this requirement, we develop an image-based privacy-preserving scheme for storing the financial service data of users as images. Transferring numerical service data into images ensures that users can interpret the content and prevents data from being recognised by machines. We evaluate our scheme by conducting numerical analyses of the effectiveness of image attack and the efficiency of smart contract functions.

In this chapter, we propose an image-based privacy-preserving scheme (IB2P) based on a blockchain platform. Our scheme adopts the Proof of Authority (PoA) consensus algorithm and Ethereum to provide a transparent and automate blockchain system. Due to machine learning attacks treat transaction records as a feature, we integrate the image encryption technique to prevent crawling numerical data on the platform. With the help of the Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA), users can understand the picture content and prevent computers from directly crawling the data on the web page. Compared to tons of standard encryption tools [99], we adopt CAPTCHA as the data encryption methods to ensure data security and readability. IB2P aims to provide an image encryption algorithm to prevent computers from using machine learning tools to infer users' real identities. Our scheme has the following features:



Previous output (index)	Amount
e631567f352f...:1	3.02887912
c284ec14325f...:0	3.04042789

iVBORw0KGgoAAAANSUhEUgAAB6oAAAE
oCAIAAABulOgxAAAgAElEQVR4AZTBB4Lk
SGAEMeT/Hx1isc30mD1JwCq/jDDyb7M08t
3liJHbHPkwGmHkm015mCVPmxoxSy6jkW
PkaVMul7eRE ....

Figure 7.1: IB2P System Architecture

- **Prevent data analysis by Image-based encryption:** In order to improve blockchain anonymity and user privacy, our system includes image encryption. The Attacker would collect numerical data and conduct data analysis to deduce the identities of users. With image-based encryption, numerical data are represented as images, thereby protecting the privacy of users and ensuring the accessibility of records. Users are able to read all numerical data as images.

- **An efficient privacy protection method:** Compared to network-level methodologies, such as ZKP and ring signature, our method has a high transformation efficiency. We treat the CAPTCHA algorithm as a microservice. When new records are generated, the algorithm will convert data into images first. Then images would transfer to bytecode and push on the blockchain.

- **Keyless data encryption:** Traditionally, encryption methodologies require users to store public or private keys locally, which calls key management. Our system adopts image encryption to enhance user privacy and reduce the risk of managing keys.

7.1 IB2P Model Overview

This section presents the state-of-the-art vulnerabilities exposed in blockchain-based applications. The architecture review illustrates the stakeholders in the IB2P model and the corresponding functions.

7.1.1 Problem Statement

Due to the dramatic increased decentralized applications (DApps) in recent years, industries start to pay more attention to integrate blockchain techniques into their services. When it comes to blockchain platforms, people usually believe that decentralization, anonymity, and transparency are guaranteed.

Although transparency is a representative feature in the blockchain area, privacy and transparency are opposites. For integrate DApps, if the security and privacy aspects are not well-designed, systems may suffer damage.

In recent years, there are several proposed methodologies [100], [101], [102] to solve privacy and security issues in DApps. These methods adopt key-centered encryption algorithms to guarantee data privacy and security in DApps. Take Tian's work [102] as an example, they propose an idea about reconstructing share keys to enhance the security in blockchain data sharing. Although their work achieves sound privacy preservation, key management is still vulnerable when multi-parties are involved in the blockchain.

To address the key-related issue, we propose an image-based encryption scheme as an alternative way to address users' privacy concerns. We adopt CAPTCHA to encrypt numerical data in our platform, especially for the token amount. We believe that our approach can protect users' privacy while also enabling them to understand encrypted data. The advantages of using image-based data encryption are as follows:

- Image encryption makes data garbled to a machine, but readable and understandable to users. By storing the image bytecode on the blockchain, we can ensure the authenticity of images.
- Image encryption does not involve key management and can reduce the risk of disclosure in the process of key sharing.
- The plain data can be vulnerable to machine learning-style attacks, even on blockchains. Machine learning can analyze data on a statistical blockchain platform to deduce identity information. By encrypting digital information through images, the attack cost of machine learning would be significantly increased, and the users' privacy would be fully protected.

Figure 7.1 illustrates the deployment process of our model. When users request financial services on DApps, the corresponding data would be generated and send to the image-encryption algorithm. The algorithm is responsible for converting the plain data into CAPTCHA form. Moreover, images would be translated into base64 bytecode and store on the blockchain. When financial DApps retrieve data from the blockchain, users can learn their numerical information through images.

The IB2P's purpose is that while encrypting data to protect users' privacy, users can also clearly understand the encrypted image data. The contributions of the model are as follows: 1) We propose an alternative way to encrypt the numerical data in financial DApps, which can enhance users' privacy and security. 2) The image encryption methodologies are keyless, avoiding any risk related to keys and improving the encryption efficiency.

7.1.2 Architecture Overview

Our model is based on the consortium blockchain and adopts PoA as the consensus algorithm. Figure 7.1 illustrates the overview of the IB2P structure. There are two main processes in the system: user consent control, numerical data encryption, and image storage. In this model, we combine on-chain and off-chain storage to reach a balance between data storage and security.

Involved Stakeholders and Roles explanation

Our model has four key stakeholders: general users, accredit service providers, financial data banks, and supervision organizations. They are responsible for different stages in our model. We illustrate their roles as follow:

- *General Users*: General users are clients that have specific needs for financial services. They would allocate consents to accredit service providers to have accurate and reliable results.
- *Accredit Service Providers(ASPs)*: Accredit Service Providers are responsible for helping users analyze and customize their financial services. Supervision Organizations would grant ASPs' qualifications in order to ensure ASPs are secure.
- *Financial Data Bank(FDB)*: Financial Data Bank is in charge of storing and managing users' financial data. When ASPs request access to users' data, FDB is responsible for checking users' signatures and ASPs' qualifications.
- *Supervision Organizations(SOs)*: The supervision organizations, also known as regulators, are mainly responsible for allocating the ASPs' qualifications. In addition, SOs also act as nodes in the consortium blockchain. They regularly check whether the image bytecode data on the blockchain matches the image data stored in the database.

These four indispensable stakeholders constitute our model. Users would request services from ASPs and allocate consent to them for more accurate results. When ASPs obtain users' consent, the corresponding results would be generated and encrypt by the image algorithm. After encryption, data segments would be stored on-chain, and complete data stored in local databases. Through smart contract scripts, SOs would be in charge of verifying image segment data before pushing on the consortium chain.

Data Encryption and Storage

As we all know, privacy protection and security are always a priority in both traditional and distributed systems. Numerical data encryption is the core section of the IB2P system. There have been various encryption methods such as homomorphic encryption, symmetric key encryption, and ring signature to ensure that the authenticated person receives the information. However, these encryption methodologies have some shortcomings in efficiency and user readability. In addition, in blockchain systems, machine learning techniques can help attackers infer users' identities.

We choose the image-based encryption method to encrypt the numerical data in our platform. This method can protect against machine learning attacks and ensure that users can comprehend the

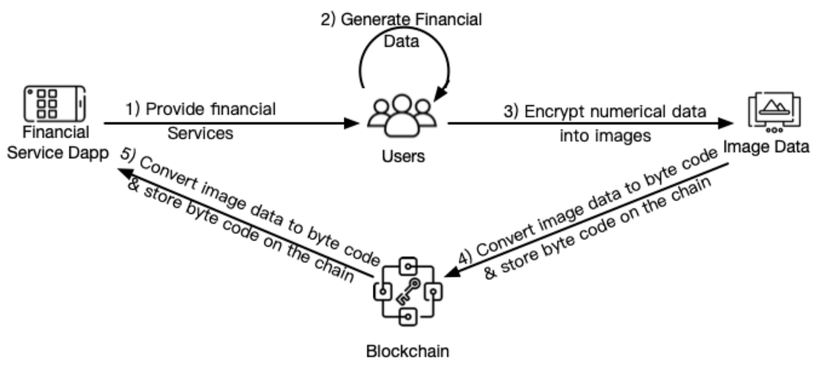

2) Generate Financial Data
Financial Service Dapp
1) Provide financial Services
Users
3) Encrypt numerical data into images
Image Data
5) Convert image data to byte code & store byte code on the chain
4) Convert image data to byte code & store byte code on the chain
Blockchain

Figure 7.2: IB2P model deploy processes

encrypted image data. We would convert the image data into base64 bytecode and store it on blocks. Due to the limited storage space on blocks, storing bytecode on blocks needs to be well-considered. Generally, an image converted to bytecode takes up more space than the image itself, and storing the entire bytecode on a block is not feasible. As a result, we store the integrated image data in local databases. The blockchain stores the first 100 characters and the last 100 characters of bytecode. Due to the immutability of blockchain ledger, regular verification of the consistency of blockchain bytecode segments and database image data can ensure the authenticity of the data.

7.2 System Design & Implementation

This section demonstrates the use case of the proposed system and presents the detailed process of image generation and data conversion between images and bytecode.

7.2.1 Use Case Illustration

The introduction of Fintech [103] promotes the development of decentralized financial applications. Due to the tamper-proof and decentralization of blockchain, consumers would choose ASPs to help them analyze assets. However, data security and privacy protection are important indicators to consider the reliability of a system. Various techniques have been applied to tackle privacy and security issues, such as access control, symmetric encryption, and consensus algorithm. We adopt image encryption to protect data while solving the security problem, and efficiency can also be guaranteed. For example, at the end of financial services, users cease to authorize access to personal data. Afterward, corresponding service data (SD) would be recorded by the system. $SD_i = \{SD_1,\ SD_2,\ SD_3\ \ldots\ SD_n\}$ is a series of data contains users basic information, service detail, and other numerical data, where SD_n is the n^{th} attribute in the transaction data. The following steps would illustrate the process of managing these numerical data and non-numerical data.

1) In most decentralized financial applications, users would request services from ASPs. When

APSs accept services requests, they need users' signatures and consents to access users' data. When users grant ASPs access to data, the expiration time of data access would also be sent to ASPs.

2) After obtaining users' consent, ASPs would request data from FDB and analyze users' data. When the analysis results are released, the service result data would be sent back to users.

3) At the same time, the data preparation would start for image encryption. The service data includes numerical data and non-numerical data. Through extracting the numerical data, the image encryption algorithm transfers data into human-readable images.

4) When image data is generated, images would be transferred to bytecode for storage. Considering the limited storage space on the blockchain, we will store part of the bytecode, while local databases would record the original image.

5) Any minor bytecode modification will cause image decryption to fail. As a result, SOs are responsible for periodically comparing blockchain bytecode data and local image data for data verification. When the translated data of the image is consistent with the storage on the blockchain, it proves that users' numerical data is credible and authenticated.

7.2.2 Data Processing and Image Generation

Financial services would generate numerical data and non-numerical data. IB2P focuses on numerical data cleaning and encryption. The reason for encrypting numerical data is that users' behaviors and identity information can be inferred through data analysis algorithms. This section presents the process of dealing with data in the image encryption algorithms. There are four procedures to encrypt numerical data: data wrangling, pre-image generation, adding noise points, and bytecode verification.

Data wrangling: Typically, numerical data includes the user's transaction account address, the amount of the user's transaction, and the transaction time. After receiving the data related to the transaction, we will conduct data pre-processing. In order to make the final image easier to comprehend, we will remove some extra spaces and invalid characters in this step.

Pre-image generation: After the data wrangling, we generate a canvas to paint characters. The canvas size is determined by CAPTCHA width (Caw) and height (Cah) in advance. Through retrieving the data, we calculate the height (Teh), width (Tew), and length (Tel) of the string based on the preset font size. In order to maximize the canvas space, we need to locate the starting points of characters. We import two constant parameters $Spac$ and λ as the gap between characters and counters, respectively.

$$SWidth = \lfloor \frac{Caw - Tew - (Tel - 1) \times Spac}{2} \rfloor$$

$$SHeight = \lfloor \frac{2 + \lambda}{Teh} \rfloor$$

After the value of starting width ($SWidth$) and height ($SHeight$) are calculated, we can generate a plain image contains numerical data with python image library (PIL).

Adding noise points: In order to increase the cost of machine learning to recognize information in images, we randomly add noise points to the canvas. Through referring to Caw and Cah, the algorithm adds noise points to each unit area of the canvas. Moreover, to ensure that users can understand the image after adding noise, we need to control the frequency of noise points. We introduce a parameter called point chance to control the probability of drawing noise points.

Bytecode generation and verification: After adding noise points to the image, we need to store the generated image. Our model plans to store the bytecode segments data on the blockchain and the original image data in local databases. Considering that the integrated image bytecode will run out of the block's storage space, we choose to perform segmentation on bytecode. We store the first 100 characters and the last 100 characters of bytecode on the blockchain to optimize storage. In addition, SOs are responsible for image data verification periodically. Due to the immutability of the blockchain ledger, SOs can compare the local bytecode data with the blockchain bytecode segment by compiling the image data in the database to bytecode, which ensures data reliability and consistency.

Through these three steps, we can complete the process of converting numerical data into image data. The image encryption methods can ensure the service data is not analyzed and enable users to understand the meaning of the data in the picture.

7.3 Evaluation

The IB2P model is based on the consortium chain and PoA consensus algorithm. To deploy the blockchain, we set up two local servers (2 x 2.5 GHz CPUs, 16 GB memory, 512 GB flash storage). In addition, we adopt Geth as the Ethereum client application and Truffle as the framework. In order to imitate the real-world blockchian system, we use the Ethereum Rinkby testnet to realize the consortium chain and PoA algorithm. To better understand the model performance in resisting attack and the generation efficiency in CAPTCHA, we conduct numerical evaluations from image attack effectiveness and smart contract functions' efficiency analysis.

7.3.1 Security Analysis

For CAPTCHA with simple design, traditional image pre-processing methods such as pixel counting and vertical and horizontal projection can extract text. It is also easy to crack for some complex CAPTCHA (e.g., distortion, blurring, tilting, and waving) by combining with learning algorithms or matching algorithms. Our design contains a few standard anti-recognition techniques such as noisy background, distortion, and blurring (see Fig. 7.1 Original).

Generally, the conventional CAPTCHA attack can be divided into three steps, including image pre-processing, character segmentation, and recognition. For segmentation, there are a variety of seg-

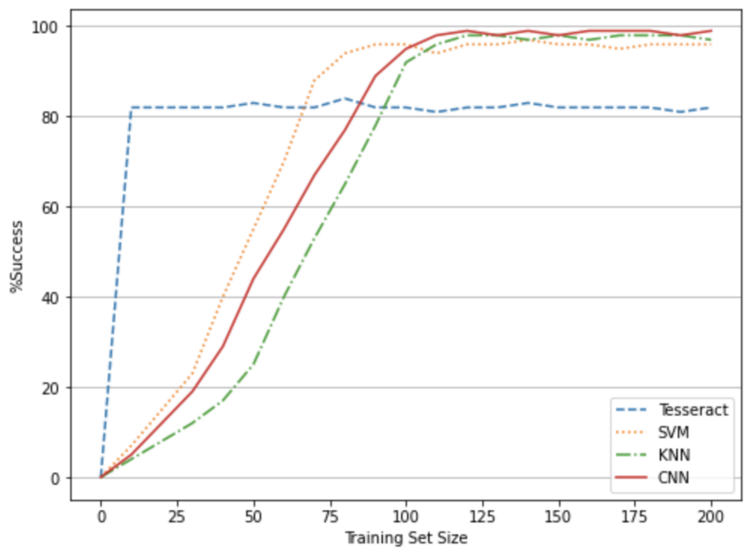


Figure 7.3: Success rate of four classifiers vs the number of training set

mentation techniques, such as clustering-based methods, region-based methods, and edge-detection methods. While in terms of character recognition, machine learning is the mainstream approach that performs strikingly well. To compare the effectiveness of different attacking approaches on the proposed model, we select four popular classifiers, including Tesseract, Support Vector Machines (SVM), K Nearest Neighbors (KNN), and Convolution Neural Network (CNN). Tesseract is an open-source OCR tool developed by Google, while the other three are the typical machine learning algorithms used to recognize characters in our case. Meanwhile, we simulate 200 and 20 blockchain transactions as training sets and test sets, respectively.

The accuracy rate is a significant metric used to evaluate CAPTCHA attack effectiveness, which is the fraction of characters recognized correctly. We feed the four classifiers using different training sets varying from 0 to 200 examples and compute the average accuracy by testing 20 examples. As can be observed from Fig. 7.3, Tesseract can recognize text from images with about 80% accuracy and does not require the model training process. In the stage of about 100 examples, a stable success rate (about 95%) can be achieved, which is much higher than Tesseract. In contrast, as more examples were added to the training set, the success rates of the other three methods showed a linear upward trend.

In addition, the time cost is another important metric to evaluate CAPTCHA attack effectiveness, including training time and test time. From the experiment observation (shown in Fig. 7.4), Tesseract is the fastest classifier (only takes about 5 seconds totally) since it does not need to spend time on the training model. On the contrary, SVM, KNN, and CNN need to spend time varying from 10s to 20s on training models. CNN is the slowest, followed by KNN. In summary, we argue that the considerable time cost can be a factor preventing attackers from decaptcha.

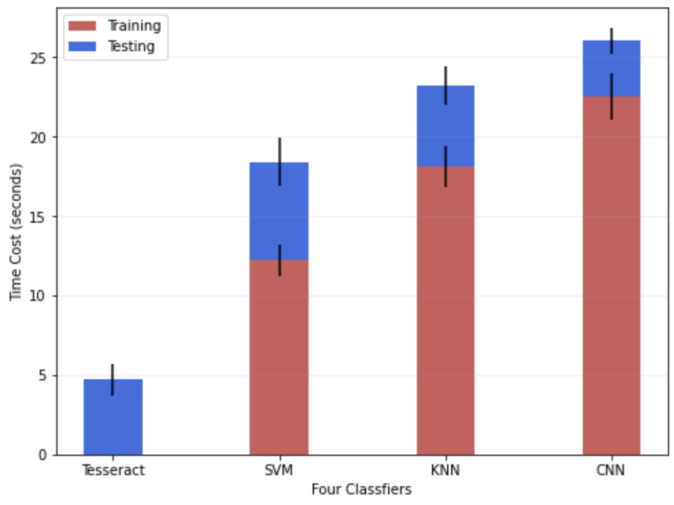


Figure 7.4: Time cost of four classifiers under 100 training examples and 20 test examples

7.3.2 Efficiency Analysis

We implement the proposed model on the Ethereum consortium blockchain and build the smart contract using Solidity. To analyze the efficiency of CAPTCHA generation, we evaluate image encryption and decryption functions built into the smart contract from two perspectives, including gas consumption and time cost (shown in Fig. 7.5).

The left graph shows the change of gas consumption with the increase in the number of transactions passing into the function call. Overall, the gas consumption of both functions presents an upward trend. When there is only one transaction in the function call, the gas consumption for executing the encryption and decryption functions is 354,719 and 246,567, respectively. While calling functions with five transactions, the gas consumption becomes 653,221 and 516,714, respectively. Based on this observation, we infer that the number of bytes that need to be compiled (converted to base64 bytecode) is directly related to gas consumption. Nonetheless, the gas cost is still within an acceptable range.

We also test the encryption function and decryption function with a different number of contained transactions in terms of time cost. As can be observed from the right graph, the time costs of the two functions show a rising in volatility. From a vertical comparison, the time cost of the encryption function (average 350ms) is slightly higher than that of the decryption function (average 240ms). Overall, the time cost of the proposed model is relatively low compared to other encryption methods (e.g., ZKP and Ring Signature) applied in the blockchain.

7.4 Conclusion and future work

In conclusion, we proposed the IB2P scheme to enhance users' financial service data privacy. In this chapter, we illustrate privacy and security issues in DApps and how can IB2P model resolve these

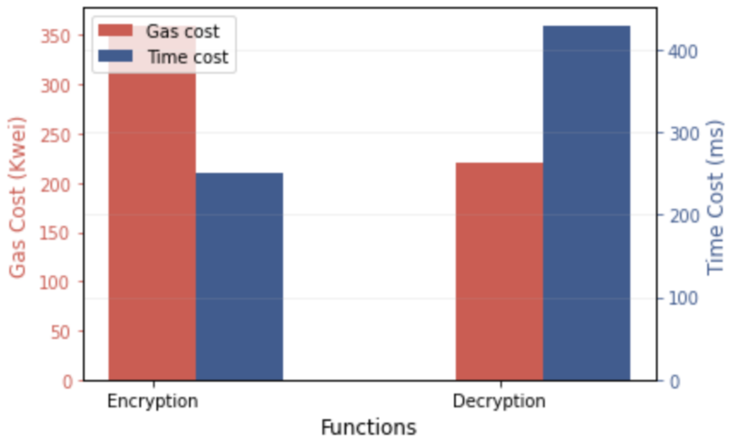


Figure 7.5: Gas and Time cost of encryption and decryption functions

issues. Our model is conducted with the features of preventing data analysis, keyless data encryption, and efficient encryption methods. Traditional encryption methodologies are unable to prevent data analysis attacks based on the web crawler. In our model, we encrypt the numerical data as images to enhance data privacy and security. By storing the image bytecode on the blockchain, we can ensure the service data is immutable and auditable. For the future work, we plan to improve the image recognition resistance against more machine learning-based attacks to further enhance image data security. Overall, the proposed IB2P scheme brings an innovative and practical method to enhance DApps' service data security and privacy.

Chapter 8

Summary

8.1 Main Contributions

These research papers intends to enhance and improve the privacy & security of decentralized applications in financial industries. The main contributions are summarized as follows:

ArtChain: blockchain-enabled platform for art marketplace: We present ArtChain, a blockchain-based art trading system, which has been piloted and in operation as a working product in practice. It is expected to provide a total solution to these issues by creating a new ecosystem for art and bring the art industry into a new era. ArtChain consists of the underlying architecture of blockchain to support a business system centered around art assets trading platforms.

A provenance-provided data sharing model for open banking via blockchain: We propose a provenance-provided data sharing model (PPM) via blockchain to meet the requirements of open banking. The model employs the programmable smart contracts as the middle witness between users and third-party services, and provides the modifications on data layer (data content, transaction structure), smart contract layer (ACL, logic), and application layer (customized APIs). Based on that, our PPM model possesses the properties of transparent authentication, privacy-provided control, and auditable provenance.

A blockchain-based hierarchical data access model for financial services: This paper proposes a hierarchical data access model for financial services, which contains consent management and dynamic credits management. We implement fine-grained data access control through rating accredited data recipients (ADRs). By accessing corresponding blockchain service logs, ADRs' credits will dynamically be updated. The credits evaluation algorithm is responsible for calculating ADRs' credits based on their completion rate, business ethics rate, and feedback positive rate. Moreover, through applying smart contracts, the efficiency of consent management can be improved, and privacy policies can be managed elastically.

An automated consent management model for blockchain financial services platform:

This paper proposes an automated consent management model based on blockchain technology for various coming financial services. We employ the Proof of Authority consensus mechanism to balance the business flexibility and efficiency with client privacy control. Our model enables automated, transparent, and accountable management for consumer consents in financial services. We implement a proof of concept prototype to demonstrate the feasibility and applicability of our model.

An image-based privacy-preserving blockchain model for financial services: We develop an image-based privacy-preserving scheme to store users' financial service data as images. Transferring the numerical service data into images ensures that users can comprehend the content and prevent data recognized by the machine.

8.2 Future Work

According to the research outcomes, we have proposed several mechanisms that aim to improve data sharing privacy and security. Although these methodologies have already presented solutions, some improvements can still be discussed in future work.

For the image-based privacy-preserving blockchain model for financial services, we plan to improve the image recognition resistance against more machine learning-based attacks to enhance image data security further. Current solutions can prevent the most common image recognition attacks. We will add some noise points in the future, for example, integrating a zero-width joiner into image bytecodes.

For the automated consent management model for blockchain financial services platform When consumers' data is transferred between entities, keeping the sensitive data private is considerable in the open data sharing system. For the privacy model in decentralized data sharing, we will integrate zero-knowledge proof (ZKP) into our model. Our prototypes currently implement automated consent and provenance management processes with the support of smart contracts. We will investigate ZKP technology to see how we can employ it in the model.

For the blockchain-enabled platform for art marketplace, we find out that:

- The performance bottleneck of our system is at the Geth IO execution.
- The way to improve the system performance is to improve node hardware configuration.

Ethereum uses LevelDB as the database to store key/value. The key to accessing database is irregular on account of the discreteness of hash. The LevelDB has an excellent performance in reading/writing continuously, while bad for random key. Therefore, the time t for accessing LevelDB would be longer as the amount of data storage increases. In fact, the test results show that if n is large enough, the value of t will increase and the eciency will degrade largely for some data which not hit LevelDB cache at times.

Geth consumes huge memory on certain transactions, e.g., personal.newAccount, and will crash when receiving multiple concurrent memory-consuming transactions. A suggestion is to enable swap memory on the node to improve system stability, at the sacrifice of the performance (See performance degradation when memory is swapped). We suggest 4GB of swap space on the nodes, based on the performance test result. This improves the system stability and does not degrade the performance significantly.

In addition, smart contracts are implemented on the blockchain. Once a smart contract is deployed online, it cannot be altered. Smart contracts include a series of Ethers transfer and account balance update operations. However, some developers and users lack security awareness and knowledge of solidity, resulting in vulnerabilities in smart contracts. If a flawed smart contract is deployed on the blockchain in the financial sector, it can cause significant financial harm to users and institutions. Therefore, the design of a user-friendly smart contract is urgent. If the tool can meet user requirements, the following questions must be considered.

- How many vulnerabilities exited in smart contracts' codes?
- How to detect these exited smart contract vulnerabilities and unknown vulnerabilities?
- What are common patterns in smart contract vulnerabilities? How can I define these patterns?
- How to make the detection results comprehensive and acceptable by non-professional users?

I will focus on these questions in my future research.

There are several features we plan to further develop in the next versions of the product:

- Anti-counterfeiting for the original works of art by integrating with the smart modules of IoT, and activating relevant smart hardware and other functionality (e.g. positioning/location tracking) as required by artists or collectors; and
- Improving the efficiency of high-value art assets insurance by using smart contracts to automate the process of claim and periodic renewal.

Chapter 9

Published Papers

Here are the accepted papers from the beginning of my PhD study.

- Minfeng Qi, **Zhiyu Xu**, Tengyun Jiao, Sheng Wen, Yang Xiang, Gary Nan, "A Comparative Study on the Security of Cryptocurrency Wallets in Android System." Proceedings of the 21st IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 2022
- Minfeng Qi, **Zhiyu Xu**, Ziyuan Wang, Shiping Chen, Yang Xiang, "DeDa: A DeFi-enabled Data Sharing and Trading System." Proceedings of the 2022 ACM International Symposium on Blockchain and Secure Critical Infrastructure. 2022
- Minfeng Qi, **Zhiyu Xu**, Ziyuan Wang, Shiping Chen, Yang Xiang, Liming Zhu, "Databox-based Delivery Service via Blockchain." Proceedings of the 2022 IEEE International Conference on Web Service. 2022
- **Zhiyu Xu**, Tengyun Jiao, Ziyuan Wang, Sheng Wen, Shiping Chen, Yang Xiang, "AC2M: An Automated Consent Management Model for Blockchain Financial Services Platform." 2021 IEEE International Conference on Smart Data Services (SMDS). IEEE, 2021
- **Zhiyu Xu**, Minfeng Qi, Ziyuan Wang, Sheng Wen, Shiping Chen, Yang Xiang, "IB2P: An image-based privacy-preserving blockchain model for financial services." 2021 IEEE International Conference on Blockchain (Blockchain). IEEE, 2021
- **Zhiyu Xu**, Lin Yang, Ziyuan Wang, Sheng Wen, Rob Hanson, Shiping Chen, Yang Xiang, "BHDA-A Blockchain-Based Hierarchical Data Access Model for Financial Services." 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). IEEE, 2020
- **Zhiyu Xu**, Qin Wang, Ziyuan Wang, Donghai Liu, Yang Xiang, Sheng Wen, "PPM: a provenance-provided data sharing model for open banking via blockchain." Proceedings of the Australasian

Computer Science Week Multiconference. 2020

- **Zhiyu Xu**, Tengyun Jiao, Qin Wang, Cuong Bui Van, Sheng Wen, Yang Xiang, "An efficient supply chain architecture based on blockchain for high-value commodities." Proceedings of the 2019 ACM International Symposium on Blockchain and Secure Critical Infrastructure. 2019
- **Zhiyu Xu**, Tengyun Jiao, Lin Yang, Donghai Liu, Sheng Wen, Yang Xiang, "RBAC-GL: a role-based access control gasless architecture of consortium blockchain." International Conference on Algorithms and Architectures for Parallel Processing. Springer, Cham, 2019.
- Ziyuan Wang, Lin Yang, Qin Wang, Donghai Liu, **Zhiyu Xu**, Shigang Liu, "Artchain: Blockchain-enabled platform for art marketplace", 2019 IEEE International Conference on Blockchain (Blockchain). IEEE, 2019

Bibliography

[1] S. Nakamoto, *Bitcoin: A peer-to-peer electronic cash system*, (2008).

[2] G. Wood, *Ethereum: A secure decentralised generalised transaction ledger*, Ethereum project yellow paper **151**, 1–32 (2014).

[3] O. Novo, *Blockchain meets IoT: An architecture for scalable access management in IoT*, IEEE internet of things journal **5**, 1184–1195 (2018).

[4] V. Harshini, S. Danai, H. Usha, and M. R. Kounte. *Health record management through blockchain technology.* In *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, pages 1411–1415. IEEE, (2019).

[5] J. Xu, K. Xue, S. Li, H. Tian, J. Hong, P. Hong, and N. Yu, *Healthchain: A blockchain-based privacy preserving scheme for large-scale health data*, IEEE Internet of Things Journal **6**, 8770–8781 (2019).

[6] K. R. Özyilmaz, M. Doğan, and A. Yurdakul. *IDMoB: IoT data marketplace on blockchain.* In *2018 crypto valley conference on blockchain technology (CVCBT)*, pages 11–19. IEEE, (2018).

[7] *Hyperledger.*

[8] *Ethereum Improvement Proposals.* http://eips.ethereum.org/all, (2015).

[9] A. Ouaddah, A. Abou Elkalam, and A. A. Ouahman. *Towards a novel privacy-preserving access control model based on blockchain technology in IoT.* In *Europe and MENA cooperation advances in information and communication technologies*, pages 523–533. Springer, (2017).

[10] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, *A survey on the security of blockchain systems*, Future Generation Computer Systems **107**, 841–853 (2020).

[11] S. Tikhomirov. *Ethereum: state of knowledge and research perspectives.* In *International Symposium on Foundations and Practice of Security*, pages 206–221. Springer, (2017).

[12] P. Otte, M. de Vos, and J. Pouwelse, *TrustChain: A Sybil-resistant scalable blockchain*, Future Generation Computer Systems **107**, 770–780 (2020).

[13] G. G. Dagher, J. Mohler, M. Milojkovic, and P. B. Marella, *Ancile: Privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology*, Sustainable cities and society **39**, 283–297 (2018).

[14] Q. Xia, E. B. Sifah, K. O. Asamoah, J. Gao, X. Du, and M. Guizani, *MeDShare: Trust-less medical data sharing among cloud service providers via blockchain*, IEEE Access **5**, 14757–14767 (2017).

[15] A. Bahga and V. K. Madisetti, *Blockchain platform for industrial internet of things*, Journal of Software Engineering and Applications **9**, 533–546 (2016).

[16] L. Lao, Z. Li, S. Hou, B. Xiao, S. Guo, and Y. Yang, *A Survey of IoT Applications in Blockchain Systems: Architecture, Consensus, and Traffic Modeling*, ACM Computing Surveys (CSUR) **53**, 1–32 (2020).

[17] Z. Xu, T. Jiao, Q. Wang, C. B. Van, S. Wen, and Y. Xiang. *An Efficient Supply Chain Architecture Based on Blockchain for High-value Commodities.* In *Proceedings of the 2019 ACM International Symposium on Blockchain and Secure Critical Infrastructure*, pages 81–88. ACM, (2019).

[18] Z. Wang, L. Yang, Q. Wang, D. Liu, Z. Xu, and S. Liu. *ArtChain: Blockchain-Enabled Platform for Art Marketplace.* In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 447–454. IEEE, (2019).

[19] E. Union. *Regulation(General Data Protection Regulation)*, (2016).

[20] P. Freebairn, *Response to the Farrell Report into Open Banking*, Policy (2018).

[21] *Terms of Reference.*

[22] M. Pratap. *Everything You Need to Know About Smart Contracts: A Beginner's Guide*, (2018).

[23] A. Moinet, B. Darties, and J.-L. Baril, *Blockchain based trust & authentication for decentralized sensor networks*, arXiv preprint arXiv:1706.01730 (2017).

[24] A. Mohsin, A. Zaidan, B. Zaidan, O. S. Albahri, A. S. Albahri, M. Alsalem, and K. Mohammed, *Blockchain authentication of network applications: Taxonomy, classification, capabilities, open challenges, motivations, recommendations and future directions*, Computer Standards & Interfaces **64**, 41–60 (2019).

[25] M. T. Hammi, B. Hammi, P. Bellot, and A. Serhrouchni, *Bubbles of Trust: A decentralized blockchain-based authentication system for IoT*, Computers & Security **78**, 126–142 (2018).

[26] R. Zhang, R. Xue, and L. Liu, *Security and privacy on blockchain*, ACM Computing Surveys (CSUR) **52**, 1–34 (2019).

[27] G. Karame and S. Capkun, *Blockchain security and privacy*, IEEE Security & Privacy **16**, 11–12 (2018).

[28] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram. *Blockchain for IoT security and privacy: The case study of a smart home.* In *2017 IEEE international conference on pervasive computing and communications workshops (PerCom workshops)*, pages 618–623. IEEE, (2017).

[29] P. Ruan, G. Chen, T. T. A. Dinh, Q. Lin, B. C. Ooi, and M. Zhang, *Fine-grained, secure and efficient data provenance on blockchain systems*, Proceedings of the VLDB Endowment **12**, 975–988 (2019).

[30] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla. *Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability.* In *Proceedings of the 17th IEEE/ACM international symposium on cluster, cloud and grid computing*, pages 468–477. IEEE Press, (2017).

[31] D. K. Tosh, S. Shetty, X. Liang, C. Kamhoua, and L. Njilla. *Consensus protocols for blockchain-based data provenance: Challenges and opportunities.* In *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, pages 469–474. IEEE, (2017).

[32] P. W. Khan, Y.-C. Byun, and N. Park, *IoT-blockchain enabled optimized provenance system for food industry 4.0 using advanced deep learning*, Sensors **20**, 2990 (2020).

[33] D. Ferraiolo, J. Cugini, and D. R. Kuhn. *Role-based access control (RBAC): Features and motivations.* In *Proceedings of 11th annual computer security application conference*, pages 241–48, (1995).

[34] S. Chakraborty, S. Aich, S. J. Seong, and H.-C. Kim. *A Blockchain based Credit Analysis Framework for Efficient Financial Systems.* In *2019 21st International Conference on Advanced Communication Technology (ICACT)*, pages 56–60. IEEE, (2019).

[35] G. Zhang, T. Li, Y. Li, P. Hui, and D. Jin, *Blockchain-based data sharing system for ai-powered network operations*, Journal of Communications and Information Networks **3**, 1–8 (2018).

[36] H. Xu, Q. He, X. Li, B. Jiang, and K. Qin, *BDSS-FA: A Blockchain-Based Data Security Sharing Platform With Fine-Grained Access Control*, IEEE Access **8**, 87552–87561 (2020).

[37] SYMSA1221. *SA54:Fraudulent Comodo SSL certificates*, (2011).

[38] B. Morton. *More Google Fraudulent Certificates*, (2014).

[39] S. Matsumoto and R. M. Reischuk. *IKP: Turning a PKI around with decentralized automated incentives.* In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 410–426. IEEE, (2017).

[40] S. Matsumoto, P. Szalachowski, and A. Perrig. *Deployment challenges in log-based PKI enhancements.* In *Proceedings of the Eighth European Workshop on System Security*, pages 1–7, (2015).

[41] M. Y. Kubilay, M. S. Kiraz, and H. A. Mantar, *Certledger: A new pki model with certificate transparency based on blockchain*, Computers & Security **85**, 333–352 (2019).

[42] G. V. Pinto, J. P. Dias, and H. S. Ferreira. *Blockchain-based PKI for crowdsourced IoT sensor information.* In *International Conference on Soft Computing and Pattern Recognition*, pages 248–257. Springer, (2018).

[43] D. Li, W. Peng, W. Deng, and F. Gai. *A blockchain-based authentication and security mechanism for IoT.* In *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–6. IEEE, (2018).

[44] N. Malik, P. Nanda, A. Arora, X. He, and D. Puthal. *Blockchain based secured identity authentication and expeditious revocation framework for vehicular networks.* In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 674–679. IEEE, (2018).

[45] B. Qin, J. Huang, Q. Wang, X. Luo, B. Liang, and W. Shi, *Cecoin: A decentralized PKI mitigating MitM attacks*, Future Generation Computer Systems (2017).

[46] S. Gan, *An IoT simulator in NS3 and a key-based authentication architecture for IoT devices using blockchain*, Indian Institute of Technology Kanpur (2017).

[47] K. Aberer, A. Datta, and M. Hauswirth, *A decentralized public key infrastructure for customer-to-customer e-commerce*, International Journal of Business Process Integration and Management **1**, 26–33 (2005).

[48] C. Fromknecht, D. Velicanu, and S. Yakoubov, *A Decentralized Public Key Infrastructure with Identity Retention.*, IACR Cryptology ePrint Archive **2014**, 803 (2014).

[49] N. Kumar, R. Iqbal, S. Misra, and J. J. Rodrigues, *An intelligent approach for building a secure decentralized public key infrastructure in VANET*, Journal of Computer and System Sciences **81**, 1042–1058 (2015).

[50] K. Rantos, G. Drosatos, K. Demertzis, C. Ilioudis, and A. Papanikolaou. *Blockchain-based Consents Management for Personal Data Processing in the IoT Ecosystem.* In *ICETE (2)*, pages 738–743, (2018).

[51] A. Ramachandran, D. Kantarcioglu, et al., *Using blockchain and smart contracts for secure data provenance management*, arXiv preprint arXiv:1709.10000 (2017).

[52] H. M. Kim and M. Laskowski, *Toward an ontology-driven blockchain design for supply-chain provenance*, Intelligent Systems in Accounting, Finance and Management **25**, 18–27 (2018).

[53] M. Westerkamp, F. Victor, and A. Kupper, *Tracing manufacturing processes using blockchain-based token compositions*, Digital Communications and Networks (2019).

[54] N. Elisa, L. Yang, F. Chao, and Y. Cao, *A framework of blockchain-based secure and privacy-preserving e-government system*, Wireless Networks , 1–11 (2018).

[55] Q. Wang, B. Qin, J. Hu, and F. Xiao, *Preserving transaction privacy in bitcoin*, Future Generation Computer Systems (2017).

[56] G. Zyskind, O. Nathan, et al. *Decentralizing privacy: Using blockchain to protect personal data.* In *2015 IEEE Security and Privacy Workshops*, pages 180–184. IEEE, (2015).

[57] H. Shafagh, L. Burkhalter, A. Hithnawi, and S. Duquennoy. *Towards blockchain-based auditable storage and sharing of IoT data.* In *Proceedings of the 2017 on Cloud Computing Security Workshop*, pages 45–50. ACM, (2017).

[58] Q. Xia, E. Sifah, A. Smahi, S. Amofa, and X. Zhang, *BBDS: Blockchain-based data sharing for electronic medical records in cloud environments*, Information **8**, 44 (2017).

[59] P. Treleaven, B. Batrinca, et al., *Algorithmic regulation: automating financial compliance monitoring and regulation using AI and blockchain*, Journal of Financial Transformation **45**, 14–21 (2017).

[60] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman. *Medrec: Using blockchain for medical data access and permission management.* In *2016 2nd International Conference on Open and Big Data (OBD)*, pages 25–30. IEEE, (2016).

[61] P. Koshy, D. Koshy, and P. McDaniel. *An analysis of anonymity in bitcoin using p2p network traffic.* In *International Conference on Financial Cryptography and Data Security*, pages 469–485. Springer, (2014).

[62] A. Biryukov, D. Khovratovich, and I. Pustogarov. *Deanonymisation of clients in Bitcoin P2P network.* In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 15–29, (2014).

[63] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage. *A fistful of bitcoins: characterizing payments among men with no names.* In *Proceedings of the 2013 conference on Internet measurement conference*, pages 127–140, (2013).

[64] C. Zhao, *Graph-based forensic investigation of Bitcoin transactions*, (2014).

[65] J. V. Monaco. *Identifying bitcoin users by transaction behavior.* In *Biometric and Surveillance Technology for Human and Activity Identification XII*, volume 9457, page 945704. International Society for Optics and Photonics, (2015).

[66] W. Shao, H. Li, M. Chen, C. Jia, C. Liu, and Z. Wang. *Identifying bitcoin users using deep neural network.* In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 178–192. Springer, (2018).

[67] B. Huang, Z. Liu, J. Chen, A. Liu, Q. Liu, and Q. He, *Behavior pattern clustering in blockchain networks*, Multimedia Tools and Applications **76**, 20099–20110 (2017).

[68] A. Macrina, *The Tor browser and intellectual freedom in the digital age*, Reference & User Services Quarterly **54**, 17 (2015).

[69] S. Nakamoto. *Bitcoin: A peer-to-peer electronic cash system.* Technical report, Manubot, (2019).

[70] F. Astolfi, J. Kroese, and J. Van Oorschot, *I2p-the invisible internet project*, Leiden University Web Technology Report (2015).

[71] Monero. *The Monero Project*, (2014).

[72] D. L. Chaum, *Untraceable electronic mail, return addresses, and digital pseudonyms*, Communications of the ACM **24**, 84–90 (1981).

[73] Dash. *Dash is digital cash*, (2015).

[74] Gmaxwell. *CoinJoin: Bitcoin privacy for the real world*, (2013).

[75] N. van Saberhagen. *CryptoNote v 2.0*, (2013).

[76] 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*, pages 459–474. IEEE, (2014).

[77] J. Poon and T. Dryja. *The bitcoin lightning network: Scalable off-chain instant payments*, (2016).

[78] D. Wang, S. Wu, Z. Lin, L. Wu, X. Yuan, Y. Zhou, H. Wang, and K. Ren. *Towards a first step to understand flash loan and its applications in defi ecosystem.* In *Proceedings of the Ninth International Workshop on Security in Blockchain and Cloud Computing*, pages 23–28, (2021).

[79] N. F. Samreen and M. H. Alalfi. *Reentrancy vulnerability identification in ethereum smart contracts.* In *2020 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, pages 22–29. IEEE, (2020).

[80] P. Praitheeshan, L. Pan, J. Yu, J. Liu, and R. Doss, *Security analysis methods on Ethereum smart contract vulnerabilities: a survey*, arXiv preprint arXiv:1908.08605 (2019).

[81] M. I. Mehar, C. L. Shier, A. Giambattista, E. Gong, G. Fletcher, R. Sanayhie, H. M. Kim, and M. Laskowski, *Understanding a revolutionary and flawed grand experiment in blockchain: the DAO attack*, Journal of Cases on Information Technology (JCIT) **21**, 19–32 (2019).

[82] *EIPs/eip-150.md at master · ethereum/EIPs - GitHub*, (Accessible: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-150.md).

[83] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor. *Making smart contracts smarter.* In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 254–269, (2016).

[84] *A disastrous vulnerability found in smart contracts of BeautyChain (BEC)*, (Accessible: https://medium.com/secbit-media/a-disastrous-vulnerability-found-in-smart-contracts-of-beautychain-bec-dbf24ddbc30e).

[85] *OpenZeppelin - SafeMath Library*, (Accessible: https://github.com/OpenZeppelin/openzeppelin-solidity/blob/master/contracts/math/SafeMath.sol/).

[86] *Parity Wallet Attack Explanation*, (Accessible: https://blog.zeppelin.solutions/on-the-parity-wallet-multisig-hack-405a8c12e8f7).

[87] *Parity Wallet Frozen Attack Explanation*, (Accessible: https://medium.com/cybermiles/i-accidentally-killed-it-and-evaporated-300-million-6b975dc1f76b).

[88] Y. Guo and C. Liang, *Blockchain application and outlook in the banking industry*, Financial Innovation **2**, 24 (2016).

[89] T. Aste, P. Tasca, and T. Di Matteo, *Blockchain technologies: The foreseeable impact on society and industry*, Computer **50**, 18–28 (2017).

[90] J. Kang, Z. Xiong, D. Niyato, P. Wang, D. Ye, and D. I. Kim, *Incentivizing consensus propagation in proof-of-stake based consortium blockchain networks*, IEEE Wireless Communications Letters **8**, 157–160 (2018).

[91] B. Curran, *What is Proof of Authority consensus*, Staking Your Identity on The (2018).

[92] *R3 Corda Website.*

[93] F. Saleh, *Blockchain without waste: Proof-of-stake*, Available at SSRN 3183935 (2020).

[94] B. S. Fogel, A. J. Kazmer, and S. B. Littlehale. *Automated data integrity auditing system*, (2003). US Patent 6,542,905.

[95] *Oreilly Website*, (Accessible: https://www.oreilly.com/library/view/advances-in-financial/9781119482086/c02.xhtml).

[96] *Truffle Suite*, (Accessible: https://www.trufflesuite.com/docs/truffle/overview).

[97] *Rinkeby: Ethereum Testnet*, (Accessible: https://www.rinkeby.io).

[98] B. CURRAN. *What is Proof of Authority Consensus? Staking Your Identity on The Blockchain*, (2018).

[99] R. Bhanot and R. Hans, *A review and comparative analysis of various encryption algorithms*, International Journal of Security and Its Applications **9**, 289–306 (2015).

[100] A. Shahnaz, U. Qamar, and A. Khalid, *Using blockchain for electronic health records*, IEEE Access **7**, 147782–147795 (2019).

[101] A. D. Dwivedi, G. Srivastava, S. Dhar, and R. Singh, *A decentralized privacy-preserving healthcare blockchain for IoT*, Sensors **19**, 326 (2019).

[102] H. Tian, J. He, and Y. Ding, *Medical data management on blockchain with privacy*, Journal of medical systems **43**, 1–6 (2019).

[103] S. Fosso Wamba, J. R. Kala Kamdjoug, R. Epie Bawack, and J. G. Keogh, *Bitcoin, blockchain and fintech: a systematic review and case studies in the supply chain*, Production Planning & Control **31**, 115–142 (2020).