HIGH-PERFORMANCE PACKING AND SEARCHING FOR BLOCKCHAIN-BASED BIG DATA SHARING

SHAN JIANG

PhD

The Hong Kong Polytechnic University

2021

The Hong Kong Polytechnic University
Department of Computing

High-performance Packing and Searching for Blockchain-based Big Data Sharing

Shan Jiang

A thesis submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy
January 2021

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

Shan Jiang (Name of Student)

Abstract

Big data has been showing its great value in revolutionizing various industries. Generally, the big data possessed by different stakeholders forms isolated data islands, which limits their values because data cooperation can make the total value greater than the mere sum of the parts. Big data sharing is the key for the transition from data islands to data ecosystems and maximizing the value of big data. Recently, blockchain technology has been attracting intensive attention from both the industries and academic. Because of the prominent features of transparency, decentralization, and immutability, blockchain is considered as a promising solution for big data sharing. In this thesis, we study blockchain-big data sharing in terms of the basic concepts, challenging issues, and high-performance solutions.

In particular, we conduct a comprehensive survey of big data sharing and present the system architecture and layered research framework of blockchain-based big data sharing. Inside the research framework, we tackle several challenges, ranging from transaction fairness, privacy preservation during data search, and anonymity of the data users, in blockchain-based big data sharing by proposing high-performance solutions. The algorithms and mechanisms are evaluated in various applications such as IoT data management and e-voting. The experimental results have indicated the high performance of our solutions. We believe this thesis can serve as a solid step towards real-world applications of blockchain-based big data sharing.

Publications Arising from the Thesis

1. Shan Jiang, Jiannong Cao, Hanqing Wu, Yanni Yang, "Fairness-based Packing of Industrial IoT Data in Permissioned Blockchains", in *IEEE Transactions on Industrial Informatics* (2020).

2. Shan Jiang, Jiannong Cao, Juncen Zhu, Yinfeng Cao, "PolyChain: a Generic Blockchain as a Service Platform", accepted by *Springer BlockSys 2021*.

3. Shan Jiang, Jiannong Cao, Julie A. McCann, Yanni Yang, Yang Liu, Xiaoqing Wang, Yuming Deng, "Privacy-preserving and Efficient Multi-keyword Search Over Encrypted Data on Blockchain", in *IEEE Blockchain 2019* (Oral Presentation).

4. Shan Jiang, Jiannong Cao, Hanqing Wu, Yanni Yang, Mingyu Ma, Jianfei He, "BlocHIE: a BLOCkchain-based platform for Healthcare Information Exchange", in *IEEE SMARTCOMP 2018*.

5. Shan Jiang, Junbin Liang, Jiannong Cao, Jia Wang, Jinlin Chen, and Zhixuan Liang, "Decentralized Algorithm for Repeating Pattern Formation by Multiple Robots", in *IEEE ICPADS 2019* (Oral Presentation).

6. Shan Jiang, Jiannong Cao, Jia Wang, Milos Stojmenovic, Julien Bourgeois, "Uniform Circle Formation by Asynchronous Robots: A Fully-Distributed Approach", in *IEEE ICCCN 2017* (Oral Presentation).

7. Shan Jiang, Jiannong Cao, Yang Liu, Jinlin Chen, Xuefeng Liu, "Programming Large-Scale Multi-Robot System with Timing Constraints", in *IEEE ICCCN 2016* (Oral Presentation).

8. Shan Jiang, Junbin Liang, Jiannong Cao, Rui Liu, "An ensemble-level programming model with real-time support for multi-robot systems", in *IEEE PerCom Workshops 2016* (Demo).

9. Xiulong Liu, Jiuwu Zhang, Shan Jiang, Yanni Yang, Keqiu Li, Jiannong Cao, Jiangchuan Liu, "Accurate Localization of Tagged Objects Using Mobile RFID-augmented Robots", in *IEEE Transactions on Mobile Computing* (2019).

10. Yuvraj Sahni, Jiannong Cao, Shan Jiang, "Middleware for Multi-Robot System", a book chapter in *Mission-Oriented Sensor Networks and Systems: Art and Science*, Springer (2019).

11. Hanqing Wu, Jiannong Cao, Yanni Yang, Cheung Leong Tung, Shan Jiang, Bin Tang, Yang Liu, Xiaoqing Wang, Yuming Deng, "Data Management in Supply Chain Using Blockchain: Challenges and a Case Study", in *IEEE ICCCN 2019*.

12. Jia Wang, Jiannong Cao, Milos Stojmenovic, Miao Zhao, Jinlin Chen, Shan Jiang, "Pattern-RL: Multi-robot Cooperative Pattern Formation via Deep Reinforcement Learning", in *IEEE ICMLA 2019*.

13. Xiulong Liu, Jiannong Cao, Yanni Yang, Shan Jiang, "CPS-Based Smart Warehouse for Industry 4.0: A Survey of the Underlying Technologies", in *Computers* (2018).

14. Hanqing Wu, Jiannong Cao, Shan Jiang, Ruosong Yang, Yanni Yang, Jianfei He, "TSAR: a fully-distributed Trustless data ShARing platform", in *IEEE SMARTCOMP 2018* (Workshop).

15. Jia Wang, Jiannong Cao, Shan Jiang, "Fault-Tolerant Pattern Formation by Multiple Robots: A Learning Approach", in *IEEE SRDS 2017* (Ph.D. Forum).

Acknowledgments

When the time elapsed slowly in The Hong Kong Polytechnic University (PolyU), the twenty-six-year-old me stands right here and watch the way behind. My eyes are occupied with the twenty-year-old me. At that time, I just entered Hong Kong as a little boy, and now, I am a husband and very likely to be a father in the near future. During the six-year RA and Ph.D. life, I have learned a lot and become stronger. My research is focused on algorithms for distributed systems such as multi-robot system, blockchain, and RFID. I believe such a experience will never fade in my memory. Besides the hard work, I do not think I can be capable of reaching this milestone of my study without the assistance and supports from my supervisor, family, friends, and colleague.

First and foremost, I would like to express my sincere thanks to my supervisor, Prof. Jiannong Cao, for his guidance and advice in both my research and life. During my Ph.D. study, my research direction has shifted from multi-robot systems [75][81][77][82][161][107][112][162][140] to blockchain [79][76][78][80][171][172], but they are all about algorithms for distributed systems. It was he that encouraged me to continue the Ph.D. study in the toughest time. It was he that taught me of the skills necessary for researchers, e.g., listening, reading, writing, presenting, and teaching. It was he that let me know the high-quality and high-impact research. I believe his guidance and advice will accompany me and continue to benefit the rest of my life.

Besides, It is my honor to spend years with students and staffs in Internet and Mobile

Computing Laboratory and Department of Computing, PolyU. They are more like my family members rather than colleagues. I would like to express my thanks to Dr. Junbin Liang, Dr. Xuefeng Liang, and Dr. Milos Stojmenovic for their guidance in the past. It was they that guided me when I knew nearly nothing about research. Meanwhile, Christy, Carmen, Jolie, Anna, and Esther have helped me a lot for the complicated administrative tasks. My special thanks go to my friends, e.g., Dr. Xiulong Liu, Mr. Yu Yang, Ms. Yanni Yang, Ms. Juncen Zhu, Dr. Jiaxing Shen, Mr. Ruosong Yang, Mr. Yang Liu, Mr. Hanqing Wu, Dr. Wengen Li, Mr. Zhiyuan Wen, Mr. Zhuo Li, Ms. Jia Wang, Dr. Fuliang Li, Dr. Lei Yang, Mr. Zhixuan Liang, Mr. Jinlin Chen, Mr. Mingjin Zhang, Mr. Qianyi Chen, Dr. Yanwen Wang, and Ms. Jiating Zhu. Sometimes we get together and sometimes separate, but I really cherish and will never forget the time with them when hiking, eating, drinking, studying, playing, and so on.

Last but not least, I would like to thank my family and friends, e.g., Ms. Shuang Chen, Mr. Zaodao Jiang, Ms. Aiqin Cheng, Dr. Shiyan Jiang, Mr. Hao Cheng, Dr. Haofeng Li, Mr. Qiaotian Lu, Mr. Zhengjie Huang, and Ms. Bing Zhao. It does not matter they know my research or not, but their understanding, support, accomplishment, and love were the power to relieve my mental burden. Without them, I am a lesser man.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background & Motivation

In recent years, the development of Internet of Things (IoT), social media, etc. has brought an increasing amount of data generated, collected, and processed everyday. The high-volume, high-velocity, and high-diversity data is generally the so-called big data. Big data is indeed a revolution of the information technology with tangible benefits of cost reduction, efficiency improvement, and intelligent decision making in industries, commerce, and social good. For example, 35% of Amazon.com's revenue is generated by its big data-driven recommendation engine.

Generally speaking, the data owners, e.g., enterprises and hospitals, are not willing to share the big data because of the great value. However, there are certain critical scenarios that demand big data from different stakeholders with conflict of interests. For example, accurate disease diagnosis requires a large amount of hospitalized cases all around the worlds while few hospitals can hardly make it. To this end, the concept of big data sharing arises to enable different stakeholders so that they can find, access, and use big data from each other.

There are many big data sharing platforms, e.g., SEEK[1], Amazon Web Services Data Exchange[2], and HKSTP Data Studio[3]. The traditional platforms can be classified into two categories, i.e., data hosting center (DHC) and data aggregation center (DAC). A DHC collects the original big data from the data owners and find possible data users to share the big data and get rewards. Since big data is easy to be replicated, DHCs are often used for open big data, that is, the big data shared on DHCs are purposely opened to the public. In terms of private data, DHC suffers a lot from the privacy issue. In terms of DAC, it collects the descriptions of the big data from the data owners, provides a platform for the potential users to search the big data, and facilitates big data sharing between the data owners and data users. Because the DAC cannot guarantee whether the claims from the data owners are correct, DAC suffers a lot from the authenticity issue.

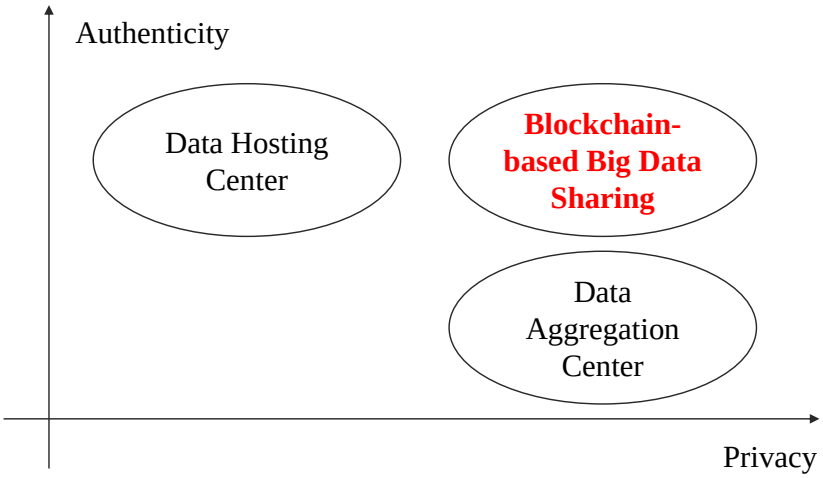

Figure 1.1: Motivation of blockchain for big data sharing

In recent years, blockchain technology has been attracting extensive attention from both industry and academia, since it enables trustless data storage with auditability. As shown in Figure 1.1, blockchain can address the issues of privacy and authenticity

[1]SEEK: Finding, Sharing, and Exchanging Data, Models, Simulations and Processes in Science
[2]AWS Data Exchange: Easily Find and Subscribe to Third-party Data in the Cloud
[3]Data Studio at Hong Kong Science Park

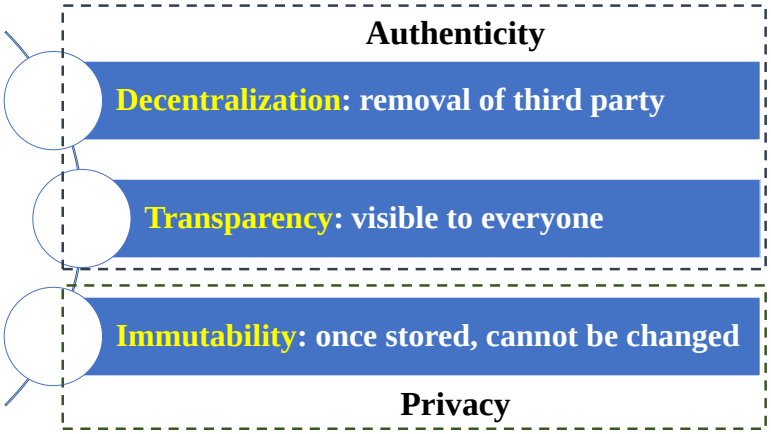

Figure 1.2: Distinctive features of blockchain

in big data sharing because of the distinctive features of transparency, immutability, and decentralization as shown in Figure 1.2. On the one hand, the big data will be hosted by the data owners themselves rather than a trustworthy third party, in which the privacy will not be violated. On the other hand, the data is verified and confirmed by the nodes in blockchain network, which guarantees the data authenticity. To this end, blockchain can be served as a promising solution to big data sharing.

Here, we briefly introduce blockchain and its underlying consensus protocol. Traditionally, a blockchain is a chain of blocks linked and secured using cryptography. As shown in Figure 1.3, each block contains four components, namely block size, transaction counter, block header, and transactions. The block size, transaction counter, and transactions are the number of bytes of the block, the number of transactions, and all the transactions respectively. The block header contains six fields, namely version, previous block hash, timestamp, difficulty target, nonce, and Merkle root. The version is a version number to track the consensus protocol upgrades, the timestamp is the approximate create time of the block, while the difficulty level and nonce are used for proof-of-work consensus protocol. The Merkle root refers to the hash of all the hashes of all the transactions. The previous block hash is a reference to the hash of the previous block along the chain. The hash value of a block, which is the

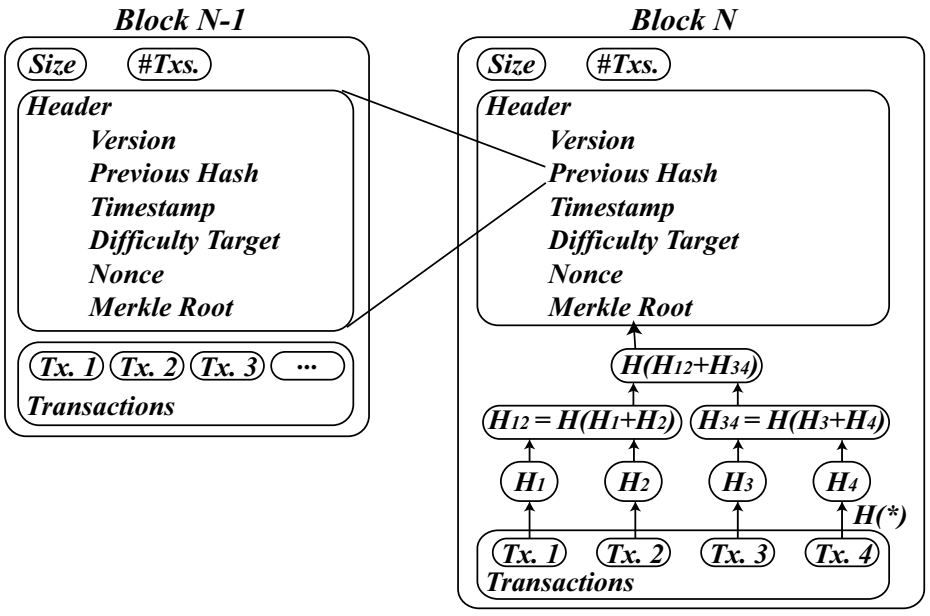

Figure 1.3: Structure of a traditional blockchain

primary identifier of a block, is made by hashing the block header twice through the SHA-256 hash function.

A blockchain is replicated among the members of a network, in which each member holds a replication of the committed transactions and a pool of the submitted but uncommitted transactions. Each member is responsible for packing the transactions from the pool to the blocks to make them committed. In order to make the blockchain remain functional, the members need to agree on a certain state of the blockchain. This procedure is accomplished by the underlying distributed consensus algorithm.

As shown in Figure 1.4, it requires three steps for a transaction to be committed. At first, the user has some *raw* transactions (the red ones) and want to publish them. Then, the user submits the raw transactions to the blockchain network. Each member in the blockchain network receives the transactions from the user and maintains a transaction pool. The transactions in pool are called *submitted* transactions (the yellow ones). At this time, the members are supposed to make consensus on the way to maintain the blockchain based on the transaction pool. The consensus consists of

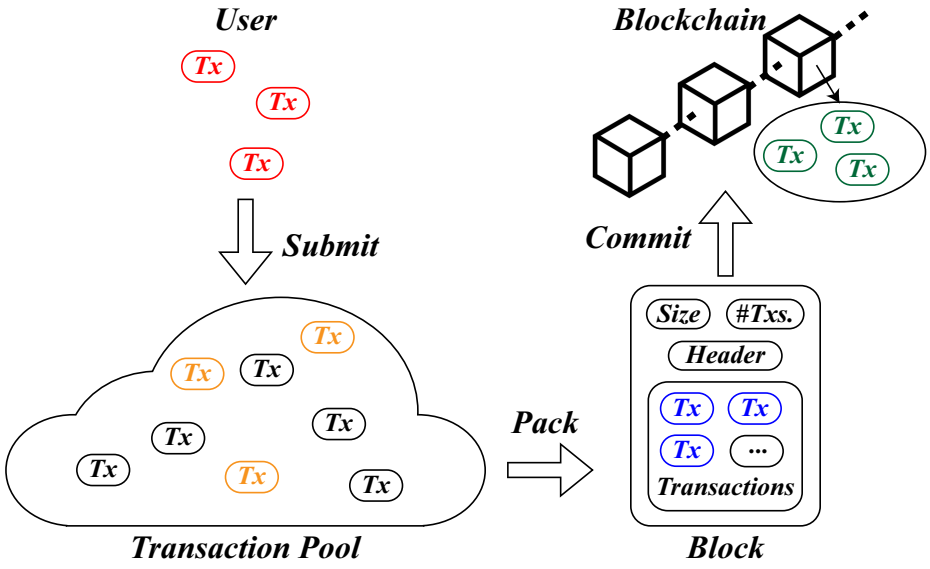

Figure 1.4: The procedure for committing transactions

two steps, namely packing and committing. At the packing stage, each member selects some submitted transactions and puts them into a block. The transactions that are packed into a block but not yet committed are called *packed* transactions (the blue ones), and the block containing packed transactions are called uncommitted block. Finally, at the step of committing, the members make efforts to get the uncommitted blocks validated and committed. If a transaction is in a validated block, it is said to be *committed* (the green ones).

1.2 System Architecture

Figure 1.5 presents the system architecture of blockchain-based big data sharing. There are three entities in the system, i.e., data sharer, data sharee, and blockchain nodes explained as follows:

- The data sharer is the data owner who wants to share the big data.
- The data sharee aims to use the shared big data.

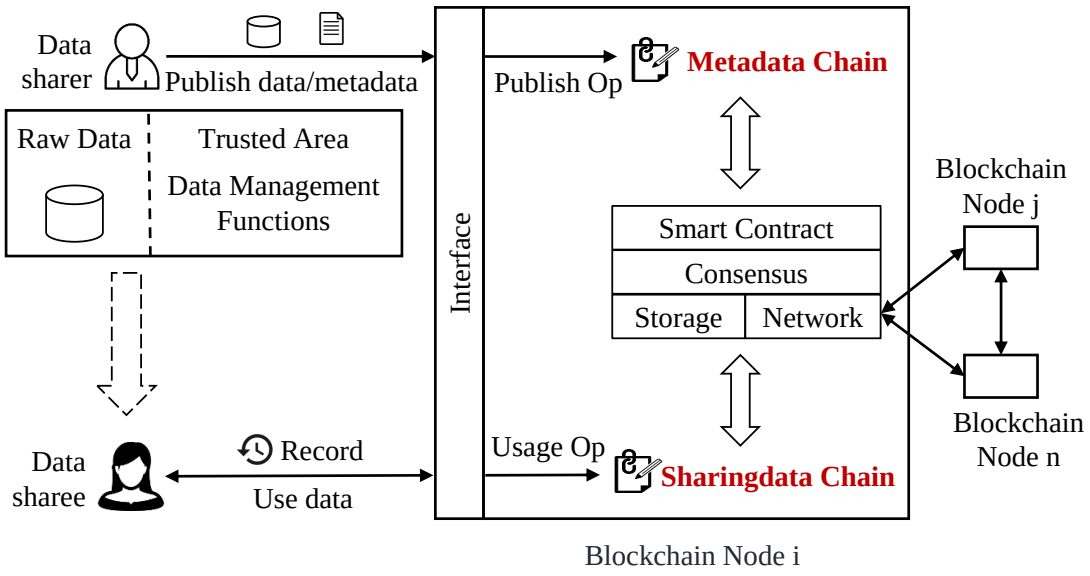

Figure 1.5: System architecture of blockchain-based big data sharing

- The blockchain nodes are the service provider who provides the big data sharing services for the data sharers and sharees.

The data sharers publish the data or metadata using the interface provided by the blockchain nodes. Meanwhile, it stores the raw data locally and manage them in a trusted area, e.g., trusted execution environments. The data sharees use the big data from the data sharers and generate sharing records. The data publishing and usage operations are stored in two blockchains, i.e., metadata chain and sharingdata chain, respectively. The blockchain nodes maintain the two blockchains and interact with each other. The metadata chain and sharingdata chain share the same layered blockchain infrastructure consisting of smart contract layer, consensus layer, storage layer, and network layer. However, metadata chain and sharingdata chain have different requirements on access control, performance, anonymity, etc. and they can will be optimized based on the requirements.

1.3 Research Framework

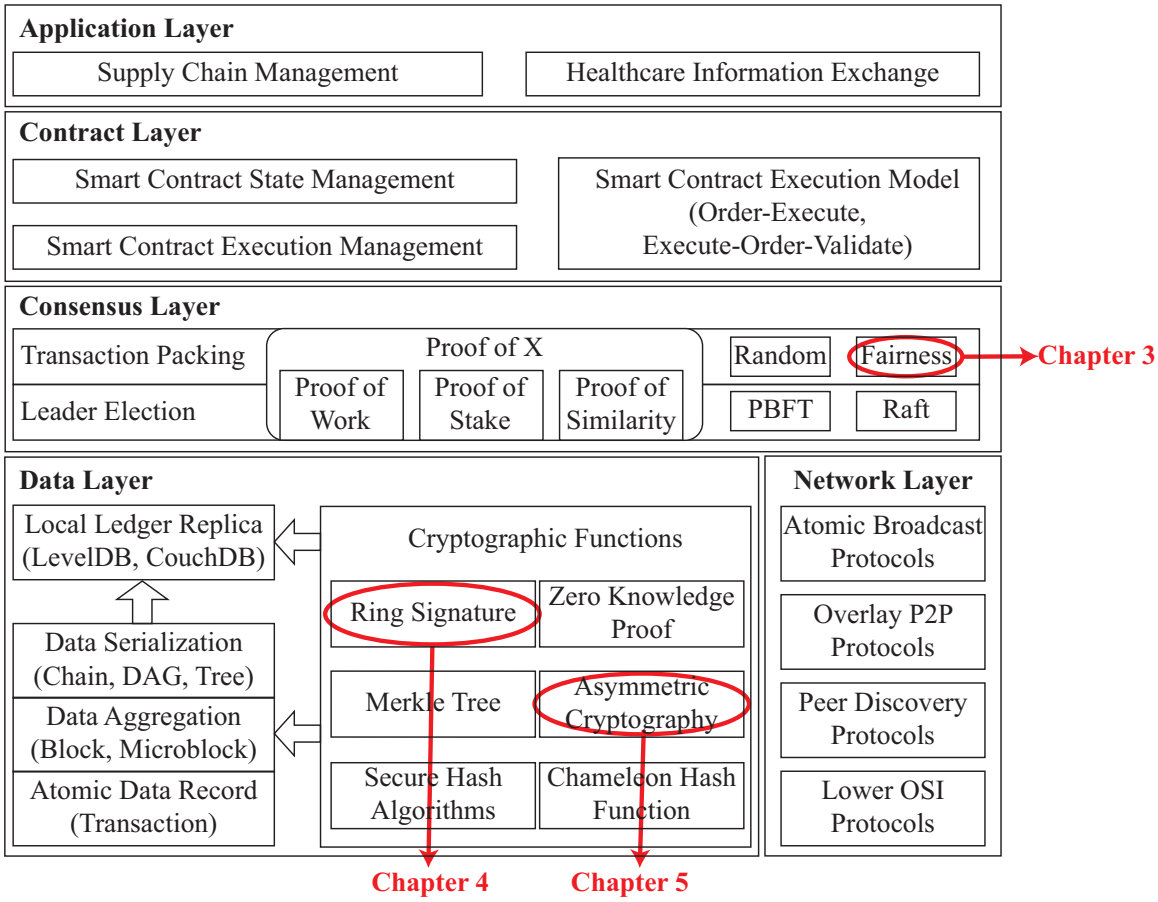

Figure 1.6: Research framework of blockchain-based big data sharing

In this thesis, we aim to identify and address the challenging issues in blockchain-based big data sharing. The research framework is shown in Figure 1.6. The research framework consists of six layers as follows:

- *Application layer*: it provides the main applications of blockchain-based big data sharing, e.g., supply chain management and healthcare information exchange. In chapter 2, we presents a comprehensive survey of big data sharing.
- *Contract layer*: it allows users to write, deploy, and use smart contracts. The

contract layer should support contract state management and the execution environment. Note that contract execution models can be different, e.g., order-execute model and execute-order-validate model.

- *Consensus layer*: it outputs an ordered list of transactions agreed by all the nodes in the blockchain network given a set of transactions as input. The procedure of consensus is divided into two phases, i.e., leader election and transaction packing. The traditional "proof of x" consensus mechanisms, e.g., proof of work and proof of stake, integrates the two phases. The leader election algorithms include PBFT and Raft while the transaction packing is normally based on random strategy. In chapter 3, we present a fairness-based transaction packing algorithm.

- *Data layer*: it provides support of cryptographic operations and reliable storage of blockchain data. The blockchain is formed through atomic data record (transaction), data aggregation (e.g., microblock and block), and data serialization (e.g., chain, DAG, and tree), and replicated locally in databases (e.g., LevelDB and CouchDB). Various cryptographic functions should be provided to support the construction of a secure blockchain such as ring signature, zero knowledge proof, Merkle tree, asymmetric cryptography, secure hash algorithms, and chameleon hash functions. In chapter 5, we present a dynamic ring signature algorithm to provide provable anonymity; in chapter 4, we present a privacy-preserving and efficient multi-keyword searchable encryption scheme.

- *Network layer*: it provides the basic network operations and functions to interact with users and other blockchain nodes. The network functions include lower OSI protocols, peer discovery protocols, overlay P2P protocols, and atomic broadcast protocols.

1.4 Thesis Organization

The rest of this thesis is organized as follows:

- In chapter 2, we present a comprehensive survey of big data sharing. Previous research and surveys about big data sharing focus on one or two technical solutions or application domains while we give the first full view of big data sharing. In particular, we articulate the definition of big data sharing and distinguish the concept with data sharing, data exchange, and big data trading. Then, the general procedures, benefits, and requirements of big data sharing are clarified. Afterward, we study the existing big data sharing platforms, and categorize them into data hosting centers and data aggregation centers. Moreover, the challenging issues for developing big data sharing solutions are demonstrated together with explanations of the potential solutions. Finally, we describe two popular big data sharing applications in recent years, i.e., big data sharing for healthcare and big data trading, and their major challenges.

- In chapter 3, we propose a fairness-based transaction packing algorithm to improve the quality of data sharing service. In existing permissioned blockchains, transactions are arbitrarily packed into blocks without consideration of their waiting times. Hence, there will be a high deviation of the transaction response times, which is known as lack of fairness. Unfair permissioned blockchain decreases the quality of experience. Moreover, some transactions can get timeouts if not responded for a long time. To address the issue, we propose FairPack, the first fairness-based transaction packing algorithm for permissioned blockchain. In particular, we gain the insight that fairness is positively related to the sum of waiting times of the selected transactions through theoretical analysis. Based on the insight, we transform the fairness problem into the subset sum problem, which is to find a valid subset from a given set with subset sum as large as possible. However, it is time-consuming to solve the problem

in a brute-force approach because there is an exponential number of subsets for a given set. To this end, we propose a heuristic and a min-heap-based optimal algorithm for different parameter settings.

- In chapter 4, we propose a multi-keyword search algorithm to enhance the privacy and efficiency during data usage between the data sharers and sharees. Recent research has demonstrated searchable blockchains that not only provide reliable search over encrypted distributed storage systems but ensure privacy is preserved. Yet, current solutions focus on single-keyword search over encrypted data on the blockchain. To extend such approaches to multi-keyword scenarios, they essentially perform a single-keyword search multiple times and take the intersection of the results. However, such extensions suffer from privacy and efficiency issues. We propose a bloom filter-enabled multi-keyword search protocol with enhanced efficiency as well as privacy preservation. In the protocol, a low-frequency keyword selected by a bloom filter will be used to filter the database when performing a multi-keyword search operation. Because the keyword is of low frequency, the majority of the data will be excluded from the result, which reduces the computational cost significantly. Moreover, we propose to use pseudorandom tags to facilitate completing a search operation in only one round. In this way, no intermediate results are generated, and the privacy is preserved.

- In chapter 5, we propose dynamic ring signature which can provide provable anonymity during data sharing. We study the anonymity problem in the scenario of blockchain-based e-voting systems (BEVs). Although blockchain brings distinctive features of auditability and immutability to e-voting systems, the public ballot information makes it challenging to preserve the voters' anonymity. There are three major approaches to achieve anonymous BEVs. First, zero-knowledge proof and blind signature are impractical due to the high computational complexity and unverifiable ballot information. Second, linkable ring

signature, in which each voter adds mixins so that the public can verify the ballot but cannot identify the exact voter, cannot preserve anonymity considering a set of ballots. Dynamic ring signature is a novel anonymous mechanism achieving provable anonymity in BEVs. The key idea is to select mixins intentionally rather than randomly as in traditional ring signature schemes. In particular, we present a maximum flow-based algorithm with rigorous proof to validate the anonymity for a set of ballots, which is not considered ever before. Moreover, a time-efficient heuristic algorithm for mixin selection is proposed based on the intuition to include voters with different choices. Theoretical analysis and experimental results indicate the superiority of dynamic ring signature in terms of fewer mixins and less time overhead than the traditional approach.

- Chapter 6 concludes this thesis with summarization of the main contents and our ideas in terms of the future of blockchain-based big data sharing.

Chapter 2

Big Data Sharing: a Comprehensive Survey

Despite the significance of big data sharing as introduced in previous chapter, there are a lot concerns and challenges hindering its process of maturity. One of the most critical issues lies in the privacy concerns. A lot of big data, e.g., healthcare records and social media posts, contains sensitive information. Direct transfer of the sensitive big data may violate the terms of usage and even break the law. Besides the privacy concerns, there are still many other issues, such as the data heterogeneity, value assessment, and pricing model, that are challenging while necessary to be tackled.

In the literature, there are a lot of solutions towards addressing the challenges while lacking a comprehensive survey. Indeed, there are a bunch of survey papers related to but are not exactly big data sharing. In particular, some surveys explain the concept and the underlying techniques of big data in a very broad sense [29][88][126] or are limited to a special aspect or applications of big data, e.g., security and privacy [177], platforms [147], machine learning [132], computational health informatics [50], Internet of Things [55], multimedia [130]. Some other papers focus on sharing of a particular type of data, i.e., scholarly data [173][154][67] and healthcare data [133][11]

because they are very important or a concrete technique, e.g., proxy re-encryption [131]. The most relevant paper is a survey about big data trading [104]. However, big data trading is only a subset of big data sharing and the survey focuses on the aspects of pricing models, issues of platform design, and digital copyright protection schemes, which is really limited in terms of the scope. In summary, the existing research and surveys about big data sharing only focus on one or two technical solutions or applications domains, which are not systematic enough.

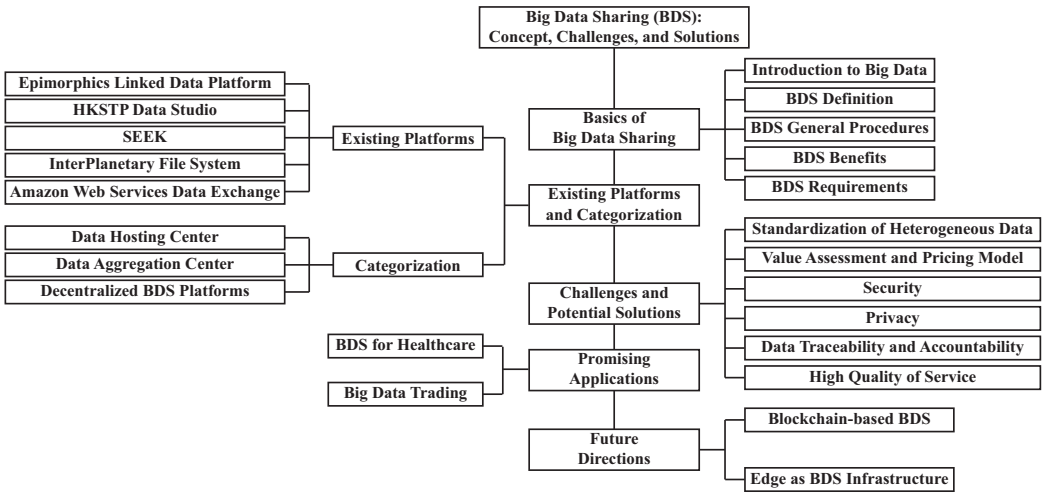

Figure 2.1: Survey structure

To the best of our knowledge, this is the first comprehensive survey about big data sharing. The structure of this survey is shown in Figure 2.1. We articulate the definition, general procedures, benefits, and requirements of big data sharing in Section 2.1. A special contribution is that we collect the concepts that are similar to big data sharing, and demonstrate the differences in detail. Moreover, we summarize the existing big data sharing platforms with description of the representative ones in Section 2.2. The platforms are categorized into data hosting center and data aggregation center based on the system architecture. In addition, the challenging issues of developing big data sharing solutions are identified with explanation of the possible solutions in Section 2.3. Finally, we list two representative application domains of big data sharing in Section 2.4.

2.1 Basics of Big Data Sharing

In this section, we introduce the basics of big data sharing. First, we explain what big data is and why big data matters. Second, we turn to our focus and give a formal definition for big data sharing. Third, the general procedures of big data sharing are demonstrated. Forth, the benefits of big data sharing are illustrated from the perspectives of both data sharers and the public good. Finally, we explain the general requirements of big data sharing.

2.1.1 Introduction to Big Data

Before introducing what big data sharing is, we should figure out what big data is. The term of big data has been used since late 1990s and got popular quickly since its appearance in *Communications of the ACM* in 2009 [73]. As a matter of fact, big data has been defined as early as 2001. At that time, big data is generally defined according to its *volume*, that is, big data refers to a large volume of data with sizes beyond the ability of commonly used software tools to capture, curate, manage, and process within a tolerable elapsed time [148]. Later on, various "Vs" are added to describe big data, among which *variety* and *velocity* are broadly accepted. Particularly, the variety of big data refers to the various modalities of data while the velocity of data is the speed at which the data is generated and processed. To this end, the "3Vs" model is formed to describe big data and such an model is adopted by Gartner and many other enterprises such as IBM and Microsoft for a long time.

Figure 2.2 illustrates the primary characteristics of big data in three dimensions explained as follows:

- *Volume*. The process of digitization in the modern society makes it plain to accumulate a large *volume* of data using numerous information-sensing Internet of Things (IoT) devices. The volume of big data increases from megabytes

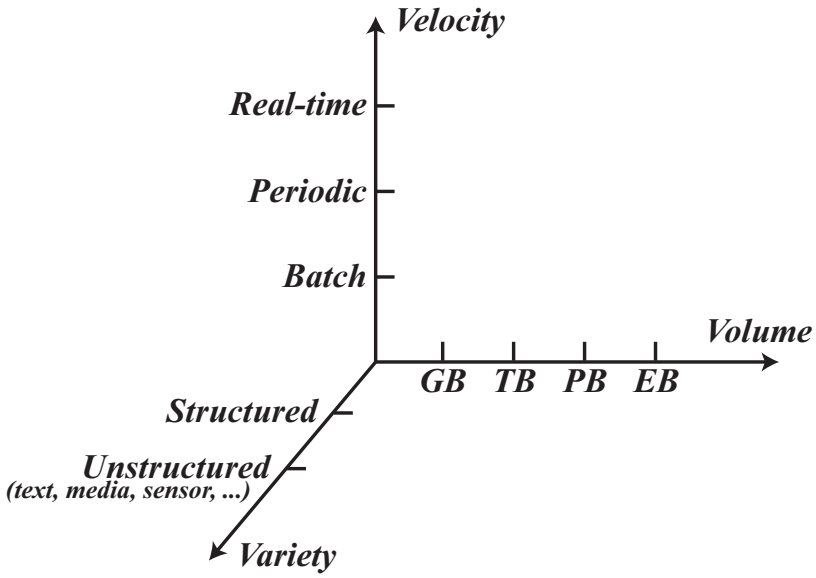

Figure 2.2: Primary characteristics of big data

(MB), gigabytes (GB), to exabytes (EB) and beyond.

- *Velocity.* The increasing generation speed of big data requires advanced processing technologies. The most common BDA techniques is batch processing, in which the data is assumed to be ready and invariant in size. As the data velocity increases, more sophisticated BDA techniques, e.g., periodic processing and real-time processing, are demanded.

- *Variety.* In terms of variety, the data can be either structured or unstructured. On the one hand, the structured data conforms to some well-defined data models or scheme, e.g., length-delineated text strings and range-restricted integers, and usually resides in relational databases. On the other hand, unstructured data is essentially everything else, which includes text, photos, audio, video, sensor data, etc., and non-relational databases are used for storage in general.

Besides volume, variety, and velocity, there is another "V", i.e., *value*, which has been attracting attention from both industry and academia recently. An example of data value comes from Google LLC during the swine flu pandemic in 2009. At that

time, Google obtained timely information by analyzing the big data coming from its search engine, and provided even more valuable information than that provided by the disease prevention centers. Google found that during the spreading of influenza, entries frequently sought at its search engines would be different from those at ordinary times, and the use frequencies of the entries were correlated to the influenza spreading in both time and location. Google found 45 search entry groups that were closely relevant to the outbreak of influenza and incorporated them in specific mathematic models to forecast the spreading of influenza and even to predict places where influenza spread from. The research results have been published in Nature [58]. Besides the social impact as value, big data is also potential to create earnings and reduce costings for enterprises, e.g., to improve the recommendation results for e-commerce platforms, and to optimize the pricing strategies for airline companies.

To summarize, we use a sentence from an International Data Corporation report to conclude the introduction of big data as follows: *big data* technologies describe a new generation of technologies and architectures, designed to economically extract *value* from very large *volumes* of a wide *variety* of data, by enabling the high-*velocity* capture, discovery, and/or analysis [53].

2.1.2 Definition of Big Data Sharing

Although big data can create remarkable values for the society and enterprises, there are still many challenges hindering its development, e.g., inefficient representation of data, inferior analytical mechanisms, and lack of data cooperation. In this chapter, we focus on data cooperation, or *big data sharing* from a more general perspective.

Big data sharing refers to the act of the data sharers to share big data so that the sharees can find, access, and use in the agreed ways.

There are several other concepts, i.e., *data sharing*, *data exchange*, and *big data trading*, that are similar or related to big data sharing. In fact, they share some

similarities, however are different, to some extent. In the following, we explain their concepts and differences with big data sharing one by one.

Table 2.1: Comparison among big data sharing, data sharing, data exchange, and big data trading

Concept	Big data sharing	Data sharing	Data exchange	Big data trading
Data	Unrestricted	Scholarly data for most of the time	Unrestricted	Unrestricted
Reward	Unrestricted	No reward for most of the time	Right of using data	Unrestricted
Commerciality	Unrestricted	Non-commercial for most of the time	Unrestricted	Commercial

The term data sharing often refers to the practice of making data used for scholarly research available to other investigators [154][51]. The data in the term data sharing is more about scholarly data rather than general data as in big data sharing. Moreover, big data sharing requires the data to be shared in a secure way while the purpose of data sharing is more about making the data open. Indeed, the high volume, high variety, and high value of big data demands high performance, unified data representation, and advanced security for big data sharing compared to data sharing.

There are two kinds of definitions for data exchange. Some researchers define data exchange as of process of taking data structured under a source schema and transforming it into data structured under a target schema, so that the target data is an accurate representation of the source data [3]. Such a definition is different from big data sharing substantially, as a result, is not considered in this chapter.

Some others consider data exchange as exchange of data, that is, the act of giving data owned by one party to another and receiving data owned by the other party in return [27]. The acts in the definition of data exchange and big data sharing differ a lot. In particular, big data sharing focuses on finding, accessing, and using data, while data exchange targets on exchange of the right of using data. Actually, the word *exchange* emphasizes that the thing transferred between two parties should be of the same kind, and the thing refers to the right of using data in data exchange.

Big data trading refers to the action or activity of buying or selling big data [104]. It has been a hot topic in recent years because many enterprises and commercial organizations have found that it is of great potential to make money by selling the collected data of high value. At the meantime, many other enterprises or organizations need the data for certain purposes, e.g., conducting research in universities and improving quality of products for companies. We can see that the concept of big data trading is a subset of the one of big data sharing. Particularly, big data trading is restricted to be commercial while big data sharing is unrestricted.

The differences among big data sharing, data sharing, data exchange, and big data trading in terms of the involved data types, the reward to get data, and the commerciality are summarized in Table 2.1. First, we can see that big data sharing is different from data sharing from all the three perspectives. Second, the main difference between big data sharing and data exchange lies in the reward to get data, which is unrestricted and the right of using data, respectively. Finally, the concept of big data sharing contains big data trading because big data trading must be commercial while the commerciality of big data sharing is unrestricted.

2.1.3 General Procedures of Big Data Sharing

After illustration of what big data sharing is, we explain the general procedures of big data sharing in this section. Generally speaking, there are three steps for big data sharing, i.e., *data publishing*, *data search*, and *data sharing*, as follows:

- *Data publishing*: the data owners prepare and issue the the big data they owned for public access. The purpose of data publishing is to let the public be aware that there is a piece of big data and its brief information. Data publishing supplies commodities for a big data sharing platform. Note that it is not necessary for the data owners to make the original big data public because the data owners may want to enforce access control upon the big data.

- *Data search*: the data users query the big data sharing platform for big data possibly with some constraints, and the big data sharing platform responds. The purpose of data search is to make it convenient for the data users to find the big data they want. Data search is significant because data users are the customers for a big data sharing platform while providing content services for the customers is crucial.

- *Data sharing*: the data users request big data in the big data sharing platform and the corresponding data owners accept or reject the request. Data sharing is the last step for a big data sharing platform. If the data owner accepts the request, then it is called successful data sharing; otherwise, it is failed data sharing. After successful data sharing, the data owner will become the data sharer while the data user will become the data sharee.

2.1.4 Benefits of Big Data Sharing

There are a lot of benefits for big data sharing. On one hand, the sharees, i.e., the parties to receive the shared data, can benefit for sure because they can use the data for their purposes. Moreover, the sharers, i.e., the parities to share data, can also benefit because they will gain more visibility from the public and possibly gain monetary rewards. Finally, there are a lot of public good during the sharing activities. Figure 2.3 illustrates the benefits of big data sharing from the perspectives of sharers and public good. The benefits for the sharees are not considered here because of the strong dependence on the purposes of the sharees.

On one hand, the benefits for the data sharers are summarized as follows:

- *If the sharers are scholars and the shared data is scholarly data, then big data sharing can increase the visibility of the research work and is potential to enhance academic reputation.* The scholarly data to be shared may include free

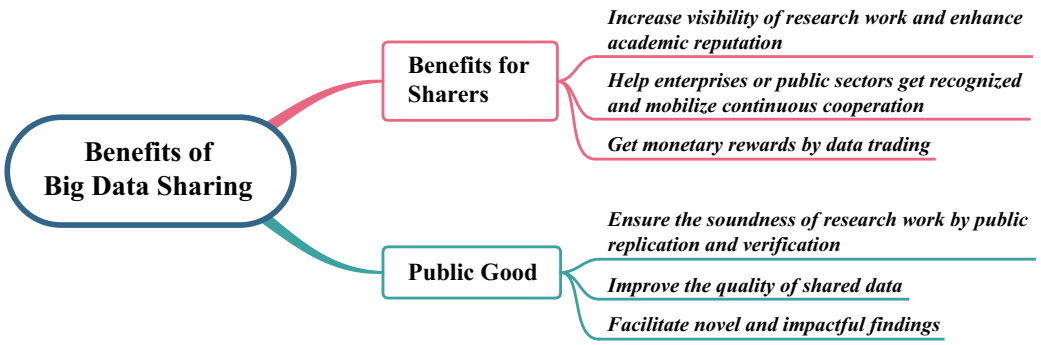

Figure 2.3: Benefits of big data sharing

the download of full text, the source code and tools for experiments, and the collected data for performance evaluation. There is no doubt that the researchers are more willing to follow those works with open data, because the experiments can be repeated and the performance can be compared more easily. To be specific, the articles assigned to open access were associated with 89% more full text downloads and 42% more PDF downloads [41]. The number of citations is an important indicator of academic reputation. In the area of medicine, it is shown that publicly available data was significantly associated with a 69% increase in citations, independently of journal impact factor, date of publication, and author country of origin [129].

- *If the sharers are enterprises or public sectors, big data sharing is potential to help them receive increasing recognition and can mobilize continuous cooperation.* In recent years, a number of open government data portals came into being, such as data.gov.uk, data.gov, and data.gov.sg, which provide means for citizens and stakeholders to obtain government information about the locality or country in question [4]. Such open government data initiatives can establish public trust in government. In terms of enterprises, they can share big data through holding data analytics competitions to attract attentions from the public. For example, Kaggle, as an online community of data scientists and machine learning practitioners, has already received thousands of public datasets

and code snippets. Moreover, a lot of cases of cooperation between companies and researchers originates from the public datasets of the companies.

- *There is no doubt that the sharees can get monetary rewards by big data sharing*, more specifically, big data trading, from a commercial perspective, while the aforementioned two benefits are considered from a non-commercial perspective. Numerous data with great value is generated everyday because of the large number of users and inexpensive devices. According to the statistics, Facebook has more than two billion monthly active users and generates four petabytes of data per day. Although the data cannot be sold due to privacy issues and the terms of service, the value of the data is still there, which makes the market cap of Facebook to be over 690 billion dollars by July 17, 2020. In recent years, various big data trading platforms, e.g., Japan Data Exchange Inc. and Shanghai Data Exchange Corp., have been developed to meet the demands of the companies who possess valuable big data.

On the other hand, the benefits for the public good are summarized as follows:

- *Big data sharing helps in promoting academic integrity* [138]. Academic integrity, whose basic requirement is to avoid plagiarism and cheating during academic activities, is the moral code or ethical policy of academia. On the one hand, if the big data involved in scholarly activities can be shared, the scientific results will be easier to be reproduced by replication of the particular experiments. On the other hand, the researchers themselves will be cautious of publishing academic findings. To this end, a virtuous circle will be formed for promotion of academic integrity progressively. More generally, big data sharing helps the public keep the evidence of scientific results, which is essential for science to move forward.

- *Making the data available to their peers and the public incentivizes the researchers to better manage their data and ensure their data are of high quality.*

Redundancy reduction and data compression has been one of the challenging issues for big data for years [134]. Generally, there is a high level of redundancy in datasets. Big data reduction is effective to reduce the indirect cost of the entire system without affecting the potential values of the datasets. During big data sharing, the sharers may improve the data quality by application of various big data reduction techniques so as to attract more sharees.

- *Big data sharing encourages more connection and collaboration among researchers, which can result in important new findings within the field* [10]. Data are the foundation of scientific progress. Considerable efforts are required by researchers to obtain useful data, mostly through projects supported by the public fundings. However, their purpose is often limited to the production of scientific publications and, unfortunately, the vast majority of data are not fully utilized. In a time of reduced monetary investment for science and research, big data sharing is more efficient because it allows researchers to share resources. Data sharing allows researchers to build upon the work of others rather than repeat already existing research.

The benefits of big data sharing are summarized as above from the perspectives of the data sharers and public good. Every coin has two sides, and big data sharing is no exception. There are some potential drawbacks for big data sharing, for example, sharing of sensitive data raises privacy concerns and the data ownership will be complicated to be figured out during sharing. However, the drawbacks of big data sharing is not the focus of this chapter, and the readers may refer to [160] and [67] for more information.

2.1.5 Requirements of Big Data Sharing Solutions

Although there are numerous benefits for big data sharing, it is non-trivial to design a solution because of the requirements. In this subsection, we identify and explain

the basic requirements, i.e., *security and privacy*, *flexible access control*, *reliability*, and *high performance*, for big data sharing solutions.

First, *security and privacy* are essential for big data sharing. If a big data sharing solution is not secure or cannot protect the privacy, then the users can hardly trust such a solution and even will not use it. We concretize security and privacy into *data security*, *user privacy*, and *data privacy* as follows:

- *Data security.* Big data sharing is a kind of act upon big data between two parties. A big data sharing solution should prevent malicious access or modification of the big data by anyone other than the two parties [182]. In another word, anyone who is not granted with the right of usage by the data sharers can never access or modify the data. Moreover, the solution should be able to recover the big data and related sharing records if they are lost or destroyed by destructive forces.

- *User privacy.* There are two parties in big data sharing, which are the data sharer and sharee. User privacy is to protect the data sharer and sharee from exposure of their identities by other parties and even each other [30]. In particular, the identities of the two parties should not be revealed to other parties. Moreover, it is preferred that the two parties involved in big data sharing focus on the data itself without knowing each other.

- *Data privacy.* In real-world applications, there are numerous big data with great value but high sensitivity. For example, electronic health records of patients is very useful for disease diagnose but is extremely sensitive. As a result, the privacy of the big data is crucial for big data sharing. Generally, data masking or data obfuscation techniques [8] can be used to preserve the data privacy before big data sharing.

Second, a big data sharing solution is expected to support *flexible access control*,

which includes "sharing what", "sharing to whom", and "how to share". Specifically, sharing what and to whom means the data to be shared and the sharees, respectively, while "how to share" means the various ways to share big data. Some possible approaches of sharing big data are as follows:

- *Big data preview*: the big data sharer allows the sharee to view only the description or part of the big data possibly in a tool with restricted functions. In this mode, the sharees may not have full access to the original big data. The sharees have to estimate the value of the big data based on the description or a small part of the big data. Moreover, preview tools are often provided to view a downgraded version of the original data. By saying downgrading, it can be a slice if the original data is text, or a low-resolution version if the original data is video or audio [65].

- *Search over big data*: the big data sharer allows the sharee to query using predefined interface and responds to the queries. In this mode, the act of usage of the big data is restricted to be search. There are a variety of formats of the big data and all kinds of search operations, e.g., keyword query [76], range query [145], ranked search [25], and similarity search [183]. The big data sharer has the freedom to choose the data formats and allowed search types that can be performed by the sharee.

- *Nearline computation*: the big data sharee is allowed to perform operations using a combination of predefined interfaces. Nearline computation is an extension of search with a broader range of acts to use the big data. Specifically, the data sharee can not only search over the big data but can also perform certain kinds of operations such as add, deletion, and update. The word "nearline" means the computation is "nearly online" [168], i.e., not immediately available, but can be made online quickly without human intervention.

- *Big data transfer*: the sharer directly transfers the big data to the sharee so

that the sharee can perform possibly all kinds of operations upon the shared big data. After big data transfer, the data will be located at the place where the data sharee is. The operations that can be performed by the sharee after big data transfer depend on the contract between the data sharer and sharee. For example, if the data ownership is not transferred, then the data sharee should not spread the data from a legal point of view.

Third, a big data sharing solution is expected to ensure high reliability. In another word, the big data sharing platform should have very low possibility to malfunction. Such a requirement is very important because failure of the system may lead to lose of the high value of the big data. One of the key issue to ensure high reliability is to avoid single point of failure. Decentralization is very important because a centralized platform to maintain data suffers a lot from single point of failure. There are three functions for a big data sharing platform, i.e., data publishing, data search, and data sharing. The requirement of reliability should take care of all the three functions. In particular, once the big data is published, it can be searched by the public and shared by the data owner. Moreover, the results of a data search request should be sound and complete. Finally, there should be no way for the data sharer and sharee to violate the predefined rules after big data sharing.

Finally, high performance is preferred for a big data sharing solution. There can be a large amount of users generating a huge quantity of transactional records in a big data sharing platform continuously. The users include enterprises, organizations, government sectors, and individuals while the transactional records include publishing records, search record, and sharing records. It is a must for the big data sharing solution to deal with the numerous users and transactional records so as to provide a high quality of service. Moreover, the data involved is big data of high volume. The big data sharing platform should be efficient enough to publish and share the big data. Last but not least, data search should be completed with high performance. Data search is a very important way for the data sharees to find big data they need.

It is preferred that there are various ways for the data sharees to discover big data in a time-efficient way.

2.2 Existing Platforms and Categorization

In this section, we first introduce some popular big data sharing platforms all around the world. Then, we demonstrate our categorization of the existing platforms and illustrate how they works.

2.2.1 Existing Platforms

In this section, we introduce five popular big data sharing platforms in brief. The selection is based on a balance of the popularity, nationality, and system features.

Epimorphics Linked Data Platform

Linked Data Platform (LDP)[1] is a big data sharing solution developed by a software company called Epimorphics Ltd located in UK. LDP can be utilized in two ways. On one hand, LDP is a software solution that can be installed as local infrastructure for big data sharing. For example, a university can install LDP for different faculties, departments, and offices to share data. On the other hand, LDP is a platform that is being used by the UK government[2] to host data for a wide range of public and private sectors. In this chapter, we only consider the second usage because our focus is the existing big data sharing platforms.

The users can publish datasets to LDP and display the description together with links to the datasets. As shown in Figure 2.4, the description includes the title, publisher,

[1]Epimorphics

[2]UK Open Government Data

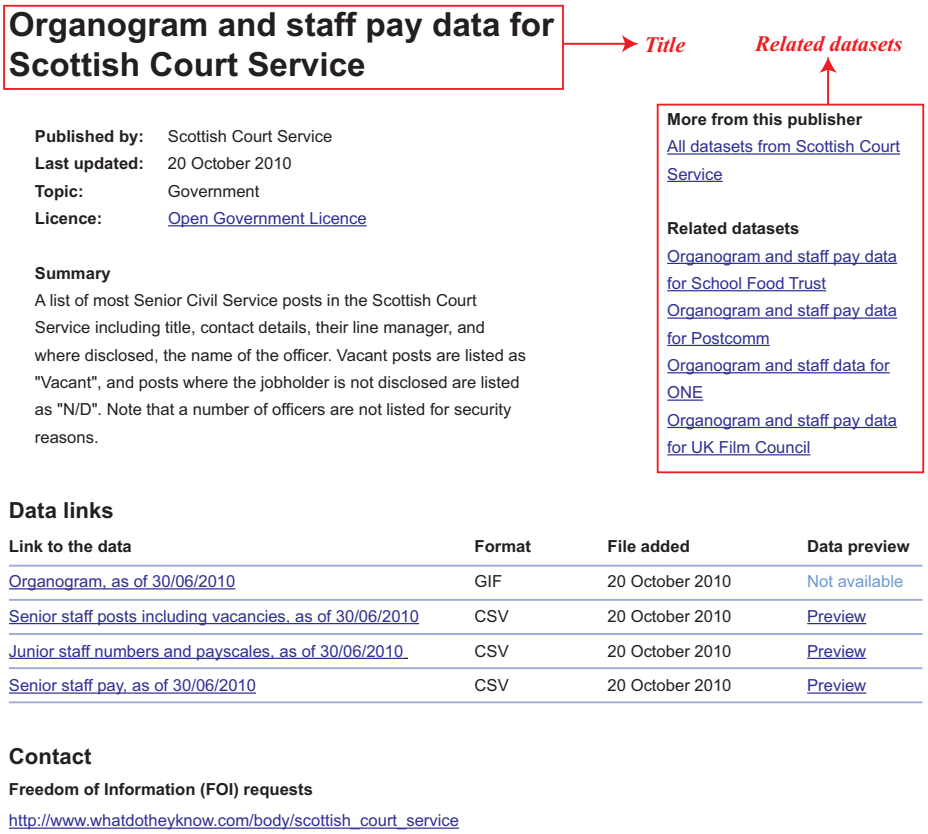

Figure 2.4: Display of big data on Epimorphics LDP

publication time, latest update time, topic, license, summary, data links, and contact. There can be multiple datasets in the field of data links, each of which contains the URL to the database, data format, publication time, and a preview if enabled by the data sharer. In short, the data sharers hold their data on their own servers and publish the description of the big data on LDP.

Besides data publishing, LDP enables data search in various ways. First, the big data can be searched based on general keywords. The data users input several keywords, and the big data with description containing the keywords will be displayed. Furthermore, the data users can filter the big data based on the publisher, topic, license, and data format. Finally, when displaying the description of a piece of big data, the related big data will also be shown to the data users.

In terms of data sharing, LDP only supports downloading raw data because the original big data is maintained by the data owners themselves. Meanwhile, LDP does not trace how the big data will be used if the data users have downloaded the data. Note that the implementation of the core functions are publicly available as ELDA[3].

HKSTP Data Studio

In 2007, Data Studio was launched by The Hong Kong Science and Technology Parks Corporation (HKSTP) in Hong Kong Science Park to support the smart city initiative of the government [114]. The main goal of Data Studio is to provide a platform for the public and private organizations to share big data so as to facilitate developing solutions for smart city. In particular, a large amount of government data[4] including population, education, housing, city management, employment and labor, and environment are available on Data Studio. Till now, there are 415 datasets contributed by 64 data publishers on Data Studio.

[3]ELDA: Epimorphics Implementation of Linked Data Platform

[4]Hong Kong Open Government Data

There are two ways for the data owners to share data in Data Studio. One of the approaches is traditional, in which the data owners can share static data via sharable links just like how LDP does. The other approach is non-trivial because the data owner can provide application interfaces (APIs) for the data users to fetch real-time data. For example, the transport department of Hong Kong government can provide APIs about the real-time locations of the buses. Such an approach is supplementary to the sharable links and improves the functions of big data sharing platform remarkably. Moreover, the data owners who provide APIs can make money by allowing data users to subscribe to their APIs. Finally, the copyright issue for the approach of big data sharing via real-time APIs is less severe because the data owners can maintain a list of data users who are subscribing to their APIs.

In terms of data search and data sharing, Data Studio shares the same approaches and disadvantages with LDP. That is, Data Studio cannot trace how the shared big data is being or will be used either. Moreover, the source code of Data Studio is not released yet and HKSTP does not provide a big data sharing solution for other enterprises to use.

SEEK

SEEK is a big data sharing platform developed by a group scientists for researchers in the area of systems biology to share datasets and model in projects [169]. Systems biology, which studies the computational and mathematical analysis and modeling of the complex biological systems, is such a special research area that highly demands sharing of heterogeneous datasets and complex models. The development of SEEK is exactly to connect the isolated datasets and models all around the world. SEEK is also provided as a big data sharing solution with open-source code[5].

The researchers can publish their data in the format of projects, in which the data

[5]SEEK Source Code

can be raw datasets, standard operating procedures (SOPs), models, publications and presentations. Version control is supported in SEEK for the researchers to update the shared data if necessary. In particular, if the models can be simulated within SEEK if following the systems biology markup language [71].

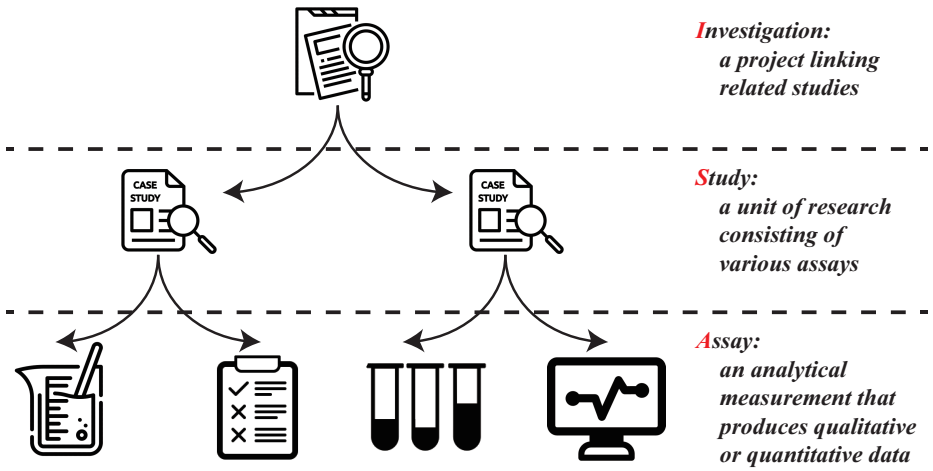

Figure 2.5: ISA model

After the data is published by the researchers, the metadata from the project will be extracted using the resource description format [24]. In this way, the data in the projects can be searched via semantic queries. A special characteristic of systems biology is that the datasets and models are inherently interlinked. In SEEK, the data model named ISA (investigation, study and assay) [137] as shown in Figure 2.5 is employed to assist in understanding and exploring the links.

One of the disadvantages of SEEK is that it is not universal for sharing general big data because of its specialization on systems biology research. Moreover, once the data is shared on SEEK and downloaded by the data users, there is no clue how the data will be used just like LDP and Data Studio.

InterPlanetary File System

IPFS (the InterPlanetary File System)[6] is a distributed peer-to-peer (p2p) system for storing and sharing data. IPFS was initiated by Juan Benet and Protocol Labs [15] and is open-source and maintained by the public now[7].

When a user wants to share a piece of content, a content identifier, or CID, will be generated by serializing and hashing the content. Such a CID is the unique identifier for the other users to find the content, which is similar to an address. The IPFS p2p network stores the users who can provide each piece of content in distributed hash tables. Each user joining the IPFS p2p network can manage the local data by indicating what they have, they want, and do not want.

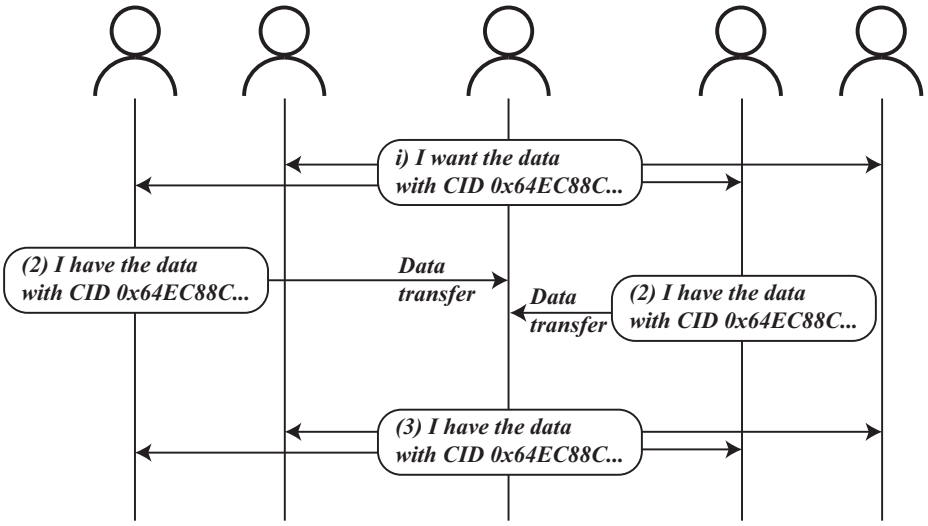

Figure 2.6: The workflow of IPFS

In the IPFS P2P network, all the users are equal to each other. As shown in Figure 2.6, if the user who wants a piece of data can fetch the data from other users, and at the same time, become a data provider. Meanwhile, if the user who shares a piece of data can get the benefits for data backup by other users.

[6]IPFS: a Peer-to-peer Hypermedia Protocol Designed to Make the Web Faster, Safer, and More Open

[7]IPFS Source Code

Although IPFS has been praised a lot by the public, it also faces many issues for big data sharing. The major challenge is that the users are not well motivated to use IPFS to share big data because they have to spend a lot of storage and network resources to exchange the data from others. Indeed, public cloud storage is a better alternative because it is easy-to-use and there is no need for maintenance. Moreover, the weak motivation for the users leads to shortage of nodes in the IPFS P2P network, which makes the distributed IPFS system unreliable in return.

Recently, Protocol Labs has proposed Filecoin, a platform to employ blockchain technology to motivate people to use IPFS, in which people can get monetary rewards by providing data storage resources [14]. However, at the meantime, the platform users have to pay a lot for data storage. To this end, Filecoin will have to compete with current cloud storage providers, which may be a tall order.

Amazon Web Services Data Exchange

Amazon, as one of the leading cloud service providers, on November 2019 launched a new service called Amazon Web Services (AWS) Data Exchange aiming to provide the customers with a secure way to find, subscribe to, and use third-party data. More than 80 data providers have contributed over $1,000$ datasets at launch as announced by Amazon. Till now, there are around $1,496$ datasets published by 110 data providers.

There are two roles on AWS Data Exchange, i.e., data providers and data subscribers. The providers can publish free or paid datasets under the terms of usage they specify, in which the terms include pricing strategy, access policy, etc. Moreover, the providers can update the published data using the built-in versioning tool. As for the subscribers, they can subscribe to the public datasets with a monthly or annual plan. Within the subscription period, the subscribers can load the datasets for local storage and will receive notifications about updates of the datasets. One of the advantages brought by AWS is that AWS Data Exchange is seamlessly integrated with other

AWS services. For example, the datasets can be easily loaded to AWS S3, which is a cloud storage service provided by AWS, through web interfaces, APIs, and even command lines. Moreover, the notifications of dataset updates are pushed via AWS CloudWatch events, which can be consumed in a real-time manner.

AWS Data Exchange provides a convenient platform for different organizations to publish significant datasets, for example, around 200 datasets for the COVID-19 epidemic. Every coin has two sides, and there are still several disadvantages hindering the development of AWS Data Exchange. First, the data providers need to pay for storage of the published datasets. The huge volume of big data increases the data sharing cost substantially. Furthermore, the customers who subscribe to datasets must provide the personal information to the data publishers. The provided information raises the privacy concern. Finally, the datasets published by the data providers are stored on AWS servers, and there is no guarantee that AWS will not abuse the datasets although there are legal terms between AWS and the data providers.

2.2.2 Categorization of Existing Platforms

In this section, we describe our categorization of existing platforms, i.e., data hosting center and data aggregation center, and decentralized data sharing platform. Furthermore, we compare the platforms described in Section 2.2.1 according to their categories, nationalities, unique features, etc.

Data Hosting Center

The functionalities of a data hosting center (DHC) are similar to the ones of a portfolio manager in finance. A portfolio manager raises money from the investors and invest in the stock or bond market to get monetary rewards. Similarly, a DHC collects the original big data from the data owners and find possible data users to share the big data and get rewards. The portfolio managers and DHCs have full control of the

money from investors and the big data from data owners, respectively. Nevertheless, DHCs are different from portfolio managers because big data is different from money in terms of the possibility of duplication. Since big data is easy to be replicated, DHCs are often used for open big data, that is, the big data shared on DHCs are purposely opened to the public.

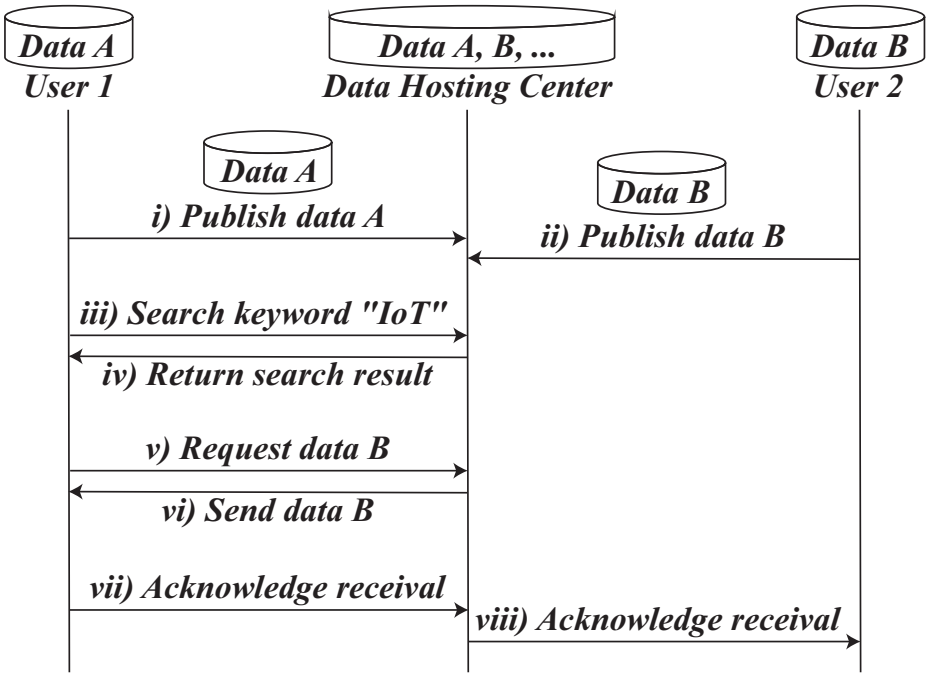

Figure 2.7: The workflow of data hosting center

Figure 2.7 shows the workflow of a DHC. First, the data users, i.e., user 1 and 2, publish big data, i.e., data A and B, at the DHC, respectively. Then, user 1 searches all the big data using the keyword "IoT", in the DHC. The DHC responds to user 1 with all the descriptions of the big data related to "IoT". Subsequently, user 1 request data B, whose owner should be user 2, from the DHC. Finally, the DHC transfers data B to user 1, user 1 sends the acknowledgment of receival to the DHC, and the DHC forwards the acknowledgment to user 2.

One of the key features of DHCs is that the DHCs collect the big data from users and are expressly delegated to share the big data. SEEK is a typical example of DHCs,

i.e., the researchers upload their project data to the SEEK platform while the SEEK platform simply makes the project data publicly accessible for everyone.

One of the major advantages of DHCs lies in the efficiency. DHCs serve as the centralized repository for all the shared big data. Because of the centralization, general optimization methods for data storage systems can be applied. For example, the popular big data can be replicated in advance to improve the data transfer speed. Moreover, caching techniques can be applied to process big data queries.

Another advantage of DHCs is the authenticity of the big data can be guaranteed. If the big data is not stored in DHCs, it is possible that the data owners claim the big data that they do not have. Even worse, the data owners can deny to share the big data although they promise to do so. the DHCs can serve as the intermediary between the big data owners and users.

Of course, there are disadvantages for DHCs, among which the data privacy is the most crucial one. In particular, the DHCs can replicate the big data without the approval from the data owners. It cannot be guaranteed that the big data will not be shared according to the policy defined by the data owners.

Data Aggregation Center

The role of a data aggregation center (DAC) to the big data is similar with the role of a real estate agency to the real estate. A real estate agency aggregates the information of the real estates from the estate owners, post advertisements to attract the buyers, and facilitates the deal between the real estate owners and the buyers. Similarly, a DAC collects the descriptions of the big data from the data owners, provides a platform for the potential users to search the big data, and facilitates big data sharing between the data owners and data users. The real estate agencies and DACs only have the general information of the real estates and big data, respectively, but cannot control at their disposals. Of course, the DACs are different from the real

estate agencies because the authenticity of big data cannot be easily verified but the real estates can.

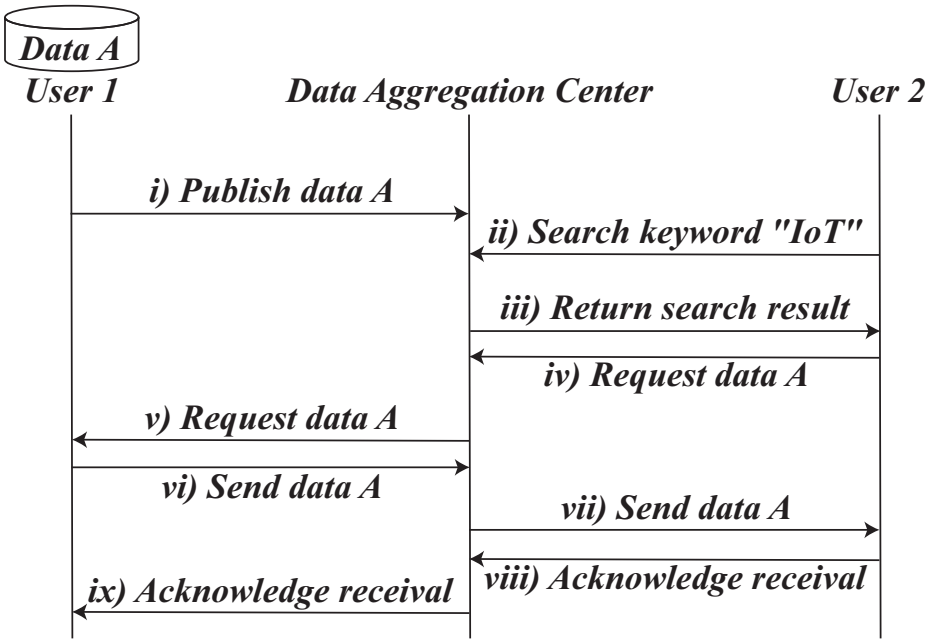

Figure 2.8: The workflow of data aggregation center

In DAC model, the Center links data services through the API interface among agencies. Data agencies do not need to report, upload to the DAC in advance. The data is still owned and managed by the data agencies. When an agency needs to search the data, it will use the real-time interaction with the DAC to send the data request. The DAC will relay and broadcast this request to other agencies. Once other agencies with the target data response to this request and return the data, the DAC will collect all the data and send back to the data demander. However, it is not hard to find that the DAC has the ability and the opportunity to retain the data. The DAC can accumulate the data during sharing, and it will gradually become a DHC.

2.3 Challenges and Potential Solutions

Unlike traditional commodities, data, as a particular non-exclusive resource, has the characteristics of rapid growth, low copy cost, unknown potential value, trying to determine ownership, and challenging to control circulation channels. It is designed to be efficient, credible, fair, secure data sharing, and the trading market has brought many challenges. In this section, we will introduce the current challenge issues in big data sharing problems and give some existing research on each issue.

2.3.1 Standardization of Heterogeneous Data

The data set provided by the data owner needs to be standardized. First, the data needs to be audited to ensure that it is stored correctly as required. Then it is necessary to normalize the structure of the data set to generate an efficient data set. Finally, to allow data users to understand the characteristics of the data better, a data summary should be provided.

Data owners tend to store their data on cloud servers, but cloud servers are not completely honest. Some malicious cloud service providers may discard some infrequently used data to reduce resource overhead. Therefore, before big data is shared, it is necessary to ensure that the data owner's requirements correctly store the data. This type of problem is called proof of retrievability, proof of storage, and data ownership. In recent years, some related issues have been studied. For example, Kun He et al. proposed an effective structure called DeyPoS [64] to achieve dynamic PoS and secure cross-user data deduplication simultaneously. Jia Yu et al. studied how to Reduce the damage of customer key exposure in cloud storage audit [180], and provide a practical solution for this new problem setting.

Many data sets have a certain degree of heterogeneity in type, structure, semantics, organization, granularity, and accessibility. Incorrect data will reduce the value of

the original data and may even hinder useful data analysis. Therefore, in order for the data to be better analyzed by the computer and understood by the users, proper pre-processing, such as data cleaning, standardization, calibration, fusion, and desensitization, is required before sharing the to generate valid data for the original data provided by the data provider. Specific data sources (such as sensor networks) may generate an alarming amount of raw data. Most of these data are meaningless and can be filtered and compressed by orders of magnitude. One challenge is to define these filters in a way that does not discard useful information. The second challenge is automatically generating the correct metadata to describe the data to be recorded and how to record and measure the data. This metadata may be essential for subsequent data analysis. Generally, the collected information will not be in a format that can be used for analysis. An efficient data representation should reflect the structure, class, and type of the data, and integration technology to achieve efficient operations on different data sets. Therefore, information extraction is also crucial. The process extracts the required information from the underlying resources and presents it in a structured form suitable for analysis.

In order to attract data users and help them better choose the data sets they need, big data sharing platforms need to display data information (such as themes, advantages, and scope of application). Different from physical commodities, data is easy to copy. Therefore, if the data is wholly displayed to all data users, users can directly obtain the data through copying without purchasing. To avoid this situation, the data needs to be processed to generate a summary of the displayed data. Part of the current work is mainly through four ways of data sampling, data version, metadata, and data summary. Existing platforms usually use manual methods to generate metadata and summaries. However, in the face of massive amounts of data, the automatic generation of data display is an important method to improve efficiency and accuracy. Since the generation of abstracts requires a semantic understanding of data, it is generally much more complicated than metadata generation. A series of

works have used machine learning and other methods to generate summaries for text, audio, images, and videos. In order to provide data summaries faster and better, in addition to the need to continuously improve the summarization model itself, it is also necessary to fully consider the execution efficiency of the method on large data sets and the security of the data, that is, too much valuable information cannot be leaked. Besides, the demand for personalized abstracts (generating abstracts that meet buyer needs to increase sales) also increases the difficulty of abstract generation.

2.3.2 Value Assessment and Pricing Model

When we share data as a commodity, we need to give a reasonable estimate and accurate quantification of the data's quality and value. Moreover, to protect the rights and interests of data sharing participants, establish a fair and credible standardized market, and maintain a healthy data sharing environment, data quality evaluation and value evaluation are even more essential. The quality evaluation focuses on the multi-dimensional characteristics of the data content itself, such as completeness, accuracy, accuracy, consistency, timeliness, etc. The value evaluation is to evaluate the quality of the data while further comprehensively considering the cost and cost of the data in the production process—output in different applications. In particular, the characteristics of easy copying of data and difficulty in controlling distribution channels make it difficult to return sold data, which further increases the requirements for accurate and reliable quality and value evaluation of data before sharing. Data users need to understand the quality and value of the data itself before paying for it, to select appropriate data, and give a reasonable budget. The data provider needs to know the quality and valuation of the data to give a reasonable price.

Moreover, strive to improve data quality and enhance data value. Big data sharing platforms need to monitor the quality and value of data to filter out low-quality data, prevent malicious quotations from disrupting the market, and at the same

time, recommend cost-effective data to users, thereby improving user satisfaction and platform reputation, and building a healthy market ecology surroundings. Related research is currently facing many challenges, such as difficulty in the quantitative evaluation of features, low efficiency in data collection quality evaluation, difficulty in data cost calculation, difficulty predicting the value of data use, and dynamic changes in data value. The characteristics of easy data duplication and delicate control of distribution channels make it difficult to return the sold data, further increasing the requirements for accurate and reliable quality and value evaluation of pre-sale data.

Current research work generally believes that data quality is a multi-dimensional concept, and different work has proposed a variety of definitions and evaluation methods for data quality dimensions [165]. Based on existing work, five critical dimensions have been generally recognized: intrinsic quality, presentation quality, content quality, accessibility, and reliability.

- *Intrinsic quality:* basic requirements, designed to measure the quality of the content itself in terms of capacity, accuracy, completeness, timeliness, uniqueness, consistency, security, source reliability, etc. This quality dimension mainly depends on the data source and the process of data collection and processing.
- *Presentation quality:* is a high-level requirement concerning the data format (concise and consistent presentation) and data meaning (interpretability and ease-to-understand).
- *Contextual quality:* the data must be relevant to the context of the current decision-making process (task) and suitable for the target application scenario.
- *Accessibility:* the extent to which the data can be used or retrieved by the buyer, such as communication costs and delays in data delivery.
- *Reliability:* refers to the reputation and credibility of the seller and the data collection source.

High-quality data should have good inherent quality, clear data representation, easy access by data consumers, and suitable for specific application scenarios (tasks).

The value of data depends on the quality of the data and is also affected by cost and market. The value evaluation methods of traditional intangible assets are roughly divided into cost-based, market-based, and revenue-based. The following is the definition of data value when these three evaluation methods are used:

- *Cost-based:* the value of data depends on the cost of collection, processing, and storage. But data is usually generated as a by-product of information systems, and production costs are shared with other products. Therefore, it is difficult to determine the production cost of data, and its production cost is also difficult to reflect the true value of the data.

- *Market-based:* the value of data depends on the market price of comparable data sets in the same market, but the definition of "comparability" is not clear, and there are usually no very similar data sets, so accurate valuation is difficult.

- *Revenue-based:* the value of data depends on the total income that the buyers can obtain from the data. Such a method is subjective and is only available when evaluating specific applications. The valuation of different buyers may vary significantly.

In summary, there is still a lack of models and methods that can accurately assess the full value of data. Facing the huge volume and various types of data in the data transaction market, the quality and value evaluation of data still faces the following challenges:

- *Multi-dimensional quantitative evaluation of quality:* existing work provides rich evaluation dimensions of data quality, but many dimensions still use qualitative analysis and lack specific quantitative models. When facing a large amount of

unstructured data, it is very challenging to analyze the content of the data, and even more challenging to quantify the quality of its content.

- *Data collection quality assessment:* Most current work assesses data quality for a single data unit (such as a text, a picture), and there is a lack of methods to assess the overall quality of the data collection. However, most of the data sets (such as 10,000 texts and 100,000 pictures) are shared or sold on data sharing and trading platforms. If the data set's overall quality is obtained through the quality statistics of a single data unit, the influence of the relationship between the data units on the quality of the data set is ignored.

- *Dynamic evaluation of data value:* Data of different modalities and contents have different rarity and difficulty in obtaining. How to reasonably evaluate these factors and combine the quality of the data to form a quantitative estimate of the overall value of the data is still difficult. Existing data evaluation work usually focuses on the static quality of data, ignoring the dynamic characteristics of data value, that is, as data collection and storage devices are updated, data mining model methods are optimized and updated, and application scenarios and data consumer needs Change, the value of data is also changing. These dynamic factors further increase the difficulty of data value estimation.

2.3.3 Security

Generally, security refers to the three main security attributes that need to be ensured to protect data, namely data confidentiality, integrity and availability [17], also known as the CIA triad.

Confidentiality: All virtual sensitive data (input, output, and any intermediate state calculations) are secret to any potentially adversarial or untrusted entities [176]. In other words, confidentiality is the protection of data to prevent unauthorized read

access. Access control and encryption technology are effective means to ensure data confidentiality, and both have been extensively studied.

For access control technology, we need to protect data from unauthorized access and manage authorized users. Generally, in the process of significant data sharing, fine-grained and flexible access control often needs to meet the following requirements:

- Decide whether there is a limit on the time the user is authorized.
- The user's authority is divided into the data ownership and right of usage.
- Determine whether the user has the right to re-share the data.
- User permissions can be completely revoked flexibly.

On related issues, Elisa Bertino and Elena Ferrari have conducted some surveys [16], this article analyzes on this basis, mainly in the following aspects:

- *Consolidation and integration of access control strategies:* In many cases, the data set required by data users is not necessarily a single one, which requires the integration of data sets from multiple possible heterogeneous sources. Therefore, it is necessary to merge and integrate the access control policies of several data sources. However, there are some automated or semi-automated strategic integration systems to resolve conflict issues [105]. However, in data sharing, allowing data providers to develop their access strategies will become much more complicated. There may even be cases where access to some data is allowed, but some obligations are required to travel. Therefore, how to automatically integrate and merge these strategies is still challenging.
- *Authorization management:* In the case of fine-grained access control, manual authorization management of large data sets is very consuming human resources. Therefore, we need technology that can automatically grant authorization, possibly based on the user's digital identity, profile and context, and

data content and metadata. Ni et al. took the first step in developing machine learning technology that supports automatic permission assignment to users [121]. However, more advanced methods are needed to deal with dynamically changing contexts and situations.

- *Implement access control on big data platforms:* With the rise of big data platforms, the sources of users who use data have become more complicated. In order to effectively implement fine-grained access control to different users, this brings significant challenges. Although there is some initial work trying to inject access control policies into the submitted work [157], more research is still needed to study how to effectively implement such strategies in the recently developed big data storage, mainly If people use fines to enforce access control policy granular encryption.

In addition to access control, encrypting the data can also effectively ensure the confidentiality of the data. Many encryption techniques can be used to help protect the security of cloud computing, including functional encryption [20], identity-based encryption [19], and attribute-based encryption [139]. Among them, attribute-based encryption can be divided into Key-Policy attribute-based encryption and Ciphertext-Policy attribute-based encryption. Although data encryption can protect data security, this technology is limited by many keys and the complexity of key management. At the same time, the demand for the computing power of existing encryption schemes cannot be ignored. Therefore, to share big data, it is still a challenge to design lighter and more flexible encryption algorithms.

The existing encryption algorithms to ensure data confidentiality mainly include homomorphic encryption (HE). Gentry launched the first fully homomorphic encryption scheme [56] in 2009. This is a revolutionary cryptographic technology achievement, but the scheme is too inefficient for any practical use. Since 2009, many researchers have improved Gentry's technology, greatly shortening the production time. How-

ever, in most cases, the speed of fully homomorphic encryption is still too slow. In addition to inefficiency, homomorphic encryption has other limitations. For example, homomorphic encryption requires all sensors and the ultimate recipient to share a key to encrypt the input and decrypt the result, which may be difficult to arrange if they belong to different organizations. Similarly, homomorphic encryption does not allow calculations on data encrypted with different keys (without incurring a large amount of additional overhead), so sensors cannot allow different access to the data they contribute. Some of these limitations can be solved by other tools, such as the aforementioned attribute-based encryption and functional encryption. However, homomorphic encryption only guarantees data confidentiality, not integrity. However, it can be used in conjunction with correct calculations to provide two guarantees.

Integrity: Any unauthorized modification of sensitive data can be detected. Besides, the output of any calculation on sensitive data is correct (i.e., consistent with the input data). In other words, integrity means that the data is not subject to any unauthorized modification. However, with the development of the Internet, people's demand for data has increased. Data integrity has been further generalized to data trustworthiness, which means not only to ensure that unauthorized parties do not modify the data but also to ensure that the data is error-free, up-to-date, and from reputable sources. Therefore, ensuring the trustworthiness of data is a tricky issue, usually depending on the application domain. Its solution requires a combination of various technologies, from cryptographic technology used to digitally sign data, access control, checking that only authorized parties modify data to data quality technology, automatic detection and repair of data errors [9], source technology [150], used to determine where the data originated, and reputation technology, used to evaluate the reputation of the data source.

Availability: It is an attribute that ensures that the data is available to authorized users. It is necessary to ensure that the data obtained by the user is available and that what the user obtains is the data he requested.

Finally, these three requirements are still very critical today, and due to more and more data collection activities and data sharing, data attacks have become more complex, and the data attack surface has expanded, so meeting these requirements is more challenging today.

2.3.4 Privacy

In recent years, with the increase in data demand and the development of significant data sharing and the CIA's attributes, privacy has become a new essential requirement. The privacy issues discussed in this article are mainly divided into two aspects: data privacy and user privacy.

Data privacy: So far, many definitions of data privacy have been proposed. Moreover, with the development of means of obtaining personal information, privacy has also developed over time. Samuel Warren and Louis Brandeis published an article [166] published in 1890, in which the concept of privacy was systematically written for the first time In one of the discussions, they defined privacy as the "right to be isolated." Nowadays, one of the most commonly used definitions of data privacy is Allan Westin's definition of data privacy as "individuals, groups, or organizations that require themselves to determine when, how, and to what extent information about them Convey the claims to others" [167].

Data privacy is generally regarded as the same requirement as data confidentiality. However, there are some differences between these two requirements. Data privacy does need to ensure the confidentiality of data because if data is not protected from unauthorized access, privacy cannot be ensured. However, due to the need to consider the requirements of laws and privacy regulations and personal privacy preferences, privacy has other issues. For example, the purpose of data sharing is critical when dealing with privacy, because it may be a good thing for one person to share their data for research purposes, while another person may not. Therefore, systems that

manage privacy-sensitive data may have to collect and record individuals' privacy preferences to whom the data refers. Also, data subjects may change their privacy preferences over time. Therefore, in order to solve the privacy problem, the system not only needs to implement the access control policy that the organization may have to control access to data but also needs to implement the preferences and laws and regulations of the data subject, among other things.

In addition to effective access control strategies, data encryption technology is also particularly important. This is because big data privacy management usually relies on the cloud platform. The primary issue for implementing privacy management under the cloud platform is the security of storage, computing on encrypted data, and communication. Data encryption technology meets this demand. Specific applications under cloud platforms usually depend on data storage, indexing and retrieval, and the credibility provided by cloud platforms. The homomorphic encryption and functional encryption mentioned in data confidentiality are standard methods to protect individual data privacy. Hu et al. [69, 68] respectively proposed key-value privacy storage methods and multi-level index processing technologies using homomorphic encryption technology to ensure that neither the data owner nor the cloud platform can be identified during the node retrieval process of the user query out of any node.

Whether it is encryption technology or access control, both are designed to design heuristic protection methods against current external attacks. In the face of new attacks, it is necessary to reformulate protection methods. In the big data environment, these two methods are not universally applicable due to the lack of a strong mathematical foundation to define data privacy and loss. The emergence of differential privacy [43][44][46][47][45] makes up for this gap. This model is a new and robust privacy-protection technology supported by mathematical theory. According to the formal definition of differential privacy, this method controls the degree of privacy protection and the size of privacy loss by privacy parameters, which can ensure that the operation of inserting or deleting a record in a data set will not affect the output

of any calculation. Also, this method does not care about the background knowledge of the attacker. Even if the attacker has mastered the information of all records except a certain record, the privacy of the record cannot be disclosed [184]. This feature makes the difference Privacy technology has good scalability. Differential privacy has become the current research hotspot of privacy protection technology. The academic community believes that differential privacy has a natural match with big data. The reason is that the large-scale and diversity of big data makes the addition or deletion of a data point in the data set have a minimal impact on the overall data. This characteristic is related to the difference. The definition of privacy coincides with the connotation. Nevertheless, the disadvantages of differential privacy protection technology include the inability to actively control privacy parameters. Small privacy parameters lead to low availability and high privacy, and vice versa, high availability and low privacy. Therefore, this parameter is difficult to control. The correlation between big data may weaken the effect of differential privacy protection.

To better protect the privacy of data, searching directly on the ciphertext is also an effective way to protect data privacy. The ciphertext retrieval processing technology is divided into symmetric encryption [85] and public-key encryption method [1, 135]. Among them, Kamara et al. [85] proposed an asymmetric encryption method that supports dynamic retrieval, which has Higher security and retrieval efficiency. Abdalla et al. [1] proposed searchable public-key encryption technology and support keyword retrieval; while Rhee et al. [135] proposed against Abdalla et al. [1]. The problem of secure channels and consistency, a public-key encryption scheme based on random prediction models, and unsecured channels is proposed. Also, functional encryption allows us to learn the information implicit in the ciphertext when processing the key.

Also, the privacy information retrieval technology [32] is usually used for query security when outsourcing data. Users can query any data on an untrusted service platform without revealing sensitive information of the queried data. The queried data can be public and anonymous, but the service platform cannot identify its spe-

cific content. Although the aforementioned encrypted search technology can also control the query, the query's complexity and computational overhead make this type of technology not practical. The technology to realize privacy retrieval usually includes two types: retrieval methods based on information theory and computable retrieval methods based on hardware. Among them, the retrieval method based on information theory [32] is usually to pass all the data to the client and allow it to be decoded locally. However, due to the transmission cost, this technique is not suitable for big data. Computational retrieval methods based on hardware are currently more commonly used especially in DNA sequence matching, content-based image retrieval, and location privacy queries. Although privacy information retrieval technology has promoted the development of security software and hardware, the application of this technology will be more difficult and complicated in the big data environment.

User privacy: In addition to the accidental protection of data privacy, the identity information of data users and data providers also needs to be protected. This is the issue of user privacy. In terms of protecting user privacy, anonymization technology and all encryption signature technologies are commonly used.

Anonymization refers to hiding or obscuring data and data sources. This technology generally uses operations such as suppression [163], generalization [52], analysis [174], slicing [102], separation [155], etc. to operate anonymous data. k-anonymous [152] is the new representative method of this technology. When publishing relational data, this method requires that each generalized equivalence class contains at least k pieces of mutually same data, that is, it requires one piece of data to represent Personal information is at least indistinguishable from other k-1 pieces of data. However, the disadvantage of k-anonymity is that it does not restrict the equivalence class's sensitive attributes, which leads to the failure of the technology. For example, any sensitive attribute in a certain equivalence class takes. If the value is the same, the attacker can infer the sensitive value. Unlike k-anonymity, the l-diversity [115] method ensures that each equivalence class contains at least l different sensitive

attribute values when anonymizing relational data. Although l-diversity guarantees the diversity of sensitive attributes but ignores the global distribution of sensitive attributes, and the attacker may confirm the sensitive value with a high probability. To make up for the shortcomings of the l-diversity method, the t-closeness [100] method requires that the distribution of sensitive attribute values in all equivalence classes is consistent with the global distribution of the attribute. Also, minvariance [175] and HD-composition [22] fill up k-anonymity, l-diversity, and The t-closeness method is only suitable for the shortcomings of static relational data to ensure that the privacy of data is not leaked when the data is released dynamically or incrementally. Compared with the anonymization problem mentioned above, the anonymization of big data is more complicated. In big data, The integration and fusion of multi-source data and correlation analysis make the above passive protection methods for small data invalid. Compared with the current privacy management framework, the traditional anonymity technology has the disadvantage of passively preventing privacy leakage, combined with a single data set. The attack assumes that to formulate the corresponding anonymization strategy. However, the large-scale and diversity of big data makes traditional anonymization technologies lose sight of the other.

In addition, secure multi-party computing [158, 144], as a commonly used encryption method, can also guarantee the privacy of users. Secure multi-party computing's core operation is to calculate the corresponding function value based on the data provided by the multi-party participants in a distributed environment and to ensure that in addition to the input and output information of the participants, no additional information of the participants is exposed. The technology is often used in the field of data mining [158] for privacy protection in a distributed environment, and gradually expanded to the fields of undirected product [42] and adding vector [159]. Also, ring signatures, zero Techniques, such as zero knowledge proof, are well used for user privacy protection.

Although the above research provides specific ideas for big data privacy management,

the flaws of this technology are obvious. Like anonymization technology, this type of technology is also passive protection against the privacy of certain types of data. Moreover, in the big data environment, Its large-scale, diversity, and other characteristics make this type of technology fall into a cyclical cycle. In the face of privacy leakage of new applications, new encryption methods must be used to protect it.

2.3.5 Data Traceability and Accountability

In an ethical and sustainable data trading market, the traceability of data dissemination after the transaction is completed very importantly to the reliability of the entire system. It determines the user's satisfaction and trust in the system. In the process of significant data sharing, it is not just the sharing process that needs to be paid attention to. It is also essential to trace the data after sharing. However, it is challenging to design a traceable data sharing mechanism.

First, it is difficult to guarantee the traceability of data because attackers may take arbitrary measures to avoid data being tracked and identified during propagation. Second, plagiarism is difficult to detect because users may modify a small part of the data and then list it as a "new" data set. Finally, it is difficult for data agents to detect illegal data transactions offline. In order to solve these problems, many researchers have made efforts in recent years. One method is to introduce a third-party trusted institution to supervise all shared transactions, add a watermark to each transaction's data, and verify the existing watermark of the data before allowing the operation of the data used to check whether the sharing rules are violated. However, collusive users and offline data circulation can bypass surveillance. At the same time, for data plagiarism detection, Jung et al. [83] designed related technologies. In order to consider the originality of the data, they defined the originality index of various data types and verified the effectiveness of this indicator by experiments. Some tools can be used for data tampering detection and data duplication, such as Merkle trees,

digital signatures, and local sensitive hashes.

Many existing data transaction platforms claim to have used blockchain technology for data traceability due to the emergence and rapid development of blockchain technology. This takes advantage of the non-tamperable, traceable, and trustworthy features of distributed data storage in the blockchain. The classic blockchain, that is, the blockchain used by Satoshi Nakamoto in the Bitcoin system in 2009, combines asymmetric encryption, digital signatures, Merkel trees, proof of work, and other technologies. The safe and reliable record of transfer information in the absence of a trusted center provides a complete solution. Then, the researchers made different modifications and additions to the classic blockchain to adapt it to various information sharing and recording requirements in different scenarios. In 2013, Vitalik Buterin (Vitalik Buterin) proposed an open-source, public blockchain platform with smart contract functions—Ethereum. Retaining the payment and transfer functions of the previous Bitcoin blockchain provides an open, modular platform that supports custom advanced applications. Ethereum supports users to edit the applications they want, that is, smart contracts. The calling process and return result of the contract are recorded in the underlying blockchain, which is equally safe, reliable, and immutable. Researchers have proposed a variety of possible data transactions and traceability solutions based on the blockchain and related technologies. For example, by building an underlying blockchain, the essential functions of distributed data transactions can be completed. Alternatively, by establishing a unique data file contract for each piece of data, and record the relevant information of the data in the file contract bound to it, and the sale of the data is realized by calling different functions of the file contract. The content will be retained. The label information and data copyright are recorded on the chain, and the immutability of the information on the chain is used to realize the anti-counterfeiting and copyright confirmation of the data. The data transaction intermediary can establish a contract warehouse locally, organize the disordered contracts on the chain in an orderly manner, and achieve efficient services through the

combination of on-chain and off-chain.

However, in addition to data piracy, the data being traded may also be leaked through offline copies. Because the data owner may hold a copy or slightly modified version of the data, and may also transfer the data to other storage devices or others. Since the data itself is easy to copy, change, and transfer, these risks of infringement cannot be eliminated. These risks also exist in existing digital goods, such as e-books, music, and movies. Existing digital rights management protects the copyright of digital goods through encryption and the development of dedicated software and hardware. For example, users can only use specific software to view e-books or listen to music and not allow downloading and use product keys to limit the number of software installations. Methods such as continuous online identity verification automatically detect piracy. In data transactions, sending plaintext data directly to buyers will result in unrecoverable infringement losses, so we must use DRM technology to restrict the use, download, and dissemination of data. The existing DRM technology is not perfect enough. In order to better adapt to significant data sharing, the following improvements need to be made:

- *High-speed real-time online access:* Data access is not continuous like streaming media, and may be arbitrary, so the efficiency and speed of online access need to be improved.

- *Improvement of dedicated software:* Existing dedicated software with copyright management generally only allows users to browse (such as watching videos and listening to music). In data transactions, such software must not only support browsing, but also allow buyers to calculate and visualize data.

- *Function restriction mechanism:* Some software needs to prohibit the screen capture function and use some mechanisms to prevent buyers from taking pictures or videos of the screen and indirectly infringing.

- *Infringement detection:* It is not only necessary to use product keys and continuous online identity verification to prevent infringement, but also to detect whether infringement has occurred, for example, recording the sending and receiving devices when data is transmitted, and combining the copyright restrictions in the data transaction contract , The software should automatically determine whether the buyer has infringed, and if so, it should punish the buyer by disrupting the data or completely banning the use.

2.3.6 High Quality of Service

There are multiple services on the network that will generate a large amount of data, and these data are significant. Many users around the world may be interested in data generated on the other side of the world. As the Internet provides a universal infrastructure, sharing scientific data for scientific research and engineering data for manufacturing has also become a modern trend [153]. Therefore, it is necessary to deliver big data on the network to users according to their needs. This is the concept of big data services.

Existing big data services are challenging to meet the needs of hardware and software, so how to develop high-performance big data services is an urgent problem to be solved. The main challenges in this regard are as follows:

- *Offline storage device:* hard disk drives with random access technology are limited for data storage, especially for fast input and output transmissions that require big data processing [86]. While robust state equipment (SSD) [72] and phase-change memory (PCM) [128] are more advanced technologies, they are far from reality.

- *Management algorithm design:* management algorithm design: There may be algorithm design limitations in defining appropriate data structures suitable for

fast-access data management. We need to optimize the design and implementation of the index to access the data [31] quickly.

- *Data transmission security:* for big data services, communication is almost essential because data collection and service delivery are usually carried out on the Internet. Big data services require large bandwidth for data transmission. There is always the possibility of data loss during transmission. In this loss situation, maintaining data integrity is a challenge [164]. More importantly, there are always data security issues [125].

- *Powerful computing power:* as the scale of data expands, the demand for computing power increases exponentially. Although Moore's Law doubles the processor's clock cycle frequency, the clock speed is still far behind the demands. Therefore, mighty computing power has become part of the demand for high-quality big data services. Both data sharing and analysis require high computation power.

- *Timeliness of services:* ensuring the timeliness of big data services is a major challenge. This not only requires high computing power, but also requires innovative computing architecture and powerful data analysis algorithms.

2.4 Promising Applications

In this section, we introduce two promising applications of big data sharing in recent years, i.e, healthcare and data trading. The two applications have been attracting extensive attention from both the industry and academia.

Table 2.2: Categorization of healthcare big data

Category	Type	Description	Sources
Research	Clinical trails	Design parameters and results of clinical trials	Medical organizations
	Medical apparatus	Design and operational guidance of medical apparatus	Manufacturers
Clinical	Electronic health records	A systematized collection of patient and population electronically-stored health information, e.g., demographics, medication, medical history, and immunization status	Hospitals & clinics
	Medical diagnosis	Results of various diagnostic tests, e.g., tissue samples, blood samples, and imaging scans	Hospitals & medical centers
	Biomarkers	Molecular data, e.g., genome, transcriptome, proteome, and metabolome	Hospitals & biotechnology companies
	Administrative data	Admission, discharge, transfer, payment, etc.	Hospitals & clinics
Claims	Medical claims	Reimbursement data from medical claims, e.g., procedures, medications, and hospitalization	Hospitals & payers
	Prescription claims	Reimbursement data from prescription claims, e.g, drug, dose, and duration	Hospitals & Pharmacies
Individuals	Social media data	Textual data from various social medias and communities, e.g., Sermo and Facebook	Social media companies
	IoT data	Environmental data and lifestyle data from various IoT devices, e.g., smart watch and smart sphygmomanometer	Individuals

2.4.1 Big Data Sharing for Healthcare

Healthcare has always been important to the society. Illness, accidents, and emergencies do arise every day, and the incurred ailments and diseases are supposed to be diagnosed, treated, and managed. There are a significant amount of invaluable health data generated everyday, e.g., the electronic health records (EHRs) in hospitals, clinical trials in medical research organizations, and daily health data of individuals from smart homes. Table 2.2 gives a categorization of the healthcare big data. Big data sharing can make these data to be interoperable and benefits healthcare a lot. In 2020, the U.S. National Institutes of Health, the largest global funder of biomedical research, is finalizing a supportive policy for healthcare data sharing [146].

First, big data sharing can enhance the understanding of each individual hospitalized case. When a hospital receives a new patient especially with some unusual symptoms, it is important for the doctors to refer to the past cases with similar circumstances. Moreover, the historical EHRs of the patient is also useful for disease diagnose. Big data sharing, which is termed as healthcare information exchange (HIE) in the medical field, has been popular for acquisition of EHRs by one hospital from another [79]. Besides, the exchange of EHRs helps the doctors to have a better understanding of different cases and diseases.

Furthermore, big data sharing benefits the discovery of scientific insights, e.g., disease prediction and therapeutic regimen, through aggregating and analyzing clinical trials. In recent years, machine learning-based medical data analysis has been attracting extensive attentions from the governments, industry, and academia [106]. The machine learning approaches demand a large number of data as input, which can be hardly provided by a single medical organization. As a result, aggregation of the clinical trials is important, which can be achieved by big data sharing.

Finally, sharing smart home data makes significant contributions to precision medicine. With the development of IoT technology, smart IoT devices for healthcare becomes

common in daily life of the human beings, e.g., smart watches to monitor the heart rates and smart sphygmomanometers to measure the blood pressure. The numerous personal health data is useful for precision medicine [33]. That is, the doctor takes the individual variability in environment and lifestyle into consideration when conducting disease treatment or giving prevention advice.

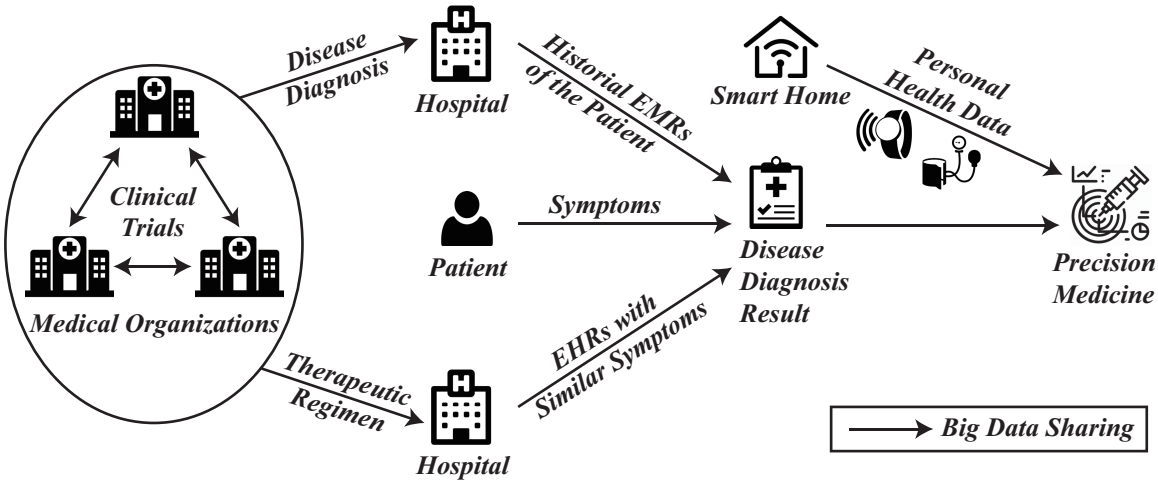

Figure 2.9: Big data sharing for healthcare

Figure 2.9 demonstrates an example case study of the application of big data sharing in healthcare. In particular, the clinical trials are shared among the medical organizations for a better investigation of the existing diseases together with the novel therapeutic regimens. Moreover, the hospitals diagnose the diseases of the patients based on the historical EHRs of the patients, the symptoms, and the EHRs with similar symptoms. Finally, precision medicine can be applied to the patients with the help of the personal health data generated from smart home.

2.4.2 Big Data Trading

In recent years, big data trading has become an emerging business model because of the great value and high demands for big data. Both the industry and academia have

been investigating the possible approaches for big data trading. In big data trading, the commodity is special, i.e., big data, and the traders are large-scale enterprises and organizations in general.

From the perspective of the industry, Xignite[8] targets for a convenient market for trading big data from the financial institutions and Fintech (financial technology) firms, Gnip.[9] provides an aggregation of the social media APIs for the users to collect big data, Amazon releases the product AWS Data Exchange as mentioned in Section 2.2.1 to link the big data providers and users. In China, a bunch of big data trading platforms were established with the support from the government e.g., Guiyang Global Big Data Exchange (GBDEx)[10] and Shanghai Data Exchange Corp. (SDE)[11] were established in 2014 and 2016, respectively.

Besides the various commercial platforms from the industry, there are an increasing number of research works that tackles the challenges of developing big data trading platforms. Zheng et al. considers the balance between online data pricing and reward sharing and presents the first architecture of mobile crowd-sensed big data market [188]. Oh et al. proposes proposes a competitive big data trading model consisting of multiple data providers who weigh the value between privacy protection and valuation [124]. Zhao et al. studies how the variety of big data affects the behaviors of the content providers and Internet service providers under the sponsored data plan [186].

2.5 Chapter Summary

Big data sharing, which refers to refers to the act of the data sharers to share big data so that the sharees can find, access, and use in the agreed ways, is the key

[8]Xignite: We Make Market Data Easy

[9]Gnip Homepage

[10]GBDEx: Global Big Data Exchange

[11]Shanghai Data Exchange Corp.

for transition from data islands to data ecosystem and articulating the value of big data. This chapter presents the first comprehensive survey of big data sharing. We explain the basics of big data sharing, i.e., definition, general procedures, benefits, and requirements. Moreover, we study the existing big data sharing platforms with categorization based on the system architecture. Moreover, the challenging issues are identified with explanation of the possible solutions. Finally, we present the representative applications of big data sharing. We believe this survey will give the audiences a full view of big data sharing and stimulate future research.

Chapter 3

Fairness-based Transaction Packing

Generally, a blockchain is an append-only list of blocks, each of which includes a set of transactions, managed by a peer-to-peer network adhering to a protocol for inter-node communication and validating new blocks [66]. The magic of blockchain lies in the protocol of validating new blocks, i.e., consensus mechanism. In permissioned blockchains, the consensus mechanism is performed round by round, and each round consists of three phases, i.e., leader election, transaction packing, block propagation. To begin a round, all the blockchain nodes run the same leader election algorithm to elect a transaction packer. Then, the packer selects several transactions from its memory pool and pack them into a block. Finally, the generated block is broadcast to the network, and the transactions in the block get confirmed.

Blockchain can be regarded as a service for data storage. In particular, the users send the data, or transactions, to the blockchain, and receive the operational results, i.e., acceptance or rejection. To improve the quality of service, the research communities have been attempting to propose more efficient [57] and reliable [108] algorithms for leader election. Since the throughput of blockchain is limited, the blockchain service is supposed to be fairly shared among multiple users. In permissionless blockchain, the users pay fees, in forms of native cryptocurrencies, for their transactions. The

nodes strategically pack those transactions with high transaction fees into a block to earn more monetary rewards. The fairness among the users is naturally achieved because the transactions with higher transaction fees tend to be served first.

The fairness in permissioned blockchain differs due to the lack of native cryptocurrencies and transaction fees. In this chapter, we consider fairness in permissioned blockchain from the perspective of transaction response time. For each transaction, its response time is the duration from its submission to the time when a block containing this transaction is confirmed. A permissioned blockchain is considered to be fair if the response times of the transactions are close to each other. That is, the transactions which are submitted first are expected to be packed into blocks first. Unfairness leads to a high deviation of the transaction response times. As a result, the transactions incurring long delays will suffer from undesirable quality of experience. More seriously, some transactions get timeouts and discarded if their response times exceed a certain period. In time-sensitive applications, e.g., energy trading [99] and manufacturing operation [98], the discarded transactions can lead to income loss and even safety issues.

Little attention has been paid to the fairness issue in permissioned blockchain although it is vital. Transaction packing is the key to enhance the fairness because it directly decides which transactions are packed into blocks. The first idea is possibly first-come-first-serve (FCFS) [143]. In permissioned blockchain, the FCFS strategy is to select the transactions with long waiting times and pack them into a block. However, the selected subset of transactions can be invalid to be packed into a block. For example, one transaction cannot be packed if its dependent transactions are not confirmed yet. On this circumstance, the next choice is supposed be generated by the transaction packing algorithm while FCFS strategy fails. In existing permissioned blockchains, transactions are arbitrarily packed into blocks. An intuitive approach is to choose subsets one after another randomly. This approach cannot achieve preferable fairness since transactions with long waiting times are not considered first. In

conclusion, traditional FCFS strategy fails to continuously generate subsets of transactions, and the random strategy cannot achieve satisfactory fairness.

In this chapter, we propose Fair-Pack, a fair transaction packing algorithm for permissioned blockchains. First, we quantify the fairness according to Jain's fairness index [74] of response times, and define the fairness problem in permissioned blockchains formally. Then, we gain the insight that fairness is positively related to the sum of waiting times of the selected transactions. In this way, we transform the fairness problem into the subset sum problem, which is to find a valid subset with subset sum as large as possible. However, the number of subsets of a given set is exponential, which makes it non-trivial to solve the subset sum problem. We divide the subset sum problem into two individual problems depending on the relationship between the maximum size of the subset and the size of the given set. Furthermore, we figure out the partial/global orders of the subsets according to the subset sum, and propose a heuristic algorithm and a min-heap-based algorithm to solve the two problems separately. Finally, we analyze the time complexity of Fair-Pack and extensively evaluate its performance in terms of fairness and average response time. Based on the experiments, we conclude that Fair-Pack not only achieves better fairness, but reduces the average response time as well. The main contributions of this chapter are as follows:

- We define the fairness problem in permissioned blockchain. We propose an overall transaction packing algorithm Fair-Pack and transform it into two subset sum problems via theoretical analysis. To the best of our knowledge, this is the first work on the transaction fairness problem in permissioned blockchain.
- Inside Fair-Pack, we propose a heuristic and a min-heap-based optimal algorithm to solve the two subset sum problems separately. The performance and time complexity of the two algorithms are formally analyzed.
- We carry out extensive experiments on how the performance of Fair-Pack is

influenced by the transaction incoming rate, block generation time, block size, and block validity ratio. The results indicate that Fair-Pack achieves better fairness and less average response time compared to the existing works.

The rest of this chapter is organized as follows. Section 3.1 introduces the system model and defines the fairness problem in permissioned blockchains. In Section 3.2, we present the overall algorithm Fair-Pack, and transform the original fairness problem into two subset sum problems SM-Sum and LM-Sum. Then, we propose a heuristic algorithm Sum-Index to solve SM-Sum in section 3.3 and a min-heap-based optimal algorithm Min-Heap-Op to solve LM-Sum in section 3.4. Furthermore, we analyze the time complexities of the algorithms in section 3.5 and conduct extensive experiments to evaluate the performance of Fair-Pack in section 3.6. The related work is discussed in section 3.7. Finally, section 3.8 concludes this chapter.

3.1 System Model and Problem Statement

In this section, we first introduce the system model of permissioned blockchain. Then, we define the fairness problem in permissioned blockchains with concrete explanations of the input, assumptions, and objective.

The block generation in a permissioned blockchain proceeds round by round as shown in Figure 3.1. At the beginning of each round, the blockchain network runs the leader election algorithm to elect a leader, node i. Then, node i invokes transaction packing algorithm to select a subset of transactions from its local memory pool and pack them into block i. Finally, block i is propagated in the blockchain network through broadcasting. From the perspective of the users, they submit their transactions to a random node in the blockchain network. Upon receiving the transactions, the node stores them in the local memory pool and broadcast the transactions to other nodes. Because broadcasting incurs network delay, the memory pools for different nodes may

be different.

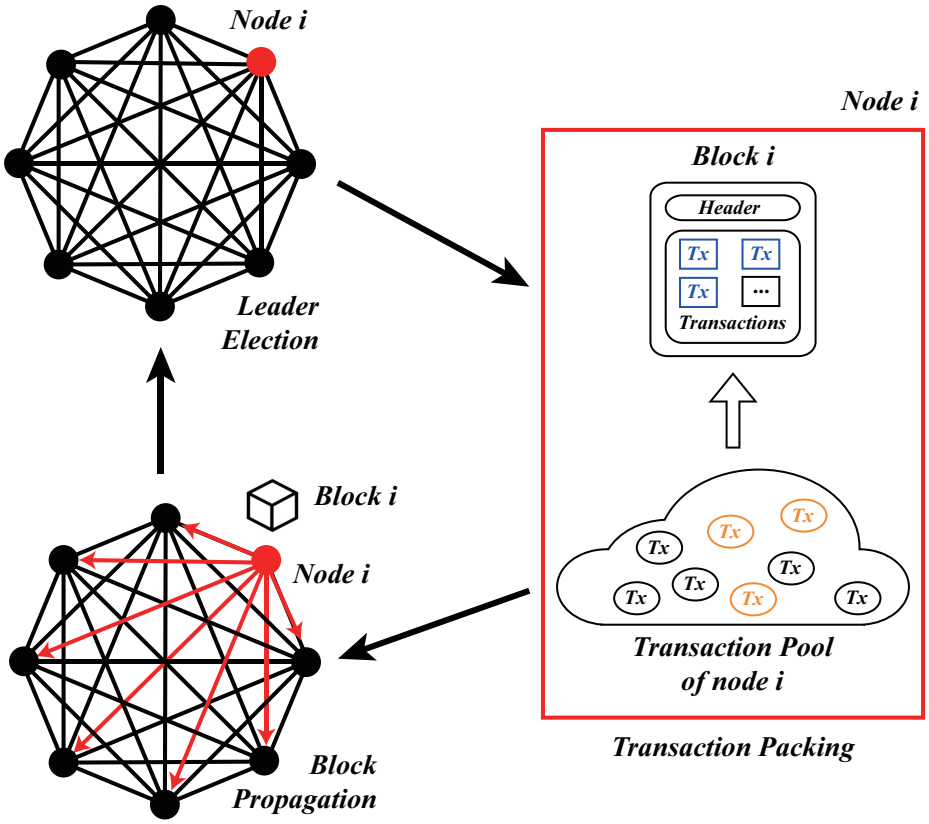

Figure 3.1: Block generation of permissioned blockchains

The *waiting time* and *response time* of a transaction are defined as follows.

Definition 1. *Suppose a transaction x_i is submitted to blockchain network at time s_i and x_i is in memory pool at the current time t_c, then the waiting time of x_i is $a_i = t_c - s_i$.*

Definition 2. *Suppose a transaction x_i is submitted to the blockchain network and packed into blocks at time s_i and e_i, respectively, then the response time of x_i is $t_i = e_i - s_i$.*

As the permissioned blockchain runs round by round, there are more and more transactions packed into blocks. This chapter studies the fairness according to Jain's

fairness index [74]. Note that the fairness index is between 0 (exclusive) and 1 (inclusive). A larger fairness index means better fairness and the fairness index equals to 1 if the response times of all the transactions are the same.

Definition 3. *Suppose there are n transactions* $\mathcal{X} = \{x_1, \cdots, x_n\}$ *packed into blocks with response times* $t_1, \cdots, t_n$*, then the fairness among the n transactions is defined as:* $\mathcal{J}(\mathcal{X}) = \frac{(\Sigma_{i=1}^{n} t_i)^2}{n \cdot \Sigma_{i=1}^{n} t_i^2}$.

To maximize the overall fairness, we should consider not only the response times of the transactions in blocks but also the waiting times of the transactions in memory pool. However, the number of transactions in blocks increases infinitely as the permissioned block runs, which hinders the development of a time-efficient transaction packing algorithm. In this chapter, we only consider the waiting times of the transactions in memory pool in a single round and the expected fairness of a given packing strategy.

Definition 4. *Suppose there are n transactions* $\mathcal{X}$*, the maximum number of transactions in a block is k, and the time to make a packed block to be committed is r. Consider a packing strategy which packs a subset* $\mathcal{X}'$ *of transactions into a block in a round. Suppose the waiting times of* $\mathcal{X}'$ *in the round are* $s_1, \cdots, s_l$ *and the waiting times of the remaining transactions* $\mathcal{X} \setminus \mathcal{X}'$ *are* $t_1, \cdots, t_m$*, where* $l + m = n$*,* $l < k$*, and* $t_1 \geq \cdots \geq t_m$*. Then, the expected fairness of the packing strategy in the round is defined to be:*

$$\mathcal{J}(\mathcal{X}) = \frac{(\Sigma_{i=1}^{l}(s_i + r) + \Sigma_{i=1}^{m}(t_i + r + \lceil \frac{i}{k} \rceil \cdot r))^2}{n \cdot (\Sigma_{i=1}^{l}(s_i + r)^2 + \Sigma_{i=1}^{m}(t_i + r + \lceil \frac{i}{k} \rceil \cdot r)^2)} \tag{3.1}$$

In short, the expected fairness assumes that the remaining transactions in the memory pool are packed into blocks in FCFS order. To this end, we aim at finding a packing strategy with the maximum expected fairness in each round, which is formally defined as follows:

Definition 5. *Problem Ori-Fair: Given 1) a set of n transactions* $\mathcal{X}$ *in memory pool at time* t_c *with submission times* $s_1, \cdots, s_n$*, respectively and 2) the maximum number*

k of transactions that can be packed into a block, assuming 1) a leader is already elected for transaction packing, 2) a subset of $\mathcal{X}$ can be valid or invalid to be packed into a block, and 3) the validity of all subsets of $\mathcal{X}$ are unknown before generation, we aim to develop a transaction packing strategy to continuously generate subsets of $\mathcal{X}$ until a valid subset is generated with the expected fairness $\mathcal{J}(\mathcal{X})$ as large as possible.

In the problem, the maximum number of transactions that can be packed into a block is given as k. The reason why k is bounded is that a large value of k leads to high network congestion when the block is propagated in the network. For example, the value of k in Bitcoin is around $3,000$ due to the limit of block size and average transaction size. In this chapter, we consider k as an adjustable parameter.

In the following, we explain the reasonability of the assumptions. First, leader election and transaction packing are conducted in sequence. As a result, we can use existing leader election methods, such as [89] and [48], to select a transaction packer to fit assumption 1). Second, a subset of transactions can be invalid to be packed into a block for various reasons, e.g., one transaction cannot be packed if its dependent transactions are not confirmed yet. Finally, we assume the validities of all subsets of transactions are unknown before the generation to separate transaction packing with block verification. The separation makes the transaction packing algorithm to be an independent component, which makes the blockchain system more modularized.

Our target is to develop a fair transaction packing algorithm. The algorithm is supposed to continuously generate subsets of transactions because a subset of transactions can be invalid and its validity is unknown in advance.

3.2 Fair-Pack: a Fairness-based Transaction Packing Algorithm

In this section, we prove that the fairness index is positively related to the sum of waiting times of the packed transactions in Ori-Fair. Then, the proved property is used to transform Ori-Fair into the subset sum problem. Finally, we propose an overall solution Fair-Pack towards solving Ori-Fair.

Theorem 1. *Given a set of n transactions $x_1, \cdots, x_n$ in pool with waiting times $a_1, \cdots, a_n$, respectively and k transactions are supposed to be packed, in each round, the larger the sum of the waiting times of the packed transactions, the larger the fairness of the packing strategy.*

Proof. Consider two permutations σ and τ of $(1, \cdots, n)$, where $\sigma = (\sigma_1, \cdots, \sigma_n)$ and $\tau = (\tau_1, \cdots, \tau_n)$. The two packing strategies σ-Pack and τ-Pack pack transactions in the order of $(x_{\sigma_1}, \cdots, x_{\sigma_n})$ and $(x_{\tau_1}, \cdots, x_{\tau_n})$, respectively.

Assume by contradictory that σ-Pack packs transactions with larger sum of waiting times in each round while τ-Pack achieves larger fairness. By definition, we have the following properties:

$$\Sigma_{i=1}^{k} a_{\sigma_i} > \Sigma_{i=1}^{k} a_{\tau_i} \tag{3.2}$$

$$\forall 2 \leq j < \lceil \frac{n}{k} \rceil, \Sigma_{i=1}^{jk} a_{\sigma_i} \geq \Sigma_{i=1}^{jk} a_{\tau_i} \tag{3.3}$$

$$\Sigma_{i=1}^{n} a_{\sigma_i} = \Sigma_{i=1}^{n} a_{\tau_i} \tag{3.4}$$

Notate the time to commit a packed block as t_p. Then the transaction response times using σ-Pack are $a_{\sigma_1} + t_p, a_{\sigma_2} + t_p, \cdots, a_{\sigma_{k+1}} + 2t_p, \cdots, a_{\sigma_n} + \lceil \frac{n}{k} \rceil \cdot t_p$. The transaction response times using τ-Pack are $a_{\tau_1} + t_p, a_{\tau_2} + t_p, \cdots, a_{\tau_{k+1}} + 2t_p, \cdots, a_{\tau_n} + \lceil \frac{n}{k} \rceil \cdot t_p$. To this end, the fairness of σ-Pack and τ-Pack and their relationship are as follows:

$$\mathcal{J}_{\sigma-Pack} = \frac{(\Sigma_{i=1}^{n}(a_{\sigma_i} + \lceil \frac{i}{k} \rceil \cdot t_p))^2}{n \cdot \Sigma_{i=1}^{n}(a_{\sigma_i} + \lceil \frac{i}{k} \rceil \cdot t_p)^2} \tag{3.5}$$

$$\mathcal{J}_{\tau-Pack} = \frac{(\Sigma_{i=1}^{n}(a_{\tau_i} + \lceil \frac{i}{k} \rceil \cdot t_p))^2}{n \cdot \Sigma_{i=1}^{n}(a_{\tau_i} + \lceil \frac{i}{k} \rceil \cdot t_p)^2} \tag{3.6}$$

$$\mathcal{J}_{\tau-Pack} > \mathcal{J}_{\sigma-Pack} \tag{3.7}$$

Since the algorithms are running on the same set of transactions, we have

$$\Sigma_{i=1}^{n} a_{\sigma_i}^2 = \Sigma_{i=1}^{n} a_{\tau_i}^2 \tag{3.8}$$

$$\Sigma_{i=1}^{n}(a_{\sigma_i} + \lceil \frac{i}{k} \rceil \cdot t_p) = \Sigma_{i=1}^{n}(a_{\tau_i} + \lceil \frac{i}{k} \rceil \cdot t_p) \tag{3.9}$$

According to Equation 3.5,3.6,3.7,3.9, we have:

$$\Sigma_{i=1}^{n}(a_{\sigma_i} + \lceil \frac{i}{k} \rceil \cdot t_p)^2 > n \cdot \Sigma_{i=1}^{n}(a_{\tau_i} + \lceil \frac{i}{k} \rceil \cdot t_p)^2 \tag{3.10}$$

Expand Equation 3.10, we get:

$$\begin{aligned} &\Sigma_{i=1}^{n} a_{\sigma_i}^2 + \Sigma_{i=1}^{n}(\lceil \frac{i}{k} \rceil \cdot t_p)^2 + 2\Sigma_{i=1}^{n}(a_{\sigma_i} \cdot \lceil \frac{i}{k} \rceil \cdot t_p) > \\ &\Sigma_{i=1}^{n} a_{\tau_i}^2 + \Sigma_{i=1}^{n}(\lceil \frac{i}{k} \rceil \cdot t_p)^2 + 2\Sigma_{i=1}^{n}(a_{\tau_i} \cdot \lceil \frac{i}{k} \rceil \cdot t_p) \end{aligned} \tag{3.11}$$

According to Equation 3.8,3.11, we have:

$$\Sigma_{i=1}^{n}(a_{\sigma_i} \cdot \lceil \frac{i}{k} \rceil) > \Sigma_{i=1}^{n}(a_{\tau_i} \cdot \lceil \frac{i}{k} \rceil) \tag{3.12}$$

Adding Equation 3.2 and all the inequations in Equation 3.3, we have

$$\Sigma_{i=1}^{\lceil \frac{n}{k} \rceil - 1} \Sigma_{j=1}^{ik} a_{\sigma_j} > \Sigma_{i=1}^{\lceil \frac{n}{k} \rceil - 1} \Sigma_{j=1}^{ik} a_{\tau_j} \tag{3.13}$$

Adding the inequations in Equation 3.12 and Equation 3.13, we have:

$$\begin{aligned} \lceil \frac{n}{k} \rceil \Sigma_{i=1}^{n} a_{\sigma_i} &= \Sigma_{i=1}^{\lceil \frac{n}{k} \rceil - 1} \Sigma_{j=1}^{ik} a_{\sigma_j} + \Sigma_{i=1}^{n}(a_{\sigma_i} \cdot \lceil \frac{i}{k} \rceil) \\ &> \Sigma_{i=1}^{\lceil \frac{n}{k} \rceil - 1} \Sigma_{j=1}^{ik} a_{\tau_j} + \Sigma_{i=1}^{n}(a_{\tau_i} \cdot \lceil \frac{i}{k} \rceil) \\ &= \lceil \frac{n}{k} \rceil \Sigma_{i=1}^{n} a_{\tau_i} \end{aligned} \tag{3.14}$$

It is clear that Equation 3.14 is contradictory to Equation 3.4. As a result, the assumption does not hold and σ-Pack achieves larger fairness than τ-Pack. □

According to Theorem 1, it achieves better fairness to pack transactions with the larger sum of waiting times. Therefore, the best strategy is to pack transactions with top-k waiting times, which is FCFS. However, such a transaction subset can be invalid and we need to continuously generate transaction subsets. According to Theorem 1, the sum of waiting times can be treated as the heuristic to generate transaction subsets. That is, we are supposed to find the transaction subset with the 1-st, 2-nd, $\cdots$, m-th largest sum of waiting times.

Because the transaction waiting times are known real numbers, the problem is to find a subset of no more than k elements from a set of n real numbers with the m-th largest subset sum among all the feasible subsets. Here, the feasibility means the subsets contain no more than k elements. If k is smaller than n, there are $\Sigma_{i=0}^{k}\binom{n}{i}$ feasible subsets. Similarly, there are 2^n ones if k is no smaller than n. In this chapter, we consider the two conditions separately with problem statements as follows.

Definition 6. *Problem SM-Sum: Given a set of n positive real numbers $\mathcal{W}$, a positive integer k where $k < n$, and a positive integer m where $m \leq \Sigma_{i=0}^{k}\binom{n}{i}$, there are $\Sigma_{i=0}^{k}\binom{n}{i}$ distinct subsets of $\mathcal{W}$ of size no larger than k. Among the subsets, find the one with the m-th largest sum.*

Definition 7. *Problem LM-Sum: Given a set of n positive real numbers $\mathcal{W}$, a positive integer k where $k \geq n$, and a positive integer m where $m \leq 2^n$, there are 2^n distinct subsets of $\mathcal{W}$ of size no larger than k. Among the subsets, find the one with the m-th largest sum.*

If the problems SM-Sum and LM-Sum are solved, then the problem Ori-Fair can be solved by Algorithm 1.

In Algorithm 1, we assume that the problem SM-Sum and LM-Sum are solved by Sum-Index and Min-Heap-Op, respectively. First, we compute the transaction waiting times based on the submission times and the current timestamp. Then, the transactions are sorted according to the waiting times in a non-increasing order. That

Algorithm 1 FAIR-PACK: a fairness-based transaction packing algorithm for problem ORI-FAIR

Input: n: the memory pool size; $\mathcal{X} = \{x_1, \cdots, x_n\}$: the transactions in memory pool; $\mathcal{S} = \{s_1, \cdots, s_n\}$: the transaction submission times; k: the maximum number of transactions in a block; t_c: the current timestamp; IS-VALID($\mathcal{U}$): a procedure to check the validity of transaction subset $\mathcal{U}$; SUM-INDEX($\mathcal{W}, n, k, m$): a procedure to solve SM-SUM; MIN-HEAP-OP($\mathcal{W}, n, k, m$): a procedure to solve LM-SUM

Output: a valid transaction subset of $\mathcal{X}$ or NIL in case that all subsets are invalid

1: $\mathcal{W} \leftarrow t_c - \mathcal{S}$
2: Sort $\mathcal{X}$ with respect to $\mathcal{W}$ in non-increasing order
3: **for** $m \leftarrow 1$ **to** ∞ **do**
4: **if** $k < n$ **then** $id \leftarrow$SUM-INDEX($\mathcal{W}, n, k, m$).MAIN()
5: **else** $id \leftarrow$ MIN-HEAP-OP($\mathcal{W}, n, k, m$).MAIN()
6: **end if**
7: **if** $id =$ NIL **return** NIL **end if**
8: $\mathcal{U} \leftarrow x_{id}$
9: **if** IS-VALID($\mathcal{U}$) **return** $\mathcal{U}$ **end if**
10: **end for**

is, w_i will be no smaller than w_j if $i < j$ after sorting. In the for-loop, we transform the problem of finding the transaction subset with the 1-st, 2-nd, $\cdots$, m-th largest sum of waiting times to SM-Sum or LM-Sum depending on the relationship between k and n. Both the procedures of Sum-Index and Min-Heap-Op will return the indexes of the selected elements. Finally, we derive the transaction subset based on the index set as the output of Fair-Pack.

3.3 Sum-Index: a Heuristic Solution to SM-Sum

In this section, we focus on solving SM-Sum. In particular, we propose to use directed acyclic graph (DAG) $\mathcal{G}$ to represent all the $\Sigma_{i=0}^{k}\binom{n}{i}$ subsets. In $\mathcal{G}$, we prove a partial order among the subsets with respect to the subset sum, which leads to the heuristic to enumerate the subsets according to the index sum. That is, the subset sum is related to the number of elements and index sum. Such a heuristic is leveraged in algorithm Sum-Index to solve SM-Sum. To begin with, we define the terminologies of *set sum* and *index sum* as follows.

Definition 8. *Given a set of n real numbers $\mathcal{W}$, the set sum of $\mathcal{W}$ is defined to be the sum of all the elements in $\mathcal{W}$, i.e., $\mathcal{E}(\mathcal{W}) = \Sigma_{i=1}^{n} w_i$.*

Definition 9. *Given a set of n positive real numbers $\mathcal{W} = \{w_1, \cdots, w_n\}$ and a subset $\mathcal{I} = \{w_{\sigma_1}, \cdots, w_{\sigma_p}\}$ of $\mathcal{W}$, the index sum of $\mathcal{I}$ is defined to be the sum of the indexes of its corresponding elements in $\mathcal{W}$, i.e., $\mathcal{D}(\mathcal{I}) = \Sigma_{i=1}^{p} \sigma_p$.*

There are $\Sigma_{i=0}^{k}\binom{n}{i}$ nodes in $\mathcal{G}$, where each node represents a subset. Moreover, $\mathcal{G}$ consists of $k+1$ connected components, where the p-th component is the collection of subsets of size $p-1$. The number of subsets of size i, i.e., $\binom{n}{p}$, is exactly the size of the p-th component. Furthermore, the subset in the i-th component tends to have a smaller subset sum than the j-th component if $i < j$ because of the essential

difference in subset sizes, which indicates a partial order among the components concerning subset sum. Next, we introduce the directed edges in each component.

In the p-th component of $\mathcal{G}$, all the subsets of size p are listed level by level according to the index sum as shown in Figure 3.2. In the figure, the "greater" operator between two subsets represents the relationship of subset sum between them. We find that the subset with a smaller index sum is likely to have a larger set sum. In particular, given a subset $\mathcal{I}$ whose index sum is not the smallest, there must be another subset $\mathcal{I}'$ with smaller index sum and larger subset sum, which is proved in Theorem 2. Moreover, we add a directed edge from the node representing $\mathcal{I}'$ to the node representing $\mathcal{I}$. In this way, all the nodes are (weakly) connected in the p-th components. Because the edges are always from the upper level, i.e., smaller index sum, to the lower level, i.e., larger index sum, the p-th components is a directed and acyclic.

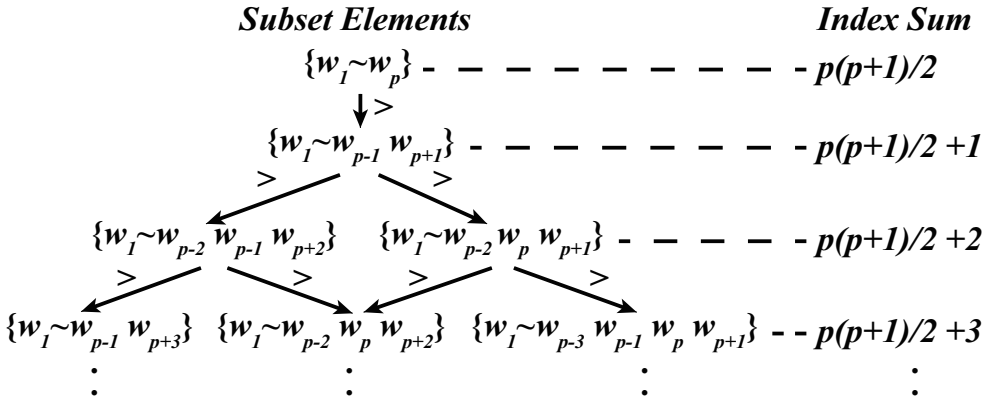

Figure 3.2: Subsets of size p listed level by level according to index sum

Theorem 2. *Given a set of n positive real numbers $\mathcal{W} = \{w_1, \cdots, w_n\}$ where $w_1 \geq \cdots \geq w_n$ and a subset $\mathcal{I}$ of $\mathcal{W}$ where $|\mathcal{I}| = p$, $\mathcal{D}(\mathcal{I}) = d$, and $d \neq p(p+1)/2$, there exists at least one subset $\mathcal{I}'$ of $\mathcal{W}$ such that $|\mathcal{I}'| = p$, $\mathcal{D}(\mathcal{I}') = d-1$, and $\mathcal{E}(\mathcal{I}') \geq \mathcal{E}(\mathcal{I})$.*

Proof. Let $\mathcal{I} = \{w_{\sigma_1}, \cdots, w_{\sigma_p}\}$ where $\Sigma_{i=1}^{p}\sigma_p = d$ and $\sigma_1 < \cdots < \sigma_p$. Assume, for the sake of contradiction, that for all i, $w_{\sigma_i - 1} \in \mathcal{I}$ if $\sigma_i > 1$ (C1). Assume, for the sake of contradiction, that $\sigma_1 > 1$ (C2). We have $w_{\sigma_1 - 1} \in \mathcal{I}$ according to assumption C1.

However, w_{σ_1} is the largest element in $\mathcal{I}$. Therefore, C2 does not hold and $\sigma_1 = 1$. Similarly, we can get $\sigma_i = i$ for every $1 \leq i \leq p$. Therefore, $d = \Sigma_{i=1}^{p} \sigma_i = p(p+1)/2$, which contradicts the condition that $d \neq p(p+1)/2$. Hence, C1 does not hold and there exists at least one i such that $\sigma_i > 1$ and $w_{\sigma_i - 1} \notin \mathcal{I}$.

Let $\sigma_t > 1$ and $w_{\sigma_t - 1} \notin \mathcal{I}$. We construct $\mathcal{I}' = \mathcal{I} \setminus \{w_{\sigma_t}\} \cup \{w_{\sigma_t - 1}\}$ satisfying the conditions in the theorem as follow.

- $\mathcal{I}' \subset \mathcal{W}$ since $\mathcal{I} \subset \mathcal{W}$ and $\{w_{\sigma_t - 1}\} \in \mathcal{W}$.
- $|\mathcal{I}'| = |\mathcal{I}| - 1 + 1 = p$.
- $\mathcal{D}(\mathcal{I}') = \mathcal{D}(\mathcal{I}) - \sigma_t + (\sigma_t - 1) = d - 1$.
- $\mathcal{E}(\mathcal{I}') \geq \mathcal{E}(\mathcal{I})$ since $\mathcal{E}(\mathcal{I}') - \mathcal{E}(\mathcal{I}) = w_{\sigma_t - 1} - w_{\sigma_t} \geq 0$.

□

We apply the above construction to all the $k+1$ components in $\mathcal{G}$, resulting in $\mathcal{G}$ to be a DAG. The partial orders among the subsets is reveal in $\mathcal{G}$, which gives birth to a heuristic algorithm Sum-Index to solving SM-Sum as shown in Algorithm 2.

In Sum-Index, we do not order all the subsets to find the subset with exactly the m-th largest subset sum since the time complexity, i.e., $O(\Sigma_{i=0}^{k} \binom{n}{i} \cdot \log(\Sigma_{i=0}^{k} \binom{n}{i}))$, is too high. Instead, we enumerate the subset size in decreasing order, the index sum of the subsets in increasing order, and the subset in lexicographical order sequentially. The algorithm Sum-Index begins with the first subset of size k and index sum $k(k+1)/2$. Such a subset contains exactly the k largest elements in $\mathcal{W}$. To find the next subset, procedure Next-Subset first tries to invoke the procedure Next-PD to find a subset with the same subset size and index sum. In detail, Next-PD takes a subset r as input and outputs the next subset of r in lexicographical order. If such a subset is not found, Next-Subset increases the desired index sum and invokes procedure First-PD to find the first subset in lexicographical order. Finally, if the

Algorithm 2 SUM-INDEX: a heuristic algorithm for SM-SUM

Input: $\mathcal{W}$: a set of n positive real numbers; k: a positive integer that $k < n$; m: a positive integer that $m \leq \Sigma_{i=0}^{k}\binom{n}{i}$

Output: the indexes of the subset of $\mathcal{W}$ with approximately the m-th largest subset sum among all the $\Sigma_{i=0}^{k}\binom{n}{i}$ subsets

```
1: procedure MAIN( )
2:     if m = 1 return FIRST-SUBSET( ) end if
3:     return NEXT-SUBSET( )
4: end procedure
5: procedure FIRST-SUBSET( )
6:     global gp ← k                          ▷ "global" variable for all procedures
7:     global gd ← gp(gp+1)/2
8:     return FIRST-PD(gp, gd)
9: end procedure
10: procedure NEXT-SUBSET( )
11:    return NEXT-PD(gp, gd) if not NIL
12:    gd ← gd + 1
13:    return FIRST-PD(gp, gd) if not NIL
14:    (gp, gd) ← (gp − 1, gp(gp+1)/2)
15:    if gp = 0 return NIL end if
16:    return FIRST-PD(gp, gd)
17: end procedure
```

18: **procedure** FIRST-PD(p, d)
19: **if** $d < \frac{(p+1)p}{2}$ **or** $d > \frac{(n-p+1+n)p}{2}$ **then return** NIL
20: **end if**
21: $r \leftarrow$ an array of size p in which all elements are 0
22: **for** $i \leftarrow 1$ **to** p **do**
23: $r_i \leftarrow \max(r_{i-1} + 1, d - \frac{(n-p+i+1+n)(p-i)}{2})$
24: $d \leftarrow d - r_i$
25: **end for**
26: **return** r
27: **end procedure**
28: **procedure** NEXT-PD(p, d, r)
29: $s \leftarrow$ an array of size p in which all elements are 0
30: **for** $i \leftarrow 1$ **to** p **do** $s_i \leftarrow s_{i-1} + r_i$ **end for**
31: **for** $i \leftarrow p - 1$ **to** 1 **do**
32: **if** $r_i + 1 < r_{i+1}$ and $d - s_i - 1 \geq \frac{(r_i+r_i+p-i+3)(p-i)}{2}$ **and** $d - s_i - 1 \leq \frac{(n+i+1-p+n)(p-i)}{2}$ **then**
33: $(r_i, d) \leftarrow (r_i + 1, d - s_i - 1)$
34: **for** $j \leftarrow i + 1$ to p **do**
35: $r_j \leftarrow \max(r_{j-1} + 1, d - \frac{(n+j+1-p+n)(p-j)}{2})$
36: $d \leftarrow d - r_j$
37: **end for**
38: **return** r
39: **end if**
40: **end for**
41: **return** NIL
42: **end procedure**

index sum exceeds the limit, NEXT-SUBSET will increase the subset size and set the index sum to the smallest one.

3.4 Min-Heap-Op: an Optimal Solution to LM-Sum

In this section, we propose algorithm MIN-HEAP-OP to solve LM-SUM. We build a min-heap to maintain the relationship in terms of subset sums among the 2^n subsets, and the subsets are generated by manipulating the min-heap.

To begin with, we construct a binary tree $\mathcal{H}(n)$ as follows:

- $\mathcal{H}(n).root = \{n\}$
- For each element $e \in \mathcal{H}(n)$ in which $\min(e) \neq 1$, $e.lc = e \backslash \{\min(e)\} \cup \{\min(e)-1\}$
- For each element $e \in \mathcal{H}(n)$ in which $\min(e) \neq 1$, $e.rc = e \cup \{\min(e) - 1\}$

An example of $\mathcal{H}(4)$ is shown in Figure 3.3(a). We see that $\mathcal{H}(4)$ is a complete binary tree which contains and only contains all the subsets of $\{1, 2, 3, 4\}$. Next, we prove it in case of n through Theorem 3 and Theorem 4.

Theorem 3. *The tree $\mathcal{H}(n)$ contains all the non-empty subsets of $U = \{1, 2, \cdots, n\}$.*

Proof. Consider an arbitrary subset I of U. We prove the lemma by induction on the minimum value of I.

Base case. When $\min\{I\} = n$, we can infer that $I = \{n\}$ because $I \subseteq U$ and $\max\{U\} = n$. Therefore, I is an element, in particular, the root of $\mathcal{H}(n)$.

Induction step. Let $1 < k \leq n$ and assume I is an element of $\mathcal{H}(n)$ as long as $\min(I) \geq k$. We aim to prove that I is an element of $\mathcal{H}(n)$ as long as $\min(I) = k - 1$. We consider three circumstances as follows.

- Case 1: $|I| = 1$. We can infer that $I = \{k-1\}$ as $\min(I) = k-1$. Consider another subset $I' = \{k\}$. Because $I' \subseteq U$ and $\min(I') = k$, we know I' is an element of $\mathcal{H}(n)$. $I'.lc = I' \setminus \{k\} \cup \{k-1\} = I$. As a result, I is an element of $\mathcal{H}(n)$ as well.

- Case 2: $|I| \neq 1$ and $\min(I \setminus \{k-1\}) = k$. Consider another subset $I' = I \setminus \{k-1\}$. As $I' \subseteq I \subseteq U$ and $\min(I') = k$, I' is an element of $\mathcal{H}(n)$. $I'.rc = I' \cup \{k-1\} = I$. Therefore, I is also an element of $\mathcal{H}(n)$.

- Case 3: $|I| \neq 1$ and $\min(I \setminus \{k-1\}) \neq k$. Consider another subset $I' = I \setminus \{k-1\} \cup \{k\}$. Because $I \subseteq U$ and $k \in U$, we can infer that $I' \subseteq U$. Furthermore, I' is an element of $\mathcal{H}(n)$ as $\min(I') = k$. $I'.lc = I' \setminus \{k\} \cup \{k-1\} = I$. Hence, I is an element of $\mathcal{H}(n)$ as well.

The above three cases cover all the subsets whose minimum value equals $k-1$. Meanwhile, we show the subsets are elements of $\mathcal{H}(n)$ for all the three cases. Therefore, I is an element of $\mathcal{H}(n)$ as long as $\min(I) = k-1$.

Based on the base case and induction step, we conclude that $I \subseteq U$ is an element of $\mathcal{H}(n)$ as long as $\min(I) \geq 1$, which completes the proof. □

Theorem 4. *The tree $\mathcal{H}(n)$ is a complete binary tree, which contains and only contains all the non-empty subsets of $U = \{1, 2, \cdots, n\}$.*

Proof. Notate the set of elements at level k and its size as $\mathcal{H}(n,k)$ and $|\mathcal{H}(n,k)|$, respectively. In the following, we prove that $|\mathcal{H}(n,k)| \leq 2^{k-1}$ and $\min(e) = n+1-k$ for any $1 \leq k \leq n-1$ and $e \in \mathcal{H}(n,k)$ by induction on the value of k.

Base Case. When $k = 1$, there is only one element $\{n\}$ in the first level of the tree $\mathcal{H}(n)$, i.e., $\mathcal{H}(n,k) = \{\{n\}\}$. We can get that $\mathcal{H}(n,k) = 1 \leq 2^{k-1}$ and $\min(\{n\}) = n = n+1-k$.

Induction Step. Let k be an integer that $1 \leq k < n-1$ and assume that $|\mathcal{H}(n,k)| \leq$

2^{k-1} and $\min(e) = n+1-k$ for any $e \in \mathcal{H}(n,k)$. We aim to prove that $|\mathcal{H}(n,k+1)| \leq 2^k$ and $\min(e) = n-k$ for any $e \in \mathcal{H}(n,k+1)$. We prove the two statements separately as follows.

- There are exactly two children for each element in $\mathcal{H}(n,k)$ according to the definition of the tree $\mathcal{H}(n)$. If there is no overlapping element among all the children of all the elements in $\mathcal{H}(n,k)$, $|\mathcal{H}(n,k+1)|$ will be exactly twice the value of $|\mathcal{H}(n,k)|$, i.e., $|\mathcal{H}(n,k+1)| = 2 \cdot |\mathcal{H}(n,k)| \leq 2 \cdot 2^{k-1} = 2^k$. If any overlapping element, the value of $|\mathcal{H}(n,k+1)|$ will be smaller, i.e., $|\mathcal{H}(n,k+1)| \leq 2^k$. As a result, $|\mathcal{H}(n,k+1)| \leq 2^k$.

- Considering an arbitrary element $e \in \mathcal{H}(n,k+1)$, e must be a child of some element $e' \in \mathcal{H}(n,k)$. We can get $\min(e') = n+1-k$ according to the assumption. If $e'.lc = e$, then $\min(e) = \min(e' \backslash \{\min(e')\} \cup \{\min(e')-1\}) = n-k$. Otherwise, $e'.rc = e$. Under this circumstance, $\min(e) = \min(e' \cup \{\min(e')-1\}) = n-k$. As a result, $\min(e) = n-k$ for any $e \in \mathcal{H}(n,k+1)$.

Based on the base case and the induction step, we draw the conclusion that $|\mathcal{H}(n,k)| \leq 2^{k-1}$ and $\min(e) = n+1-k$ for any $1 \leq k \leq n-1$ and any $e \in \mathcal{H}(n,k)$. Consider $k = n$, we can infer that $\min(e) = 1$ for any $e \in \mathcal{H}(n,n)$. Therefore, all the elements in $\mathcal{H}(n,n)$ have no child, i.e., $\mathcal{H}(n)$ consists of exactly n levels. Therefore, $|\mathcal{H}(n)| = \Sigma_{i=1}^{n} |\mathcal{H}(n,i)| \leq \Sigma_{i=1}^{n} 2^{i-1} = 2^n - 1$. In another word, the size of the tree $\mathcal{H}(n)$ is no more than $2^n - 1$.

The number of non-empty subsets of U is $2^n - 1$. According to Lemma 3, the size of $\mathcal{H}(n)$ is no less than $2^n - 1$, which is the number of non-empty subsets of U. Therefore, the size of $\mathcal{H}(n)$ is exactly $2^n - 1$. Meanwhile, $\mathcal{H}(n)$ contains and only contains all the non-empty subsets of U. Moreover, $\mathcal{H}(n)$ consists of exactly n levels. As a result, $\mathcal{H}(n)$ is a complete binary tree, which completes the proof. □

Based on the binary tree $\mathcal{H}(n)$, we construct a minimum heap $\mathcal{W}_{\mathcal{H}(n)}$ as follows:

- $\mathcal{W}_{\mathcal{H}(n)}.root.value = \mathcal{H}(n).root$
- For each element $e \in \mathcal{W}_{\mathcal{H}(n)}$ in which $e.value.lc \neq$ Nil, $e.lc.value = e.value.lc$
- For each element $e \in \mathcal{W}_{\mathcal{H}(n)}$ in which $e.value.rc \neq$ Nil, $e.rc.value = e.value.rc$
- For each element $e \in \mathcal{W}_{\mathcal{H}(n)}$, $e.key = \mathcal{E}(\mathcal{W}_{e.value})$

Each element in $\mathcal{H}(n)$ represents the indexes of a selected subset of $\mathcal{W}$. For example, Figure 3.3(b) shows the subsets generated according to $\mathcal{H}(n)$ when $n = 4$. In Figure 3.3(b), we can see the subset sum of a parent node is always no less than the one of a child node, which implies $\mathcal{H}(4)$ to be a min-heap. In the following, we formally prove that $\mathcal{H}(n)$ is a min-heap based on the subset sum as shown in Theorem 5.

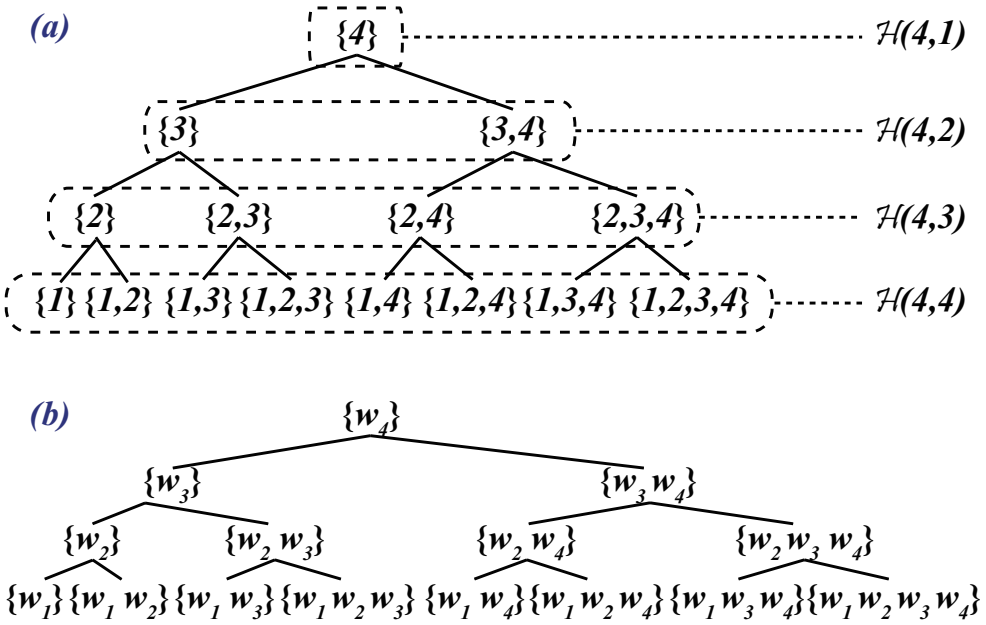

Figure 3.3: (a) $\mathcal{H}(4)$ and (b) $\mathcal{W}_{\mathcal{H}(4)}$

Theorem 5. *$\mathcal{W}_{\mathcal{H}(n)}$ is a binary min-heap.*

Proof. On one hand, $\mathcal{W}_{\mathcal{H}(n)}$ is a complete binary tree since the value field in $\mathcal{W}_{\mathcal{H}(n)}$ is exactly the same with $\mathcal{H}(n)$ while $\mathcal{H}(n)$ is a complete binary tree as proven in

Theorem 4. On the other hand, $\mathcal{W}_{\mathcal{H}(n)}$ satisfies the min-heap property, which is demonstrated as follows.

- For each e in $\mathcal{W}_{\mathcal{H}(n)}$ with left child, i.e., $e.lc \neq \text{NIL}$, the key of $e.lc$ must be no smaller than the key of e.

$$\begin{aligned} & e.lc.key \\ = & \mathcal{E}(\mathcal{W}_{e.lc.value}) = \mathcal{E}(\mathcal{W}_{e.value.lc}) \\ = & \mathcal{E}(\mathcal{W}_{e.value \backslash \{\min(e.value)\} \cup \{\min(e.value)-1\}}) \\ = & \mathcal{E}(\mathcal{W}_{e.value}) - w_{\min(e.value)} + w_{\min(e.value)-1} \\ = & e.key - (w_{\min(e.value)} - w_{min(e.value)-1}) \\ \geq & e.key \end{aligned}$$

- For each e in $\mathcal{W}_{\mathcal{H}(n)}$ with right child, i.e., $e.rc \neq \text{NIL}$, the key of $e.rc$ must be no smaller than the key of e.

$$\begin{aligned} e.rc.key = & \mathcal{E}(\mathcal{W}_{e.value.rc}) \\ = & \mathcal{E}(\mathcal{W}_{e.value \cup \{\min(e.value)-1\}}) \\ = & \mathcal{E}(\mathcal{W}_{e.value}) + w_{\min(e.value)-1} \\ = & e.key + w_{min(e.value)-1} \\ \geq & e.key \end{aligned}$$

To summarize, $\mathcal{W}_{\mathcal{H}(n)}$ is a binary min-heap as it is a complete binary tree and satisfies the min-heap property, which completes the proof. □

Note that min-heap is an efficient data structure to find the k-th minimum element. We leverage the min-heap property to solve LM-SUM as shown in Algorithm 3.

In Algorithm 3, we build a min-heap $\mathcal{W}_{\mathcal{H}(n)}$ to represent all the subsets of $\mathcal{W}$. In case that m equals 1, Algorithm 3 directly returns the whole set $\mathcal{W}$ because $\mathcal{W}$ owns the

Algorithm 3 MIN-HEAP-OP: a min-heap-based algorithm to solve LM-SUM

Input: $\mathcal{W}$: a set of n positive real numbers; k: a positive integer that $k > n$; m: a positive integer that $m \leq 2^n$

Output: the indexes of the subset of $\mathcal{W}$ with the m-th largest subset sum among all the 2^n possible subsets

```
1: procedure MAIN( )
2:     if m = 1 return FIRST-SUBSET( ) end if
3:     return NEXT-SUBSET( )
4: end procedure
5: procedure FIRST-SUBSET( )
6:     global W_{H(n)} ← a min-heap built as stated
7:     return {1, 2, ··· , n}
8: end procedure
9: procedure NEXT-SUBSET( )
10:    r ← DELETE-MIN(W_{H(n)})
11:    if r ≠ ∅ then return {1, 2, ··· , n} \ r.value end if
12:    return NIL
13: end procedure
```

largest subset sum essentially. Otherwise, we applies the DELETE-MIN operation to $\mathcal{W}_{\mathcal{H}(n)}$ to get its root r. The key of r is the smallest subset sum according to the min-heap property. Therefore, we exclude the elements in $r.value$ from the whole set and get the transaction indexes to be selected. DELETE-MIN will also deletes the minimum element, i.e., the root, from the min-heap and maintains the min-heap property. In this way, different subsets can be generated continuously.

3.5 Time Complexity Analysis

In this section, we analyze the time complexity of the algorithms SUM-INDEX, MIN-HEAP-OP, and FAIR-PACK.

Theorem 6. *The time complexity of* SUM-INDEX *is* $O(n)$.

Proof. As shown in Algorithm 2, the main procedure of SUM-INDEX finally calls the procedure FIRST-PD on line 13 or 16, or the procedure NEXT-PD on line 11. Note that the time complexity of SUM-INDEX is irrelevant to the value of m since m is only an indicator for whether the first subset is to be generated.

In terms of FIRST-PD, it contains a for-loop from 1 to p. Because p, as the number of elements in the subset, is of $O(n)$ size, FIRST-PD takes $O(n)$ time.

There are three for-loops in procedure NEXT-PD on line 30, 31, and 34. The first for-loop on line 30 takes $O(n)$ time. The third for-loop on line 34 is inside the second for-loop on line 31 but will be invoked no more than once because there is a RETURN statement right after it. As a result, the second and third for-loops takes $O(n)$ time in total. Overall speaking, NEXT-PD takes $O(n)$ time.

Finally, we conclude that the time complexity of the algorithm SUM-INDEX is $O(n)$. □

Theorem 7. *The time complexity of* MIN-HEAP-OP *is* $O(n)$.

Proof. The major time overhead of Min-Heap-Op lies in the construction of the min-heap $\mathcal{W}_{\mathcal{H}(n)}$ on line 6 and the operation Delete-Min on line 10. In the construction of $\mathcal{W}_{\mathcal{H}(n)}$, we only generate its root and store how the other elements are generated instead of generating all the elements in memory. As a result, it only takes $O(1)$ for the min-heap construction. In terms of Delete-Min, the time complexity should be logarithmic to the size of the heap. As a result, each Delete-Min takes $O(n)$ because the size of $\mathcal{W}_{\mathcal{H}(n)}$ is $2^n - 1$. Note that there are also $O(n)$ key comparisons between any two elements of $\mathcal{W}_{\mathcal{H}(n)}$, in which the subset sums are to be calculated. However, the calculation of subset sum of a child node can be derived from the subset sum of its parent because the difference between them is only one or two elements. To this end, The subset sums of the $O(n)$ subsets can be calculated in $O(n)$ time. In conclusion, the time complexity of the algorithm Min-Heap-Op is $O(n)$. □

Finally, it comes to the time complexity of the algorithm Fair-Pack. As shown in Algorithm 1, Fair-Pack iterates the variable m from 1 to infinity until a valid subset (block) is found. Hence, the running time of Fair-Pack heavily depends on the block validity ratio defined as follows.

Definition 10. *Block Validity Ratio: the possibility for a block to be valid (%).*

Theorem 8. *Supposing the block validity ratio to be α, the algorithm* Fair-Pack *terminates in $\frac{\log(1-\beta)}{\log(1-\alpha)} \cdot O(n)$ with a possibility no less than β.*

Proof. The possibility that Fair-Pack terminates in k rounds is $1-(1-\alpha)^k$, in which each round is a call of Sum-Index or Min-Heap-Op. Hence, we have $(1-(1-\alpha)^k) \geq \beta$, which leads to $k \geq \frac{\log(1-\beta)}{\log(1-\alpha)}$. Each round of Fair-Pack takes $O(n)$ time according to Theorem 6 and Theorem 7. Finally, Fair-Pack terminates in $\frac{\log(1-\beta)}{\log(1-\alpha)} \cdot O(n)$ with a possibility no less than β, which completes the proof. □

For example, assume that the block validity ratio is 0.5%, Fair-Pack will terminate in around $460 \cdot O(n)$, $597 \cdot O(n)$, and $919 \cdot O(n)$ with possibilities of 90%, 95%, and

99%, respectively.

3.6 Performance Evaluation

In this section, we conduct extensive experiments to evaluate the performance of FAIR-PACK.

```
syntax= "proto2";
service Discovery {
    rpc ExchangeNode(Node) returns (Node);
    rpc Hello(Message) returns (Message);
}
service Synchronization{
    rpc BlockFrom(Message) returns (Block);
    rpc BlockTo(Block) returns (Message);
    rpc ExchangeBlock(Block) returns (Block);
    rpc TransactionTo(Transaction) returns (Message);
    rpc TransactionFrom(Message) returns (Transaction);
}
message Transaction{
    required bytes unixtime = 1;
    required bytes body = 2;
    required bytes txhash = 3;
    required int32 type = 4;
    required bytes txfrom = 5;
    optional bytes txto = 6;
}
message Node{
    required int32 number = 1;
    repeated bytes ipport = 2;
}
message Block{
    required uint64 height = 1;
    required bytes unixtime = 2;
    required bytes previoushash = 3;
    required bytes blockhash = 4;
    required bytes difficulty = 5;
    required bytes answer = 6;
    repeated bytes txshash = 7;
    required bytes miner = 8;
    required int32 number = 9;
}
message Message{
    required bytes value = 1;
}
```

Figure 3.4: Protocol buffers of the blockchain prototype

First, we developed a proof-of-concept blockchain prototype using around 1070-line python code based on gRPC. Figure 3.4 shows the communication interfaces among blockchain nodes. In the prototype, two services are implemented to support the blockchain runtime, i.e., peer discovery ("Discovery") and data synchronization ("Synchronization"). The "Discovery" service is used for discovering the nodes inside the blockchain network. When a node is started, it will greet several static nodes (the same as bootnodes in Ethereum) and exchange the connectivity information. The block and transaction synchronization is achieved by the "Synchronization" service, which consists of five remote procedure calls. One thing in particular is that Proof of Work (PoW) serves as the consensus protocol of the blockchain prototype.

Furthermore, three transaction packing algorithms, i.e., FAIR-PACK, FAIR-FIRST [79], and RANDOM-PACK are implemented with around 510-line C++ code. The three packing algorithms are integrated into the blockchain prototype with the help of *ctypes*, using which the packing algorithms are compiled as dynamic link libraries and can be called in python programs.

Finally, we deploy the blockchain prototype together with the three packing algorithms on Amazon Web Services with up to 60 Elastic Compute Cloud (EC2) instances. The 60 C4.LARGE EC2 instances constitute 120 nodes, in which each instance with 2 vCPUs and 3.75GB RAM is shared by two nodes.

The Bitcoin data in year 2012, whose size is around 804 megabytes containing around 1.9 million transactions, is used as the input for the permissioned blockchain.

The performance metrics are the fairness and average response time as discussed in problem definition. The performances of the packing algorithms can be affected by the transaction incoming rate, block generation time, block size, and block validity ratio. We study how the three factors influence the performance of the three packing algorithms. In particular, transaction incoming rate, block size, and block validity ratio are tuned by direct parameter setting, while block generation time is tuned by varying the PoW difficulty. The experiment runs for 5 minutes and 100 times for each parameter setting, e.g., transaction incoming rate as $600tx/s$, blockchain generation time as $5.0s$, block size as $3000tx/bk$, and block validity ratio to be 0.5%. Figure 3.5 presents the results.

3.6.1 Influence of Transaction Incoming Rate

As the transaction incoming rate increases, there will be more transactions in memory pool when generating a block. Moreover, the backlog of memory pool will increase if transaction incoming rate is larger than transaction processing speed. We study the influence of the transaction incoming rate with results shown in Figure 3.5 (a) and

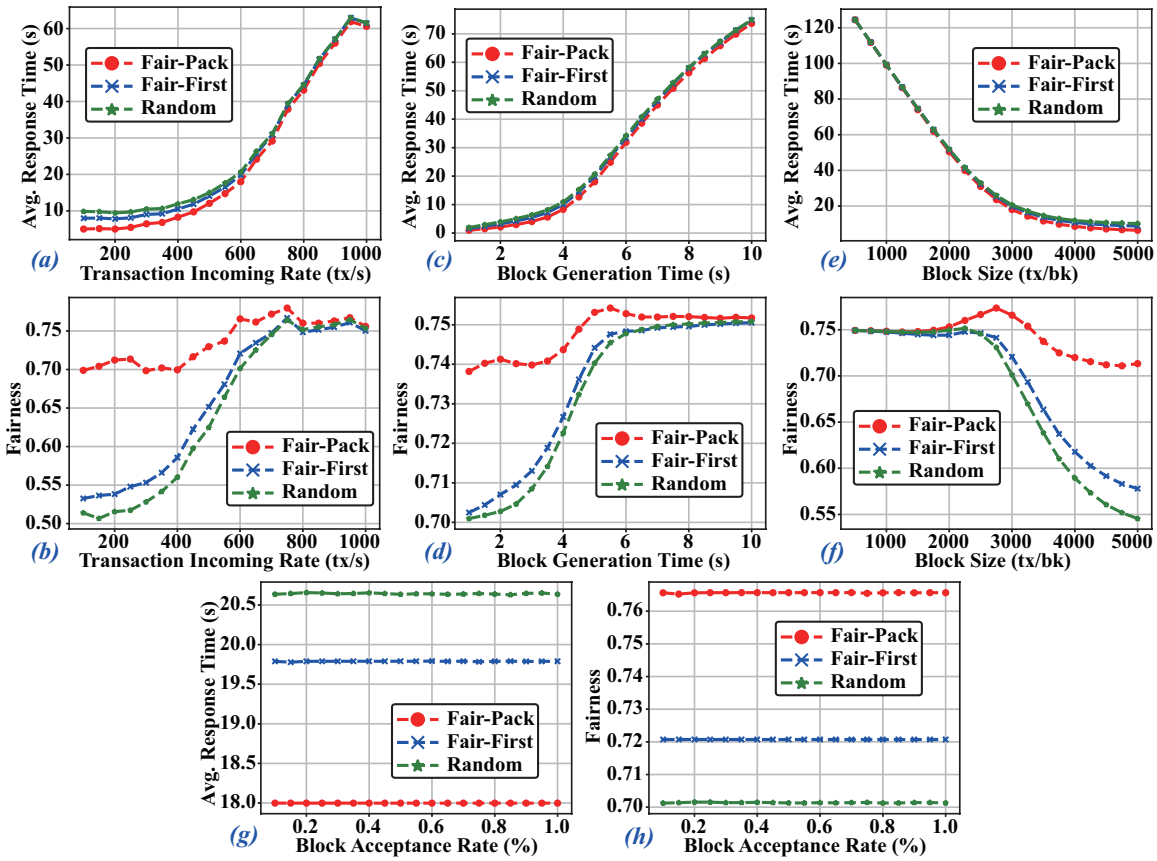

Figure 3.5: Experimental result

(b). Particularly, we vary the transaction incoming rate from $100tx/s$ to $1000tx/s$ with a step of $50tx/s$ and fix the block generation time, block size, and block validity ratio to be $5.0s$, $3000tx/bk$, and 0.5%, respectively.

The transaction incoming rate, if less than $600tx/s$, will be less than or equal to the transaction processing speed, which is calculated to be $\frac{3000tx/bk}{5.0s/bk} = 600tx/s$. On this circumstance, the average response time only slightly increases as the transaction incoming rate increases for all the three transaction packing algorithms. This is because there is nearly no backlog of the memory pool. In terms of fairness, all the three algorithms perform better with the increase of the transaction incoming rate. The reason behind it is that the increasing number of transactions decreases response time differences among the transactions.

When the transaction incoming rate is over $600tx/s$, the backlog of the memory pool will increase as time passes because the transaction processing speed is less than the transaction incoming rate. In this case, more and more transactions remain unpacked in the memory pool, which increases the average response time regardless of the packing algorithms. However, the fairness only fluctuates and even increases. This is because all the transactions in the blockchain incur long response times and the deviation among the response times of the transactions will be smaller.

Overall speaking, all three transaction packing algorithms are influenced by the transaction incoming rate. With different transaction incoming rates, the response time using Fair-Pack is slightly better than one using the other two algorithms. Moreover, Fair-Pack can achieve fairness of 0.70 when the transaction incoming rate is no more than $1000tx/s$ while the fairness using Fair-First and Random are unsatisfactory, i.e., up to 0.54 and 0.52, respectively.

3.6.2 Influence of Block Generation Time

In this subsection, we study how the performances of the three algorithms are affected by the block generation time. The results in terms of the average transaction response time and the fairness are shown in Figure 3.5 (c) and (d), respectively. We vary the block generation time from $1.0s$ to $10.0s$ with a step of $0.5s$. Nonetheless, the transaction incoming rate, the block size, and the block validity ratio are fixed to be $600tx/s$, $3000tx/bk$, and 0.5%, respectively.

The average response time increases with the increase of the block generation time whatever the transaction packing algorithm is. A short block generation time decreases the waiting times of the transactions and increases the possibility for the transactions to be packed. Moreover, Figure 3.1 (c) indicates that the average response time increases remarkably when the block generation time is over $5.0s$, which is the time when the transaction incoming rate is higher than the transaction processing speed. On such circumstances, transactions will be stacked in the memory pool and remain unpacked for a long time. In general, the three algorithms achieve similar average response time regardless of the block generation time.

In terms of fairness, our algorithm outperforms the other two algorithms remarkably when the block generation time is no more than $5.0s$. The number of transactions in the memory pool will be smaller than the block size when the block generation time is less than $5.0s$. In this case, our algorithm Fair-Pack will employ Min-Heap-Op as the underlying transaction selection algorithm, which achieves larger fairness. When the block generation time is over $5.0s$, Fair-Pack still outperforms two other algorithms although with degraded advantages. The reason is that the heuristic algorithm Sum-Index is employed for most of the time on this circumstance.

3.6.3 Influence of Block Size

Block size is another significant factor influencing the performance of the transaction packing algorithms. In the setting, we vary the block size from $500tx/bk$ to $5000tx/bk$ with a step of $250tx/bk$ and set the transaction incoming rate, block generation time, and block validity ratio to be $600tx/s$, $5.0s$, and 0.5%, respectively. The results of average response time and fairness are shown in Figure 3.5 (e) and (f), respectively.

At first glance, Figure 3.5 (e) and (f) are nearly symmetric with Figure 3.5 (c) and (d), respectively. Indeed, the influence of large block size is similar to the effect of a short block generation time. The distinct difference lies in the scale of the y-axis. For example, the average response time can be as least as $1.5s$ when the block generation time is $1s$. However, the best average response time is up to $9s$ when the block size is $5000tx/bk$. The reason is that the average response time depends much on the block generation time, which is fixed to be $6s$ in this subsection.

In Figure 3.5 (f), the fairness among transactions using Fair-First and Random can be as least as 0.58 and 0.55 when the block size is $5000tx/bk$. This is because the deviation of waiting times of the transactions can be vast with large block size. However, our algorithm Fair-Pack remains effective on this circumstance, which results from the theoretically optimal algorithm Min-Heap-Op.

3.6.4 Influence of Block Validity Ratio

Finally, we study the influence of the block validity ratio. In terms of block validity ratio, it naturally comes to our minds that a low block validity ratio can lead to the invalidity of all the blocks containing transactions with large waiting times. Then, a small number of transactions with long waiting times result in long response times of a small set of transactions, substantial deviation of the response times in terms of the full transaction set, and finally poor fairness.

To verify the idea, we conduct experiments in which the transaction incoming rate, the block generation time, and the block size are fixed to be $600tx/s$, $5.0s$, and $3000tx/bk$, respectively and the block validity ratio varies from 0.1% to 1.0% with a step of 0.05%. However, neither the fairness nor the average response time is distinctly affected by the block validity ratio as shown in Figure 3.5. The reason is that the validity of a block is random and can not be set deliberately. As a result, we can pack a transaction with long waiting time as long as a block containing it is valid.

In terms of fairness, Fair-Pack, Fair-First, and Random achieves fairness of 0.765, 0.720, and 0.701, respectively. Note that, an improvement to 0.765 from 0.701 or 0.720 is significant since the value of fairness only varies from 0 (exclusive) to 1 (inclusive), and a fairness of 0.700 is trivial to achieve by random packing. The average response times using Fair-Pack, Fair-First, and Random are around $18.00s$, $19.75s$, and $20.65s$, respectively. That is, Fair-Pack reduces the average response time of Fair-First and Random by 8.9% and 12.8%, respectively.

3.7 Related Work

The existing works related to blockchain fairness can be classified into three categories, i.e., fairness among service providers, between service providers and requesters, and among service requesters.

In terms of fairness among service providers, a service provider contributing a certain proportion of resources is supposed to gain the same portion of rewards in fair blockchains. The research communities have studied such fairness in permissionless blockchains, in which the resource is computational resource and the rewards refer to monetary rewards. It is shown that the Bitcoin mining protocol is not incentive compatible and an attack can make the miners' revenue larger than their fair share [49]. In [48], the authors only consider the rewards of the miner contributing the largest

computational resource and propose Bitcoin-NG, which improves such fairness. In [127], the authors consider approximate fairness among all the miners and propose FruitChains with theoretical analysis.

A service provider is supposed to receive some rewards if the service requester enjoys its service, which is fairness between the service providers and requesters. In traditional systems, the rewards are transferred with the help of a trustworthy third-party. The smart contract in blockchains provides great potential to enhance such fairness since it removes the third party and the transactions are automatically executed. In [94], the authors solve the problem that malicious contractual parties may prematurely abort from a protocol to avoid financial payment. In [107], the authors explore the solution space for enabling the fair exchange of a cryptocurrency payment for a receipt. The fairness between cloud service providers and requesters are investigated in [70] and [185].

The fairness among the service requesters is insufficiently explored in blockchain. In permissionless blockchains, the service requesters are supposed to pay transaction fees in order to make their transactions confirmed [61]. As a result, the transactions with high transaction fees are more likely to be confirmed earlier, which achieves general fairness although not quantified. There is no native cryptocurrency to be paid as transaction fee in permissioned blockchains. Hence, fairness is not defined or investigated. In [79], the fairness problem in permissioned blockchains was first studied and Fair-First was proposed. However, it lacks theoretical analysis, and the performance is not satisfactory.

3.8 Chapter Summary

This chapter presents Fair-Pack, the first fairness-based transaction packing algorithm for permissioned blockchain. In particular, we formally define the fairness

problem and transform it into the problem of subset sum through a proof of the correlation between the fairness and the subset sum of the transaction waiting times. Then, a heuristic algorithm and a min-heap-based optimal algorithm are proposed to solve the subset sum problem for different parameter settings. The proof and the two algorithms contribute to Fair-Pack, a fairness-based transaction packing algorithm for permissioned blockchain. Extensive experimental results have articulated the advantages of Fair-Pack over prior packing algorithms in terms of both fairness and average response time.

Chapter 4

Multi-keyword Search

Nowadays, enterprises tend to store their data in data centers rather than locally due to the increasing demands for storage and computation resource [63]. Because there can be sensitive information, e.g., trade-union membership and health-related records, and even the data center and be malicious, the data is typically encrypted before outsourcing. The encryption, in turn, hinders data utilization, e.g., frequent search operations. Therefore, we need to bring out the technology of searchable symmetric encryption (SSE).

SSE is a technique enabling searching over encrypted data, in which the data owners encrypt not only the data but also the search requests [149]. By this means, the data center knows nearly nothing about the data. However, the data center is assumed to be technically curious but honest in this field [84]. In practice, the data center has the potential to be malicious and deviate from the predefined protocol, e.g., to return only part of the result to save computational resources. Hence, reliability issues arise.

To deal with the reliability issue, the research community has proposed verifiable searchable encryption, in which the data center attaches some flag bits to the result [113]. Upon receiving the result from the data center, the data owners can decode the flag bits to verify the correctness of the result. However, it requires noticeable com-

putational resources for the data owners to decode the flag bits [21]. It is preferable to outsource as many computational tasks as possible to the end devices, especially those with limited battery.

Recently, blockchain technology [119] [79] shows its potential to solve the reliability issue. In these schemes, the blockchain, a distributed ledger maintained by a trustless peer-to-peer network, serves as the data center. The encrypted data with indexes, which is stored in the blockchain and smart contract [170], is used to implement the functions of data storage and data search. Since all the operations are completed by all the nodes in the network, the correctness of the result can be guaranteed as long as the majority of the nodes are honest [70].

Existing blockchain-based searchable encryption schemes [185] [23] focus only on single-keyword search. They can be extended to multi-keyword scenarios by performing a single-keyword search for multiple times and taking the intersection of the results. However, such extensions suffer from privacy and efficiency issues. In particular, the intermediate results, i.e., the data associated with each individual keyword, will be exposed to the service peers. Such data leakage raises the privacy issue. Moreover, the service peers have to handle the single-keyword search requests one after another. Since some keywords can appear in the majority or even all the data, the computational cost to handle such keywords multiple times seems a substantial burden. The large amount of intermediate results also leads to a significant financial cost since they are written to the smart contract by the service peers. After the results for all the keywords are calculated, the service peers have to calculate the intersection of the results, which demands an extra computational cost.

In this chapter, we design a privacy-preserving and efficient data management system with the functions of database setup, dynamic update, and multi-keyword search. The original database to be outsourced is defined as a set of identifier-keyword pairs. That is, there are several keywords associated with each of the identifiers. After setting the database up, the data owner can add or delete some identifier-keyword

pairs dynamically. Meanwhile, the data owner can query all the identifiers that are associated with a set of keywords. The data center is a blockchain network composed of multiple service peers which rent out their computational resources to earn monetary rewards. Inside the blockchain, smart contracts are deployed to fulfill the data services. Because smart contracts are automated programs executed by all the service peers, the data services can be provided with reliability. Furthermore, SSE is employed in smart contracts to preserve privacy. Finally, we propose a bloom filter-enabled multi-keyword search protocol, which reduces the time delay and financial cost remarkably. Next, we discuss the challenges in designing our system and the proposed approaches to overcome them.

The first challenge is to set up an encrypted database without violating the cost limit rule in smart contracts [5]. That is, the service peers contribute their computational resources to maintain the state of the smart contracts. Such work is not free since each operation in the smart contract, e.g., adding two numbers and storage to the local disks, takes certain costs. To guarantee the validity of a smart contract, the cost for a single transaction is bounded, which is called the cost limit rule. In the setup phase, a large number of encrypted data is outsourced to the service peers. In particular, for each identifier-keyword pair, a reversed and encrypted keyword-identifier pair and a tag to support multi-keyword search will be generated in our algorithm. To prevent the setup operation from violating the cost limit rule, we devise a way to estimate the number of bytes that will be generated for each identifier-keyword pair and calculate the number of encrypted data that can be contained in a single transaction. Then, we slice the encrypted database based on the calculation to comply with the cost limit rule. Finally, the encrypted keyword-identifier pairs and the tags are randomly shuffled to prevent data leakage.

The second challenge lies in designing a time- and finance-efficient protocol for multi-keyword search. As discussed previously, the existing approaches are inefficient in terms of time delay and financial cost due to the large number of intermediate results.

In this chapter, we design our multi-keyword search protocol based on the insight that the appearance times of the majority of the keywords are low in the database. First, we generate a tag for each identifier-keyword pair in the setup phase. Meanwhile, we build a bloom filter to record all the high-frequency keywords in the database. In the multi-keyword search phase, we use the bloom filter to find a low-frequency keyword from the search request and use it to filter the database. Note that most of the keywords will be excluded from the result since the selected keyword is of low frequency. Finally, we use the tag of each identifier-keyword pair to check whether each candidate identifier meets the search request.

The rest of this chapter is organized as follows. In Section 4.1, we introduce the system architecture and the protocols of setup, addition, deletion, and multi-search. In Section 4.2, we conduct extensive experiments to evaluate the performance of our protocols in terms of time delay and financial cost. Finally, Section 4.3 discusses the related work and Section 4.4 concludes the chapter.

4.1 Privacy-preserving and Efficient Data Management via Blockchain

4.1.1 System Overview

There are two actors in the blockchain-based data management system, i.e., *service peer* and *data owner*. The service peers are individual nodes in the blockchain network who are renting out computational resources to earn monetary rewards. Data owners are the service requesters who want to outsource their data and later enjoy content update and search services.

There are four kinds of actions between the two actors, i.e., *setup*, *addition*, *deletion*, and *search*. The formal definition of each action is described as follows.

- *Setup.* The data owner outsources a database $W = \{(id_i, w_i)|i = 1, 2, \cdots, l\}$, a list of l identifier-keyword pairs to the service peers. Each $id_i \in \{0,1\}^{\mu}$ is a string of certain length while each $w_i \in \{0,1\}^*$ is a string of uncertain length. In the later sections, we will use l, m, and n to notate the number of identifier-keyword pairs, keywords, and identifiers respectively.

- *Addition.* The data owner adds a set of identifier-keyword pairs $\{id, W_a\}$ to the database W, where W_a is a set of keywords.

- *Deletion.* The data owner deletes a set of identifier-keyword pairs $\{id, W_d\}$ from the database W, where W_a is a set of keywords.

- *Search.* The data owner sends $W_s = \{w_1, w_2, \cdots, w_k\}$ to the service peers to find out all the identifiers ids such that there exists $w \in W_s$ and $(id, w) \in W$.

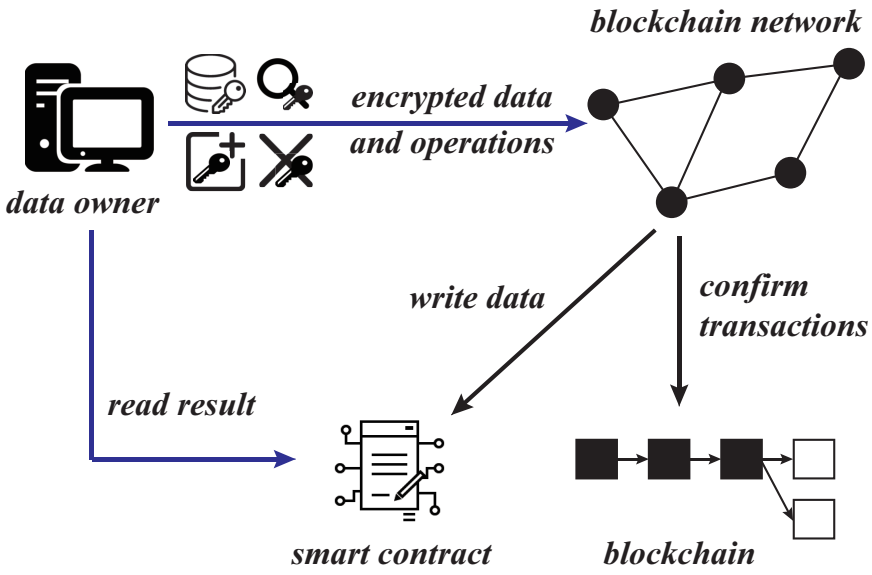

Figure 4.1: System overview

The flowchart of the actions is demonstrated in Figure 4.1. When the data owners perform the operations of setup, addition, deletion, or search, they will send one or more transactions containing the encrypted data and operations to the service

peers. The service peers process the transactions and pack the transactions into the blockchain. After the transaction is confirmed into the blockchain, the service peers will perform the operations which write data to the smart contract. Finally, the data owners can get the results according to the state of the smart contract.

Algorithm 4 Initialization

Service Peers on Initializing the Smart Contract:

1: Allocate a dictionary D_{ori}

2: Allocate two sets S_{del} and S_{tag}

3: Allocate two lists $Flag$ and $Result$

4: Set balance to be B, the money deposited by data owner

We design privacy-preserving and reliable protocols to fulfill the four actions described in the following subsections[1]. Before the illustration of the four actions, we first demonstrate the initialization of the smart contract as shown in Algorithm 4. The smart contract stores five variables, a dictionary D_{ori}, two sets two sets S_{del} and S_{tag}, and two lists $Flag$ and $Result$ for future usage, in which D_{ori} is to store encrypted keyword-identifier pairs, S_{del} and $Flag$ are to support dynamic update of the database, S_{tag} is to support multi-keyword search, and $Result$ is used for storage of the search result. Finally, the balance of the smart contract is set to be B, which is the money deposited by the data owner. The operations cannot cost more than B in the future.

4.1.2 Database Setup

After initializing the smart contract, the data owner can set the database up as shown in Algorithm 5. The data owner aims to store a set of identifier-keyword pairs. First, the data owner generates four secret keys K, K^+, K^-, and K^T. The four keys are all of size λ, which is an adjustable security parameter. Then, for each keyword w, derives two keys K_1 and K_2 are derived from a predefined pseudorandom function

[1]The protocols of addition and deletion are omitted due to page limit.

(PRF) [59] F and the secret key K. The key K_1 is to derive pseudorandom labels for the keywords while the key K_2 is to encrypt the identifiers. Using the keyword w, we can get all the identifiers that are associated with w, which is notated as a set W_w. Afterward, we iterate the identifiers id over W_w. For each id, the PRF F is applied to a counter c using the key K_1 to generate a pseudorandom label l. Meanwhile, we use K_2 to encrypt id using K_2 and get the encrypted identifier d. Then, we add the keyword-identifier pair (l, d) to a list L.

To summarize, we reverse each identifier-keyword pair to an encrypted keyword-identifier pair and store the result in a list L. Besides this, we generate a unique tag for each identifier-keyword pair (id, w) by applying the PRF F over w and id sequentially using the secret key K^T. The tags are accumulated into the list L_{tag}. Note that the tags are used for the multi-keyword search.

The data owner has to send the database to the service peers after encryption. However, the encrypted database has to be sliced before sending due to the cost limit rule in the smart contract. Generally, each operation in smart contract takes a specific cost, and there is a cost limit for each transaction sent to the smart contract. As a result, the number of data that can be attached for each transaction is limited. In our protocol, the data generated for each identifier-keyword pair is designed to be bounded, i.e., a keyword-identifier pair and a tag. The size of the keyword-identifier pair and the tag will be fixed, e.g., 512 bits and 256 bits respectively, if the PRF F is fixed, e.g., HMAC-SHA256 [95].

As a result, we can calculate the maximum number δ of identifier-keyword pairs that can be handled in one transaction. Assume that the PRF F digests message into δ_f bits and the number of bits that can be stored in a single transaction to be δ_t. Then, the value of δ should be $\lfloor \delta_t/(3\delta_f) \rfloor$. In this chapter, at most 10KB can be stored in a transaction and HMAC-SHA256 is used as the PRF. As a result, δ_t equals $80,000$, δ_f equals 256, and δ is calculated to be 104. When the counter reaches δ, we shuffle the lists L and L_{tag}, and send a transaction of setup containing them to the service

Algorithm 5 Setup

Data Owner on Setting W up:

$(K, K^+, K^-, K^T) \leftarrow (\{0,1\}^\lambda, \{0,1\}^\lambda, \{0,1\}^\lambda, \{0,1\}^\lambda)$
Allocate two lists L and L_{tag}
Allocate two local dictionaries D_{count} and D_{key}
$count \leftarrow 0$
for each keyword $w \in W$ **do**
    $K_1||K_2 \leftarrow F(K, w)$
    $K_w^T \leftarrow F(K^T, w)$
    $c \leftarrow 0$
    for each $id \in W_w$ **do**
        $l \leftarrow F(K_1, c)$
        $d \leftarrow Enc(K_2, id)$
        $c \leftarrow c + 1$
        $tag \leftarrow F(K_w^T, id)$
        Append (l, d) to L
        Append tag to L_{tag}
        if $count \geq \delta$ **then**
            Shuffle L and L_{tag} randomly
            Send $(\textsc{Setup}, L, L_{tag})$ to the service peer
            $count \leftarrow 0$
            Empty L and L_{tag}
        end if
    end for
    $cw \leftarrow Get(D_{key}, w)$
    if $cw = \perp$ **then** $cw \leftarrow 0$ **end if**
    $Set(D_{key}, w, cw + c)$
end for

27: Sort all the keywords $w \in W$ according to $Get(D_{key}, w)$ in descending order to get a list of keywords W_s

28: Allocate a list bf of α 0/1 bits initialized to be all 0

29: **for** each keyword w in the first β percentage of W_s **do**

30: $\quad bf \leftarrow (H(w) \mid bf)$

31: **end for**

32: Send $(\textsc{Setup}, L, L_{tag})$ to the service peer

33: Store K, K^+, K^-, K^T, bf, and D_{count} locally

Service Peers on Receiving $(\textsc{Setup}, L, L_{tag})$:

34: Add all the elements (l_i, d_i) in L to the dictionary D_{ori} with l as the key and d as the value

35: Add all the elements in L_{tag} to the set S_{tag}

peer. The reason for shuffling L and L_{tag} is to prevent the service peers from inferring any information related to the data. For example, L and L_{tag} are in the same order with regard to each identifier-keyword pair. After sending each setup transaction, the counter will be reset to be 0 and that lists L and L_{tag} will be emptied. Note that there are additional operations related to the bloom filter bf, which will be explained in the later subsections.

From the perspective of the service peers, they are receiving several transactions of setup together with two lists L and L_{tag}. For each transaction, they enumerate the elements (l_i, d_i) in the list L and add it into the dictionary D_{ori} with l_i as the key and d_i as the vale. Afterward, they store all the elements in L_{tag} to the set S_{tag} in the smart contract. We can see that the data is stored in the smart contract with unordered data structures, i.e., dictionary and set. Therefore, the setup protocol is immune to the order of the transactions received, which is a notable feature.

4.1.3 Dynamic Update

The data owner may update the database after setting it up. We discuss the two kinds of update operations, i.e., addition and deletion, separately in our protocol. In general, a set of deleted elements in S_{del} will be maintained in the smart contract. For each operation of addition and deletion, the data owner will instruct the service peers the way to update S_{del}. The protocols of addition and deletion are shown in Algorithm 6 and Algorithm 7 respectively.

Algorithm 6 Addition

Client on Adding (id, W_a) to W:

Allocate two lists L and L_{tag}
for each $w \in W_a$ **do**
    $K_1^+ || K_2^+ \leftarrow F(K^+, w)$
    $K_1^- \leftarrow F(K^-, w)$
    $c \leftarrow Get(D_{count}, w)$
    if $c = \perp$ **then** $c \leftarrow 0$ **end if**
    $l \leftarrow F(K_1^+, c)$
    $d \leftarrow Enc(K_2^+, id)$
    $delid \leftarrow F(K_1^-, id)$
    $tag \leftarrow F(F(K^T, w), id)$
    Append $(l, d, delid)$ to L
    Append tag to L_{tag}
end for
Shuffle L and L_{tag} randomly
Send $(\textsc{Add}, L, L_{tag})$ to the service peer
Wait for value change of $Flag$
$i \leftarrow 1$

In the protocol of addition, the data owner aims to add a set of keywords W_a to an

18: **for** each $w \in W_a$ **do**
19: **if** the i-th bit of $Flag$ is 0 **then**
20: $c \leftarrow Get(D_{count}, w)$
21: $c \leftarrow c + 1$
22: $D_{count} \leftarrow D_{count} \cup \{(w, c)\}$
23: **end if**
24: $i \leftarrow i + 1$
25: **end for**

Service Peers on Receiving (ADD, L, L_{tag}):

26: Add all the elements in L_{tag} to S_{tag}
27: Allocate a list F of $|L|$ bits
28: $i \leftarrow 1$
29: **for** each $(l, d, delid) \in L$ **do**
30: **if** $delid \in S_{del}$ **then**
31: Set the i-th bit of F to be 1
32: $S_{del} \leftarrow S_{del} \setminus \{delid\}$
33: **else**
34: Set the i-th bit of F to be 0
35: $D_{ori} \leftarrow D_{ori} \cup \{(l, d)\}$
36: **end if**
37: $i \leftarrow i + 1$
38: **end for**
39: Flag $\leftarrow$ F

identifier id. Note that in Algorithm 5, the data owner initializes an empty dictionary D_{count}, which is used to keep the latest counters of all the keywords. For id and each keyword $w \in W_a$ to be added, the data owner will first fetch the counter of w in the dictionary D_{count}. If the counter does not exist, it means w is a new keyword for id and will be processed; otherwise, the identifier-keyword pair will also be processed but will make no effect on the existing data. Then, the data owner will use the key K^+ and the counter to derive the pseudorandom label of the keyword l and the encrypted identifier d. In the meantime, the data owner will generate a $delid$ and a tag for each identifier-keyword pair, in which $delid$ is used to testify whether the identifier is previously deleted from the keyword and tag is used for the multi-keyword search. After processing all the keywords in W_a, the data owner will get a list L of $(l, d, delid)$ and a list L_{tag} of tag, send to the service peers for addition, and wait for the response. Note that the lists L and L_{tag} should also be shuffled before sending for privacy reason.

When the service peers receive an addition request along with two lists L and L_{tag}, they proceed as follows. First, they add all the elements in L_{tag} to the dictionary S_{tag} in the smart contract. Then, for each element $(l, d, delid)$ in the list L_{tag}, they check whether $delid$ is in the set S_{del}. If so, it means (l, d) is previously deleted from the database and should be added back to the database now. Under this circumstance, $delid$ should be removed from the set S_{del} to indicate that the deletion of (l, d) is revoked. Otherwise, it means (l, d) is a new keyword-identifier pair for the database. In this case, (l, d) will be added into the dictionary D_{ori} with l and d to be the key and value respectively. At the same time, the service peers should inform the data owner to update its local counter dictionary D_{count}. To do so, the service peers save a list of $|L|$ 0/1 bits to the variable $Flag$ in the smart contract, in which 0 means (l, d) is a new keyword-identifier pair. In this way, the data owner can update D_{count} accordingly once the value of $Flag$ in the smart contract changes.

In the protocol of deletion, the data owner aims to delete a set of keywords W_d

Algorithm 7 Deletion

Data Owner on Deleting (id, W_d) from W:

1: Allocate two lists L_{del} and L_{tag}
2: **for** each $w \in W_d$ **do**
3: $\quad delid \leftarrow F(F(K^-, w), id)$
4: $\quad tag \leftarrow F(F(K^T, w), id)$
5: $\quad$ Append $delid$ to L_{del}
6: $\quad$ Append tag to L_{tag}
7: **end for**
8: Shuffle L_{del} and L_{tag} randomly
9: Send $(\text{DELETE}, L_{del}, L_{tag})$ to the service peer

Service Peers on Receiving $(\text{DELETE}, L_{del}, L_{tag})$:

10: Add all the elements in L_{del} to S_{del}
11: Remove all the elements in L_{tag} from S_{tag}

associated with an identifier id. The protocol of deletion is more concise than the one of addition. In particular, the data owner derive $delid$ and tag using the key K^- and K^T respectively for each $(id, w) \in W_d$. Then, the data owner collects all the $delid$ and tag into the lists L_{del} and L_{tag} respectively. After shuffling L_{del} and L_{tag} randomly, the data owner sends them to the service peers for the service of deletion. On receiving L_{del} and S_{del} from the data owner, the service peers add their elements into the set S_{del} and S_{tag} respectively, which completes the deletion protocol.

4.1.4 Multi-keyword Search

In the above subsections, we demonstrate the protocols of setup, addition, and deletion, which enables dynamic, reliable, and privacy-preserving storage and update of the data. In this subsection, we demonstrate the protocol for multi-keyword search upon the encrypted database.

To begin with our approach, we introduce the way to build the bloom filter which includes the keywords that frequently appear in the database. In Algorithm 5, we create a dictionary D_{key} to count the appearance time of each keyword, where the appearance time of a keyword w is defined to be:

$$F(w, W) = |\{(id_i, w)|(id_i, w) \in W\}|$$

A keyword is defined to be high-frequency in a database W if it is in the first β percentage when sorting $\{w|w \in W\}$ in a non-increasing order according to the appearance times. A keyword is defined to be low-frequency in W if it is not high-frequency in W.

For each high-frequency keyword w in W, we use a hash function H to hash w into an α-bit 0/1 string $H(w)$ and apply bitwise OR operation to bf using $H(w)$. In this way, bf is a bloom filter containing all the keywords of high frequency. Note that α and β are parameters that should be fine-tuned to make the bloom filter efficient. A large value of α increases the storage burden for the data owner while decreases the false positive rate when judging whether a keyword of high frequency. On the other hand, a large value of β increases the false positive rate while reduces the cost if true positive. In this chapter, we set α and β to be $8,000$ and 10% respectively, which is enough to handle a database of up to 9.1M identifier-keyword pairs.

After setting the bloom filter up, the data owner can use it to find an arbitrary low-frequency keyword among the k keywords in the search request. If the hash value of a keyword does not equal to the result of applying bitwise AND operation to itself with bf, then the keyword must be low-frequency. If such a low-frequency keyword is found, we swap it with the first keyword in the multi-keyword search request; otherwise, there is no low-frequency keyword among the k keywords, and the first keyword will remain high-frequency. Note that we will not make the first keyword to be high-frequency if it is low-frequency before the operation since no true negative judgment happens a bloom filter.

Algorithm 8 Search

Data Owner on Searching $(w_1, w_2, \cdots, w_k)$ upon W:

for $k \leftarrow 1$ to k **do**
    $h \leftarrow H(w_i)$
    if $(h \ \& \ bf) \neq h$ **then**
        Swap w_1 and w_i
        Break
    end if
end for
$K_1 || K_2 \leftarrow F(K, w_1)$
$K_1^+ || K_2^+ \leftarrow F(K^+, w_1)$
$K_1^- \leftarrow F(K^-, w_1)$
for $i \leftarrow 2$ to k **do** $K_i^T \leftarrow F(K^T, w_i)$ **end for**
Send $(\textsc{Search}, K_1, K_2, K_1^+, K_2^+, K_1^-, K_2^T, \cdots, K_k^T)$ to the service peer

Service Peers on Receiving $(\textsc{Search}, K_1, K_2, K_1^+, K_2^+, K_1^-, K_2^T, \cdots, K_k^T)$:

$res \leftarrow \emptyset$
for $c \leftarrow 0$ to ∞ **do**
    $l \leftarrow F(K_1, c)$
    $d \leftarrow Get(D_{ori}, l)$
    if $d = \perp$ **then** Break **end if**
    $res \leftarrow res \cup \{Dec(K_2, d)\}$
end for

20: **for** $c \leftarrow 0$ to ∞ **do**
21: $l \leftarrow F(K_1^+, c)$
22: $d \leftarrow Get(D_{ori}, l)$
23: **if** $d = \perp$ **then** Break **end if**
24: $res \leftarrow res \cup \{Dec(K_2^+, d)\}$
25: **end for**
26: **for** each $id \in res$ **do**
27: $delid \leftarrow F(K_1^-, id)$
28: **if** $delid \in S_{del}$ **then** $res \leftarrow res \setminus \{id\}$
29: **else for** $i \leftarrow 2$ to k **do**
30: **if** $F(K_i^T, id) \notin S_{tag}$ **then**
31: $res \leftarrow res \setminus \{id\}$
32: Break
33: **end if**
34: **end for end if**
35: **end for**
36: $Result \leftarrow res$

Now, it comes to the phase of generating encrypted search request by the data owner. The data owner will take the secret keys K, K^+, and K^- to generate three pseudorandom labels K_1, K_1^+, and K_1^- and two symmetric keys K_2 and K_2^+ for the first keyword w_1. Meanwhile, a tag will be generated for each keyword k_i, where i ranges from 2 to k, using the secret key K^T. In this way, an encrypted search request containing three pseudorandom labels, two symmetric keys, and $k-1$ tags will be generated and sent to the service peers.

On receiving a search request from the data owner, the service peers will start with the first keyword and get a set of candidate identifiers. In particular, they traverse D_{ori} in the smart contract using the pseudorandom K_1 and K_1^+ in sequence. Then, the service peers accumulate all the counters that exist in the key field of D_{ori} after encryption using K_1 or K_1^+. Afterward, the service peers add the identifiers after decryption corresponding to each of the accumulated counters using the corresponding symmetric key. Finally, we deal with the deletion set and the other $k-1$ keywords at the same time. For each of the identifiers id after decryption, we check whether the encrypted result of id using the pseudorandom deletion label K_1^- is in the set of S_{del} and whether any of the $k-1$ tags after applying PRF F to id is not in the tag list S_{tag}. If any of the two conditions hold, id will be excluded from the result.

Finally, we analyze the time delay and financial cost for the traditional method and our method. In the traditional method, it takes $O(l)$ and generates $O(n)$ identifiers for each single-keyword search request. Then, it takes an extra $O(k \cdot n \cdot \log n)$ time to calculate the intersection of k sets, each of which is of size $O(n)$. Since the writing operations dominate the financial cost for a smart contract [103], we approximate the financial cost as the number of identifiers in the intermediate and final results. Hence, the time delay and financial overhead are $O(k \cdot l + k \cdot n \cdot \log n)$ and $O(k \cdot n)$ respectively. In terms of our method, it takes $O(l)$ to use the first keyword to filter the database. Then, $\beta \cdot n$ identifiers will be generated and verified through $k-1$ tags, which takes $O(k \cdot \beta \cdot n)$ time. Therefore, the time overhead for our approach is $O(l + k \cdot \beta \cdot n)$.

In the experiments, we figure that β can be as small as 10%, which reduce the time complexity remarkably. The financial cost overhead is $O(n)$ since only the final result will be written to the smart contract.

4.2 Experimental Result

We implement the operations of database setup, dynamic update, and multi-keyword search in python 2.7 with the PRF implemented by HMAC-SHA256 in the PyCryptodome package and the bloom filter implemented by the pybloom package [2]. We run both the data owner and the service peers on laptops running Ubuntu 16.04.5 with 16GB RAM and two Intel i7-6500U cores. The service peers form a local simulated blockchain network, accept the request from the data owner, and run the smart contract. To focus on the performance of our protocol, we set the block generation time to be 0, which means the influence of the complex network topology is not taken into consideration.

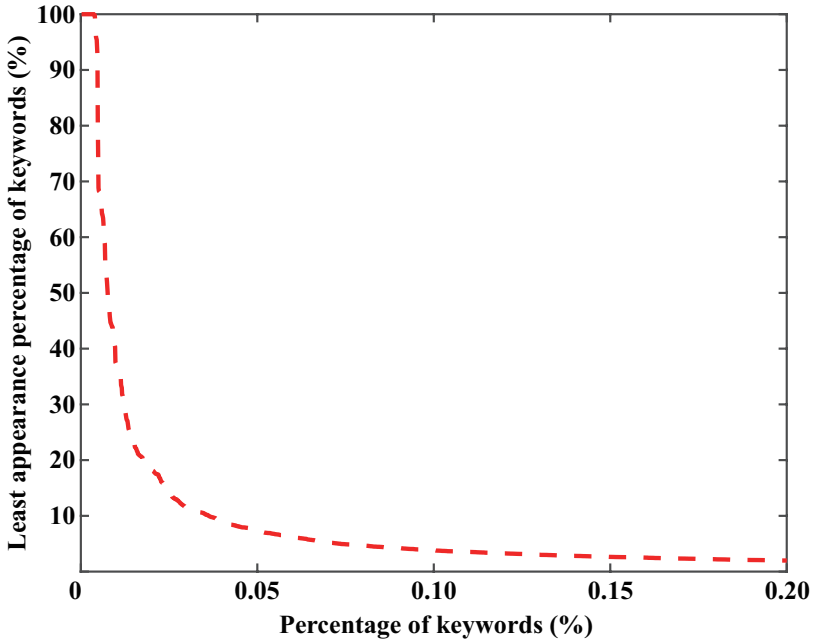

Figure 4.2: Distribution of keyword appearance

After setting up the experimental environment, we conduct extensive experiments on the Eron email dataset [93], which consists of 517K emails. The original database is generated from the dataset as follows. Each email is treated as a new identifier id, and each word after lowercasing in the email is treated as a keyword associated with id. By this means, we get a database consisting of 517K identifiers, 622K distinct keywords, and 9.1M identifier-keyword pairs. The distribution of the keyword appearance is shown in Figure 4.2. Note that some keywords are associated with all the identifiers since some words, e.g., "message", "content", and "type", appear in all the emails. We can see that no more than 0.05% keywords appear in at least 10% identifiers, which means there are very few high-frequency keywords.

4.2.1 Setup and Update

The experimental results of the operations of setup, addition, and deletion are summarized in Table 4.1. In terms of the setup operation, it takes a large amount of time and cost to encrypt such a large database and send the encrypted result to the smart contract through blockchain transactions. At the server side, it takes 272s to encrypt the database, i.e., to generate L, L_{tag}, the four secret keys, and the bloom filter bf. Since the database is of large volume, it takes as many as $42,356$ transactions between the data owner and the service peers to set the encrypted database up. With regard to the service peers, they handle the received transactions by local storage, which takes little time. After accumulating all the transactions, the encrypted database is of size up to 532MB.

The operations of addition and deletion consumes much less time and few transactions compared to the one of setup from the perspectives of both the data owner and the service peers. When the number of keywords to be added or deleted is not too much, e.g., no more than 20 in our experiments, the addition and deletion can be finished within 1s. Meanwhile, it takes only a single transaction to complete each operation.

Table 4.1: With v.s. Without Support of Multi-keyword Search

Ops.	Without Support of Multi-keyword Search	With Support of Multi-keyword Search
Setup	Data Owner 209s Smart Contract 28,219 Txs Service Peer 2.6s Encrypted Database 378MB	Data Owner 272s Smart Contract 42,356 Txs Service Peer 3.0s Encrypted Database 532MB
Addition	Data Owner 1.4s Smart Contract 1 Tx Service Peer 1s	Data Owner 2s Smart Contract 1 Tx Service Peer 1s
Deletion	Data Owner 1s Smart Contract 1 Tx Service Peer 1s	Data Owner 1s Smart Contract 1 Tx Service Peer 1s

To support multi-keyword search, we add some extra data structure such as the bloom filter bf, the secret key K^T, and the tag list L_{tag}. As a result, our protocol requires around 1.5x time, transactions, and storage space compared to those protocols that do not support multi-keyword search. Fortunately, only the setup operation is affected, which is acceptable since setup operation happens once throughout the data usage. As a result, the average time and cost overhead will decrease dramatically with the increasing number of operations of addition, deletion, and search.

4.2.2 Single-keyword Search

In the single-keyword search operation, the time consumption at the data owner side can be neglected since only several operations of symmetric encryption are involved. At the service peer side, it needs to traverse the dictionary D_{ori} twice and write the data to the local state. We conduct experiments on searching keywords with various appearance times. The result is shown in Figure 4.3. We can see that 5.12s is needed when there is no matched identifier, which is the time to traverse the dictionary D_{ori}. Moreover, it takes 15.10s when all the identifiers are associated with the keyword.

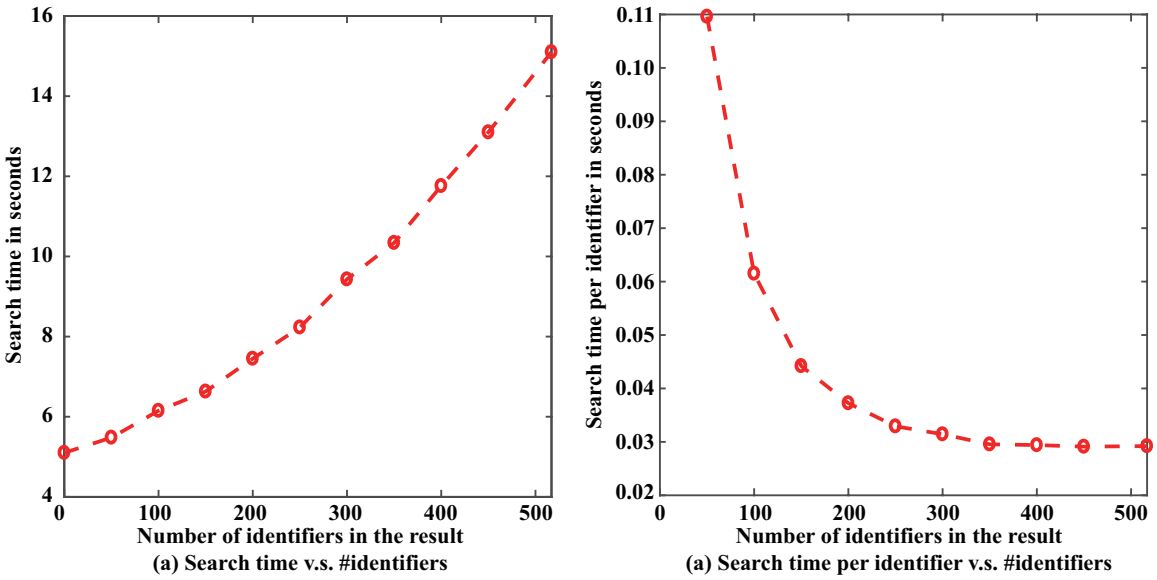

Figure 4.3: Single-keyword search

The search time per identifier decreases as the number of identifiers in the result increases. The reason is that a large number of identifiers can average the time to traverse the dictionary.

4.2.3 Multi-keyword Search

We conduct experiments for multi-keyword search over traditional method and our method for the number of keywords ranging from 2 to 7. The traditional method, or the intersection method, is to apply single-keyword search multiple times and take the intersection of the results as the final result. In our method, we set β to be 10 since no more than 0.05% keywords appears in at least 10% identifiers after analysis. We run the experiments for 50 times and the comparison results in terms of time and financial overhead are shown in Figure 4.4. Note that extreme cases, e.g., all the keywords are of high frequencies, are included in the experiments because the keywords are randomly generated.

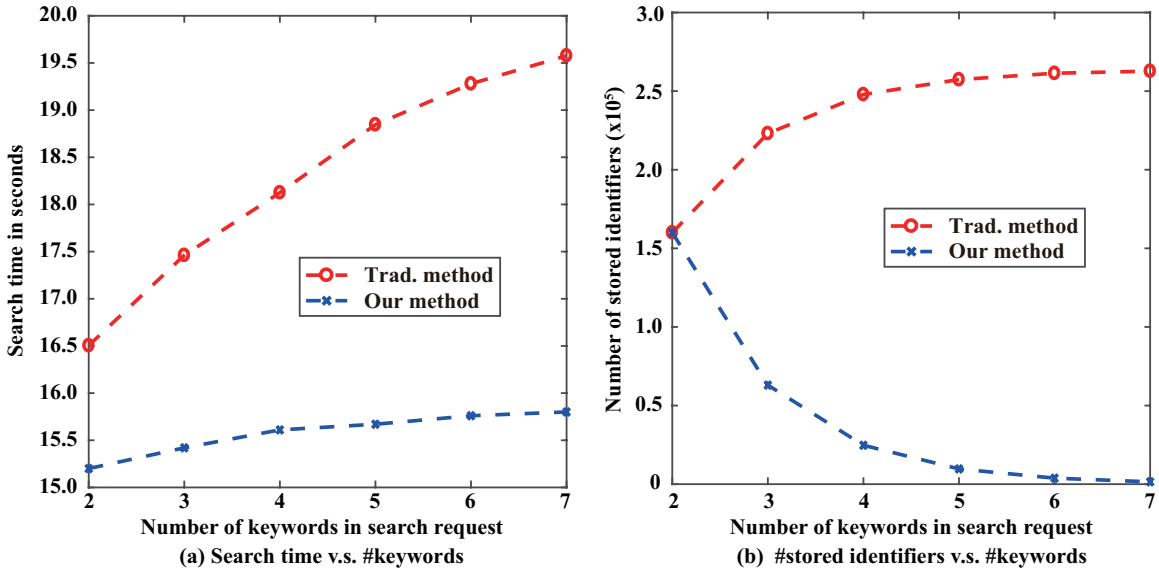

Figure 4.4: Time evaluation for multi-keyword search

In terms of time overhead, the intersection method is significantly affected by the number of keywords since the intersections of more sets should be calculated when the number of keywords increases. For our method, the time to execute a multi-keyword search does not vary too much as the number of keywords increases. The reason is that there will be few candidate identifiers after filtering the database using the first keyword. Moreover, only one operation of tag comparison will be added to the computational burden in case of one more keyword. On average, our method outperforms the intersection method by 14.67% in terms of time efficiency.

In terms of financial overhead, the major financial cost for search operation lies in data storage because data storage dominates the cost compared to other operations. Therefore, we treat the number of identifiers that are written to the state of the smart contract as the financial cost. For the intersection method, the number of stored identifiers increases about 60% when the number of keywords increases from 2 to 5 and remains nearly unchanged for 5 and more keywords. The reason is that there will be few identifiers with the same 5 or more keywords. The financial cost of

our method decreases when the number of keywords increases since we call the smart contract and write the result only once. Besides, the number of eligible identifiers decreases when the number of filtering keywords increase. On average, our method outperforms the intersection method by 59.96% concerning financial cost.

4.3 Related Work

Searchable symmetric encryption is a technique to enable privacy-preserving and secure search over encrypted data between client and server [149]. The research community has been devoted this area for enabling dynamic operations [85], supporting boolean queries [84], and extending to graph database [117]. However, none of these research works considers dishonest servers. Verifiable searchable encryption has the same target as SSE while considering the dishonest servers. The researchers enable the client-side verification by introducing verifiable hash table [21]. Nevertheless, the client is assumed to be honest in their works. Moreover, the client takes non-negligible efforts to verify the results from the servers. In recent years, the research community introduces blockchain to searchable encryption to solve the dishonesty issue of both the client and server[70][185][23][28]. However, they focus on the financial fairness between the miners in blockchain and the clients and suffer from privacy and efficiency issues when extended to multi-keyword search.

4.4 Chapter Summary

In this chapter, we present a blockchain-based data management system with functions of privacy-preserving and efficient database setup, dynamic update and multi-keyword search. The technique to divide the encrypted database into several pieces in the protocols is general for other blockchain applications. The key contribution lies in enabling multi-keyword search over encrypted database on blockchain and improving

its efficiency in terms of time delay and financial cost. To do so, we propose to use a bloom filter to find out a low-frequency keyword and filter the encrypted database using the keyword, which significantly narrows down the searching space. The future work can be tuning the parameters in the bloom filter to further enhance efficiency.

Chapter 5

Dynamic Ring Signature

A limitation to modern elections is that the voters are required to be physically present at the polling booth to cast their vote, which is not convenient at all. Electronic voting (e-voting), in which electronic means are leveraged to take care of the procedure of casting and counting votes, can address the issue of inconvenience and promote voter participation significantly. In addition to convenience, e-voting is environmentally friendly, increases the speed of tallying votes, and eliminates ambiguities [120].

Although e-voting provides many distinctive features, traditional e-voting systems [141][12][39] suffer a lot with regards to auditability and immutability [37]. In particular, they are manipulated by centralized sectors who can fake electronic identities [38] and even tamper the ballot tickets from the voters [40] with great ease. Moreover, the voters have no guarantees that their votes are counted due to non-transparency [35]. Finally, the e-voting machines can be vulnerable to tampering by entities other than the authorities. For example, WINVote touchscreen machines, which were used in the 2016 US elections, were reported to be easily hacked within half a mile[1].

Recently, blockchain, a technology for trustless data storage with auditability, shows great potential in e-voting systems. In particular, the ballot information will be

[1]WIRED: America's Electronic Voting Machines Are Scarily Easy Targets

publicly auditable and can hardly be tampered once stored on blockchain. With the maturity of blockchain technology, blockchain-based e-voting systems (BEVs) are gradually adopted by community, city, and even national voting [96]. For example, West Virginia made history in 2018 when it became the first state to employ BEV in a federal election[2]. Furthermore, governments around the world, e.g., Thailand, South Korea, Sierra Leone, and India, have been experimenting with BEVs[3].

In the beginning, the e-voting systems are constructed based on Bitcoin system [187][18]. These systems cannot guarantee anonymity because Bitcoin pseudo-anonymous with no guarantee on anonymity [119]. Later on, the researchers have been seeking cryptographic methods such as zero-knowledge proof, blind signature, and linkable ring signature, for anonymous blockchain-based e-voting (BEV) system. However, the e-voting systems based on zero-knowledge proof [116] or blind signature [62] either incur high computational complexity or sacrifice auditability.

One of the notable advances in anonymous BEV is the employment of linkable ring signature [179]. In particular, a voter adds some other voters, i.e., *mixins*, when signing a ballot so that the public can verify the signature but cannot identify the exact voter. Linkable ring signature, which is used in the most famous anonymous cryptocurrency Monero [123][151], seems to be perfect for anonymous BEV. Indeed, anonymity can be preserved in a single ballot. However, the anonymity will be compromised when taking a number of transactions or ballots into consideration [97][181]. In particular, up to 94.39% of the voters will be de-anonymized when there are 100 voters, and each ballot contains 2 mixins according to our experiments whose results are shown in Figure 5.4.

In this chapter, we present Roshan, a BEV system with auditability, immutability, and anonymity. We propose *dynamic ring signature*, the first ring signature scheme for e-voting with provable anonymity. The key idea of dynamic ring signature is to select

[2]CoinDesk: West Virginia to Offer Blockchain Voting Statewide in Midterm Elections

[3]CoinDesk: Moscow Said to Hire Kaspersky to Build Voting Blockchain With Bitfury Software

mixins intentionally rather than randomly as in traditional approaches. In particular, we first present a network flow-based algorithm to check whether the anonymity is compromised for a given e-voting system. Then, we propose a time-efficient heuristic algorithm to select mixins for a new ballot. The algorithms of anonymity validation and mixin selection constitute dynamic ring signature. The main contributions of this chapter are as follows:

- We present Roshan, a BEV framework taking the three essential elements of e-voting, i.e., auditability, immutability, and anonymity, into account.
- We propose dynamic ring signature, the first anonymous mechanism for BEV systems with rigorous proof. Such an anonymous mechanism can also be generalized for usage in cryptocurrencies.
- We analyze the time complexity and of dynamic ring signature and the number of mixins required to preserve anonymity. The experimental results indicate that dynamic ring signature is time-efficient and only requires few mixins to provide provable anonymity.

The rest of this chapter is organized as follows. In section 5.1, we demonstrate Roshan, a BEV framework considering auditability, immutability, and anonymity. The key technical depth lies in section 5.2.1, in which we present dynamic ring signature, an anonymous mechanism for BEV. The time complexity analysis and experimental results are shown in section 5.3. Finally, section 5.4 summarizes existing works and section 5.5 concludes the chapter.

5.1 System Architecture

The system architecture of Roshan, a reliable and anonymous e-voting system, is demonstrated in Figure 5.1. Roshan is composed of four entities, i.e., identity authen-

tication center (IAC), ballot organizer (BO), ballot machines (BMs), and blockchain (BC). In brief, BO is the organization to initiate a voting competition, IAC is responsible for identity authentication, BMs are the voters, and BC serves as the storage platform. In the following, we illustrate how Roshan works after the initiation of a voting competition.

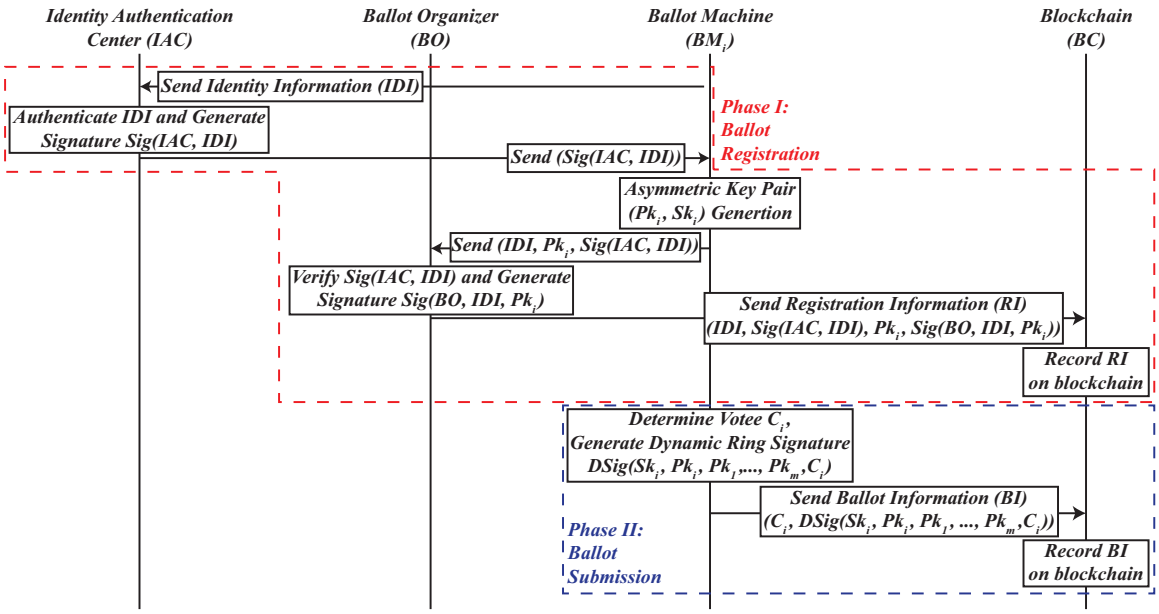

Figure 5.1: System architecture

As a voter, the procedure of e-voting can be divided into two phases, i.e., ballot registration and ballot submission, as shown in Figure 5.1. In Phase I, each BM requests an empty ballot from BO by showing the identity authentication result from IAC. Such a process is called ballot registration, and the registration information is stored on blockchain. In Phase II, each BM fills in the empty ballot and signs it with dynamic ring signature to guarantee anonymity. The filled ballot, represented as ballot information, is also stored on blockchain. It enables auditability to store both the registration information and the ballot information on blockchain.

Asymmetric encryption is used for signature generation and verification in Phase I.

Particularly, the public keys of IAC and BO, i.e., Pk_{iac} and Pk_{bo}, are public to all the entities while their private keys, i.e., Sk_{iac} and Sk_{bo}, are held on their own. First, the IAC authenticates the identity information (IDI) sent from a BM and endorse the IDI using Sk_{iac} if valid. Note that the validation rule for IDI is predefined by the BO, e.g., the requirement for a legal passport. The endorsement of IAC, i.e., $Sig(Sk_{iac}, IDI)$, will be sent back to the corresponding BM. Then, the BM generates an asymmetric key pair Pk_i and Sk_i, in which Pk_i represents its identity, and Sk_i is used for the signature. Afterward, the BM sends its public key Pk_i, it IDI, and the endorsement of IAC to the BO. On receiving such information, the BO will verify the endorsement of IAC using Pk_{iac} and generate signature $Sig(Sk_{bo}, IDI, Pk_i)$. Finally, IDI, Pk_i, the endorsement of IAC, and the endorsement of BO will be packed as the registration information (RI) and recorded on blockchain.

In a piece of RI, IDI and the endorsement of IAC guarantees identity authenticity. Moreover, one valid identity can only bound with one public key Pk_i, which avoids multiple voting by one person. Finally, the endorsement of BO gives birth to a new empty ballot and serves as proof to prevent repudiation. The RI stored on blockchain can be accessed by all the entities in Roshan including the BM. As a result, the BM can notice the empty ballot and fill in it using its private key Sk_i, which is exactly what Phase II achieves.

In Phase II, the voter, i.e., the owner of BM, determines the supporting candidate and signs the ballot. Afterward, the signed ballot will be stored on blockchain for publicity. The transparency of ballot information (BI) increases the trustworthiness of the voting process. However, the identity of the voter will be exposed if traditional asymmetric encryption is employed as the signature algorithm. This is because the public key of the voter, which is used to sign the ballot, is associated with his/her identity in the publicly accessible RI. Such an identity exposure disobeys the anonymity requirement and brings out the privacy issue. In this chapter, we propose a new signature scheme, namely dynamic ring signature, to take good care of anonymity.

5.2 Provable Anonymity via Dynamic Ring Signature

In this section, we introduce dynamic ring signature, the key algorithm to guarantee anonymity in Roshan. First, we give the preliminary about ring signature and a concrete example showing that the existing ring signature is not enough for anonymity and define terminologies such as ballot, e-voting system, ballot assignment, intention, and anonymity. Then, we introduce the problem *anonymity validation* based on the terminologies and present *MFAV*, an algorithm to solve it. Finally, we introduce the problem *mixin selection* and present algorithm *HeurMS* to solve it.

5.2.1 Introduction to Ring Signature

Ring signature [136] is a type of digital signature that can be performed by any member of a group of entities. That is, a message signed with a ring signature is endorsed by someone in a particular group of identities. Consider a group of n entities $e_1, e_2, \cdots, e_n$, each of which has a public/private key pair (Pk_i, Sk_i). Entity e_i can compute a *ring signature* $\sigma = Sig(m, Sk_i, Pk_1, Pk_2, \cdots, Pk_n)$ for a message m such that given σ, m, and $(Pk_1, Pk_2, \cdots, Pk_n)$,

- everyone can validate that σ is generated by one of the entities $e_1, e_2, \cdots, e_n$;
- if $n > 2$, no one knows σ is generated by e_i except e_i itself; and
- it is hard for anyone to create σ without knowing one of the private keys $Sk_1, Sk_2, \cdots, Sk_n$.

Linkable ring signature [111] is derived from ring signature by adding the property of linkability, which allows one to determine whether any two signatures have been produced by the same entity. Consider entity e_x with key pairs (Pk_x, Sk_x)

in two groups G_1 and G_2. Entity e_x can compute two linkable ring signatures $\sigma_1 = Sig(m_1, Sk_x, Pk_{G_1})$ and $\sigma_2 = Sig(m_2, Sk_x, Pk_{G_2})$, in which Pk_{G_1} and Pk_{G_2} are the lists of public keys of the two groups G_1 and G_2. Besides the properties of ring signature, the linkability makes it identifiable that σ_1 and σ_2 are generated from the same entity, in particular, the one with public key Pk_x. Linkable ring signature is used in anonymous cryptocurrencies such as Monero [151]. In linkable ring signature, the public keys, i.e., Pk_{G_1} and Pk_{G_2}, are called mix-in public keys, or *mixins* in short.

5.2.2 Example & Terminologies

It seems enough to guarantee anonymity if BI is endorsed using linkable ring signature as introduced in section 5.2.1. Indeed, anonymity can be preserved in one piece of BI for sure due to features of linkable ring signature, however, will no longer be preserved if a set of BI is taken into consideration. An example is given as follows.

Consider two ballots $b_1 = (\{v_1, v_2\}, c_1)$ and $b_2 = (\{v_1, v_3\}, c_2)$, where b_1 means one of v_1 and v_2 votes for candidate c_1, and b_2 means one of v_1 and v_3 votes for candidate c_2. There are three possibilities between the voters and the candidates as follows:

- v_1 votes for c_1 using b_1, v_2 does not vote, and v_3 votes for c_2 using b_2;
- v_1 votes for c_2 using b_2, v_2 votes for c_1 using b_1, and v_3 does not vote; and
- v_1 does not vote, v_2 votes for c_1 using b_1, and v_3 votes for c_2 using b_2.

As a result, the anonymity is preserved because we cannot infer the intention of any voter. Assume that another ballot $b_3 = (\{v_2, v_3\}, c_1)$, which means one of v_2 and v_3 votes for candidate c_1, is submitted at this moment. Then there are only two possibilities between the voters and the candidates as follows:

- v_1 votes for c_1 using b_1, v_2 votes for c_1 using b_3, and v_3 votes for c_2 using b_2; and

- v_1 votes for c_2 using b_2, v_2 votes for c_1 using b_1, and v_3 votes for c_1 using b_3.

In both cases, v_2 votes for c_1, which means the anonymity is compromised. In the example, all the ballots are submitted using linkable ring signature. However, the anonymity is no longer preserved as the ballots accumulate. It means linkable ring signature is not enough for anonymity. In the following, we introduce dynamic ring signature proposed in Roshan to assure anonymity.

Dynamic ring signature consists of two main components, which are anonymity validation and mixin selection. The core idea is to generate some mixins for linkable ring signature randomly, validate whether the anonymity is compromised, and select new mixins if necessary. To introduce the algorithm, we start from the formal definition of e-voting systems, ballots, ballot assignment, intention, and anonymity.

Definition 11. *An **e-voting system** $\mathcal{E} = \{\mathcal{V}, \mathcal{C}, \mathcal{B}\}$ consists of a set $\mathcal{V} = \{v_1, v_2, \cdots, v_n\}$ of n voters, a set $\mathcal{C} = \{c_1, c_2, \cdots, c_m\}$ of m candidates, and a set $\mathcal{B} = \{b_1, b_2, \cdots, b_k\}$ of k ballots.*

Initially, $\mathcal{B}$ is empty. As the voting activity proceeds, new ballots submitted by voters in $\mathcal{V}$ are added to $\mathcal{B}$ while $\mathcal{V}$ and $\mathcal{C}$ remains unchanged. Note that each voter can submit at most one ballot. As a result, $k \leq n$.

Definition 12. *In an e-voting system $\mathcal{E} = \{\mathcal{V}, \mathcal{C}, \mathcal{B}\}$, a **ballot** $b_i = (V_i, C_i)$ consists of a non-empty set $V_i = \{v_i^1, v_i^2, \cdots, v_i^{l_i}\}$ of voters and a candidate C_i, in which $V_i \subset \mathcal{V}$ and $C_i \in \mathcal{C}$.*

In the definition, V_i are the mixins of the ballot b_i such that everyone knows one voter in V_i submit b_i but no one knows which one. In a ballot, the number of mixins is not bounded while the number of candidate is constrained to be 1.

Definition 13. *In an e-voting system $\mathcal{E} = \{\mathcal{V}, \mathcal{C}, \mathcal{B}\}$, a **ballot assignment** is defined to be an injective function $\mathcal{M} : \mathcal{B} \to \mathcal{V}$ from $\mathcal{B}$ to $\mathcal{V}$ with properties as follows:*

- $\forall b_i \in \mathcal{B}, \mathcal{M}(b_i) \in V_i$; *and*
- $\forall b_i, b_j \in \mathcal{B}, b_i \neq b_j \Rightarrow \mathcal{M}(b_i) \neq \mathcal{M}(b_j)$.

In the definition, the first condition means that each ballot must come from a voter in the mixins, and the second condition means that each voter can submit at most one ballot. There are many possibilities that a ballot comes from because of the mixins. As a result, there can be many possibilities that a set of ballots come from. A ballot assignment stands for one of the possibilities, i.e., to assign each ballot with a voter. Note that there should be at least one ballot assignment given an e-voting system.

Definition 14. *In a ballot assignment $\mathcal{M}$ of an e-voting system $\mathcal{E} = \{\mathcal{V}, \mathcal{C}, \mathcal{B}\}$, the* ***intention*** *$\phi_i^{\mathcal{M}}$ of a voter $v_i \in \mathcal{V}$ is defined as follows:*

- *if there is a ballot $b_j \in \mathcal{B}$ such that $\mathcal{M}(b_j) = v_i$, then $\phi_i^{\mathcal{M}} = C_j$;*
- *otherwise, $\phi_i^{\mathcal{M}} = \perp$, which means the intention of v_i is unclear.*

Given a ballot assignment, we can be aware of the intentions of some voters. Then, we can claim that the anonymity is compromised if the intentions of some voters are clear and remain the same regardless of the ballot assignments. Formally speaking, the anonymity of an e-voting system is defined as follows.

Definition 15. *Given an e-voting system $\mathcal{E} = \{\mathcal{V}, \mathcal{C}, \mathcal{B}\}$, its* ***anonymity is compromised*** *if there is a voter $v_i \in \mathcal{V}$ such that for all ballot assignments $\mathcal{M}_1, \cdots, \mathcal{M}_t$ of $\mathcal{E}$, $\phi_i^{\mathcal{M}_1} = \phi_i^{\mathcal{M}_2} = \cdots = \phi_i^{\mathcal{M}_t} \neq \perp$. The* ***anonymity is preserved*** *if not compromised.*

5.2.3 Anonymity Validation

Given the definition of e-voting system, ballot, ballot assignment, intention, and anonymity, we can define the problem of anonymity validation as follows.

Problem 1. ***Anonymity validation****: given an e-voting system* $\mathcal{E} = \{\mathcal{V}, \mathcal{C}, \mathcal{B}\}$*, determine whether its anonymity is compromised.*

According to the definition of anonymity, an intuitive solution to anonymity validation is to find all the ballot assignments and check whether the intention of each voter is unclear or variant. However, such an approach is time-consuming because the number of ballot assignments can be up to $\Pi_{i=1}^{k} l_i$, in which l_i is the number of mixins in ballot b_i. Because l_i can be up to n, the intuitive approach takes exponential time, in particular, $O(n^k \cdot n)$.

Algorithm 9 GC: Graph Construction

Input: $\mathcal{E}$: an e-voting system

Output: $\mathcal{G}$: a weighted directed graph

1: $\mathcal{G} \leftarrow$ an empty graph with vertex set $V_\mathcal{G}$ and edge set $E_\mathcal{G}$
2: Add a source vertex S and a sink vertex T to $V_\mathcal{G}$
3: **for** $i \leftarrow 1$ **to** n **do** ▷ each voter
4: Add a vertex a_i to $V_\mathcal{G}$ with label v_i
5: Add an edge $(S, a_i, 1)$ to $E_\mathcal{G}$
6: **end for**
7: **for** $i \leftarrow 1$ **to** k **do** ▷ each ballot $(\{v_i^1, \cdots, v_i^{l_i}\}, C_i)$
8: Add a vertex b_i to $V_\mathcal{G}$ with label C_i
9: Add an edge $(b_i, T, 1)$ to $E_\mathcal{G}$
10: **for** $j \leftarrow 1$ **to** l_i **do** ▷ each mixin
11: Add an edge $(u, b_i, 1)$ to $E_\mathcal{G}$, in which u is the vertex with label v_i^j
12: **end for**
13: **end for**
14: **return** $\mathcal{G}$

In this chapter, we propose a time-efficient algorithm MFAV, which solves anonymity validation in polynomial time. In MFAV, we construct a directed weighted graph

Algorithm 10 MFAV: Anonymity Validation

Input: $\mathcal{E}$: an e-voting system; MF: a maximum flow algorithm

Output: If anonymity is compromised in $\mathcal{E}$, output (v_i, c_j) indicating v_i vote for c_j in all ballot assignments; otherwise, output $\perp$ indicating anonymity is preserved

$\delta \leftarrow \text{MF}(\text{GC}(\mathcal{E}))$
for $i \leftarrow 1$ **to** n **do** ▷ voter v_i
  for $j \leftarrow 1$ **to** m **do** ▷ candidate c_j
    $\mathcal{E}' \leftarrow \mathcal{E}$
    for $u \leftarrow 1$ **to** k **do** ▷ ballot $(\{v_u^1, \cdots, v_u^{l_u}\}, C_u)$
      if $C_u = c_j$ **then**
        remove v_i from $\mathcal{E}'.\mathcal{B}.b_u.V_u$ if exists
      end if
    end for
    $\delta' \leftarrow \text{MF}(\text{GC}(\mathcal{E}'))$
    if $\delta > \delta'$ **then**
      return (v_i, c_j)
    end if
  end for
end for
return $\perp$

according to the input e-voting system and solve anonymity validation using maximum flow theory. In particular, we prove that for an e-voting system, the maximum flow of the constructed graph equals the number of ballots. Moreover, there is a one-to-one correspondence between the ballot assignments and the maximum flow solutions. Then, we assume that the intention of voter v_i is not candidate c_j for all ballot assignments. With this assumption, we can prune the e-voting system, i.e., delete voter v_i from all the ballots whose candidate is c_j, and get a new e-voting system. Note that the new e-voting system can be invalid because modifications are made upon the ballot set. We compare the maximum flow of the constructed graph of the new e-voting system and the number of ballots. If not equal, it means there is no ballot assignment for the new e-voting system, which means the new e-voting system is invalid. Therefore, the assumption does not hold, and the intention of v_i is c_j in all ballot assignments of the original e-voting system. According to the definition of anonymity, the anonymity of the original e-voting system is compromised.

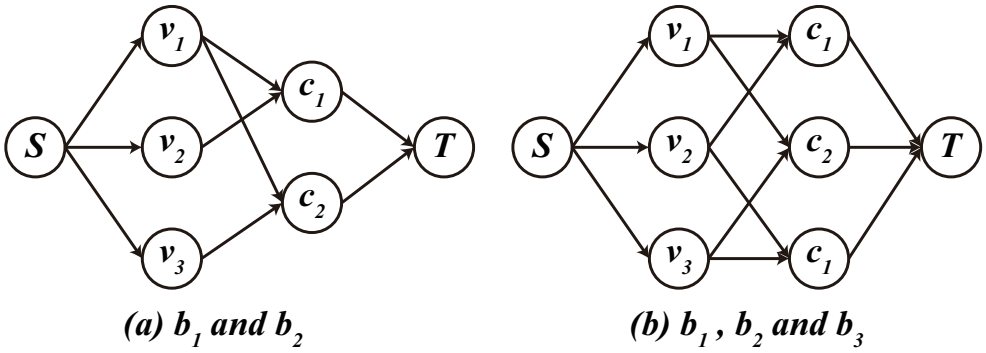

Figure 5.2: Example construct graphs ($b_1 = (\{v_1, v_2\}, c_1), b_2 = (\{v_1, v_3\}, c_2), b_3 = (\{v_2, v_3\}, c_1)$)

The graph construction algorithm is shown in Algorithm 9. In short, we first construct a source vertex, a sink vertex, a vertex for each voter, and a vertex for each ballot, then add edges between the voters and the ballots based on the ballot information. Example constructed graphs are shown in Figure 5.2.

Theorem 9. *For an e-voting system, the maximum flow of the constructed graph equals to the number of ballots.*

Proof. Notate the e-voting system as $\mathcal{E} = \{\mathcal{V}, \mathcal{C}, \mathcal{B}\}$ in which $|\mathcal{V}| = n, |\mathcal{C}| = m, |\mathcal{B}| = k$. Notate the constructed graph according to Algorithm 9 as $\mathcal{G}$.

On one hand, the maximum flow is no larger than k as shown below. In Algorithm 9, only line 9 constructs edge linked with the sink vertex T. In particular, line 9 runs for k times in the loop of line 7. Therefore, the input flow of sink vertex T is at most k in the constructed graph $\mathcal{G}$. One the other hand, there is a flow of size k in the constructed graph $\mathcal{G}$. Consider an arbitrary ballot assignment $\mathcal{M}$ for $\mathcal{E}$. For each ballot $b_i = (V_i, C_i)$, there is an edge between b_i and $\mathcal{M}(b_i)$ because $\mathcal{M}(b_i) \in V_i$ (definition) and b_i is linked with all the vertices in V_i (line 11). Moreover, $\mathcal{M}(b_i)$ is linked with S is $\mathcal{M}(b_i) \in \mathcal{V}$ (definition) and S is linked with all the vertices in $\mathcal{V}$ (line 5). Finally, b_i is linked with T as shown in line 9. As a result, there is a flow $S - \mathcal{M}(b_i) - b_i - T$ of size 1 for each ballot b_i. Nevertheless, all the $\mathcal{M}(b_i)$ are different according to the definition of ballot assignment, which means the flows do not share any edge. Because there are k ballots, so the total flow is of size k.

Because the maximum flow is no larger than k and a flow of size k is constructed in $\mathcal{G}$, we can conclude that the maximum flow of $\mathcal{G}$ is k, which is the number of ballots. □

Theorem 10. *For an e-voting system, there is an one-to-one correspondence between the ballot assignments and the maximum flow solutions for the constructed graph using Algorithm 9.*

Proof. On one hand, we construct a maximum flow solution for each ballot assignment according to the proof for Theorem 9. On the other hand, we construct a ballot assignment for each maximum flow solution, which is demonstrated as follows.

Notate the e-voting system as $\mathcal{E} = \{\mathcal{V}, \mathcal{C}, \mathcal{B}\}$ in which $|\mathcal{V}| = n, |\mathcal{C}| = m, |\mathcal{B}| = k$. Notate the constructed graph according to Algorithm 9 as $\mathcal{G}$. Consider an arbitrary

maximum flow solution $\mathcal{F}$ in $\mathcal{G}$. The size of $\mathcal{F}$ must be k according to the proof for Theorem 9. Because there are $n > k$ vertices $a_1, \cdots, a_n$ linked to the source vertex S and the edges are all of weight 1, the solution $\mathcal{F}$ should contain k edges $(S, a_{\sigma_1}, 1), \cdots, (S, a_{\sigma_k}, 1)$, in which $\{\sigma_1, \cdots, \sigma_k\}$ is a subset of size k of $\{1, \cdots, n\}$. Note that the label of vertex a_{σ_i} is v_{σ_i}. Because there are k vertices $b_1, \cdots, b_k$ linked to the sink vertex T and the edges are all of weight 1, the solution $\mathcal{F}$ should contain k edges $(b_1, T, 1), \cdots, (b_k, T, 1)$. In order to connect two vertex sets $\{a_{\sigma_1}, \cdots, a_{\sigma_k}\}$ and $\{b_1, \cdots, b_k\}$, the solution $\mathcal{F}$ should contain k edges in between . Without loss of generality, we assume the edges are $(a_{\sigma_1}, b_1, 1), \cdots, (a_{\sigma_k}, b_k, 1)$, in which $v_{\sigma_i} \in V_i$. As a result, the solution $\mathcal{F}$ is a collection of edges $\mathcal{F} = \{(S, a_{\sigma_i}, 1), (a_{\sigma_i}, b_i, 1), (b_i, T, 1) \mid i = 1, \cdots, k\}$.

Based on $\mathcal{F}$, we can construct a function $\mathcal{M}$, in which $\mathcal{M}(b_i) = v_{\sigma_i}$ for each $(a_{\sigma_i}, b_i, 1) \in \mathcal{F}$. The function $\mathcal{M}$ is an injective function from $\mathcal{B}$ to $\mathcal{V}$ because:

- For each $b_i \in \mathcal{B}$, $(a_{\sigma_i}, b_i, 1) \in \mathcal{F}$ and $v_{\sigma_i} \in V_i$. Hence, $\forall b_i \in \mathcal{B}, \mathcal{M}(b_i) = v_{\sigma_i} \in V_i$; and
- $\forall b_i, b_j \in \mathcal{B}$, if $b_i \neq b_j$, then $\mathcal{M}(b_i) = v_{\sigma_i} \neq v_{\sigma_j} \mathcal{M}(b_j)$.

Therefore, $\mathcal{M} : \mathcal{B} \to \mathcal{V}$ is a ballot assignment constructed from the maximum flow solution $\mathcal{F}$. The two constructions demonstrate the one-to-one correspondence relationship, which completes the proof. □

An example of the one-to-one correspondence between the ballot assignments and maximum flow solutions is shown in Figure 5.3. There are three ballots, i.e., $b_1 = (\{v_1, v_2\}, c_1)$, $b_2 = (\{v_1, v_3\}, c_2)$, and $b_3 = (\{v_2, v_3\}, c_1)$, in the example e-voting system and exactly two ballot assignments for it. In the first ballot assignment, b_1 is sent by v_1, b_2 is sent by v_3, and b_3 is sent by v_2, which are represented as red, blue, and yellow paths in Figure 5.3(a). Note that the colored edges in Figure 5.3(a) are also a maximum flow solution for the graph. Similarly in the second ballot assignment, b_1

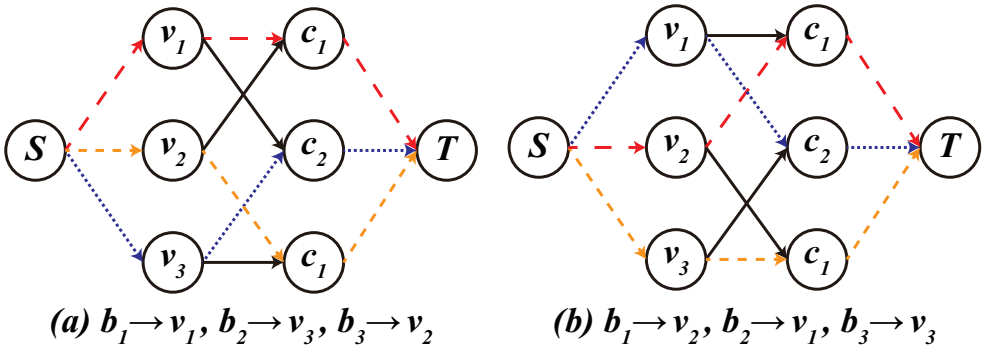

Figure 5.3: Example ballot assignments and maximum flow solutions for $\{b_1 = (\{v_1, v_2\}, c_1), b_2 = (\{v_1, v_3\}, c_2), b_3 = (\{v_2, v_3\}, c_1)\}$

is sent by v_2, b_2 is sent by v_1, and b_3 is sent by v_3. The corresponding maximum flow solution is showcased in Figure 5.3(b).

Next, we explain the correctness of Algorithm 10. First, we explain the construction of a new e-voting system $\mathcal{E}'(v_i, c_j)$ given a voter v_i and a candidate c_j in an e-voting system $\mathcal{E}$ as follows:

- $\mathcal{E}'.\mathcal{V} \leftarrow \mathcal{E}.\mathcal{V}$;
- $\mathcal{E}'.\mathcal{C} \leftarrow \mathcal{E}.\mathcal{C}$;
- for each ballot $(\{v_u^1, \cdots, v_u^{l_u}\}, C_u), \mathcal{E}'.\mathcal{B}.b_u.C_u \leftarrow \mathcal{E}.\mathcal{B}.b_u.C_u$; and
- for each ballot $(\{v_u^1, \cdots, v_u^{l_u}\}, C_u)$, if $C_u = c_j$, then $\mathcal{E}'.\mathcal{B}.b_u.V_u \leftarrow \mathcal{E}.\mathcal{B}.b_u.V_u \backslash \{v_i\}$; otherwise, $\mathcal{E}'.\mathcal{B}.b_u.V_u \leftarrow \mathcal{E}.\mathcal{B}.b_u.V_u$.

Such a construction is showcased in Algorithm 10 from line 4 to 9. In short, $\mathcal{E}(v_i, c_j)$ deletes all the possibilities that v_i votes for c_j from $\mathcal{E}$.

Theorem 11. *In an e-voting system $\mathcal{E} = \{\mathcal{V}, \mathcal{C}, \mathcal{B}\}$, for an arbitrary voter $v_i \in \mathcal{V}$ and an arbitrary candidate $c_j \in \mathcal{C}$, the maximum flow of $GC(\mathcal{E}'(v_i, c_j))$ is of size $|B|$, if and only if there is a ballot assignment $\mathcal{M}$ of $\mathcal{E}$ such that $\phi_i^{\mathcal{M}} \neq c_j$.*

Proof. The theorem can be expressed as a logic formula as follows:

$$\forall v_i, c_j, (\exists \mathcal{M}, \phi_i^{\mathcal{M}} \neq c_j) \Leftrightarrow MF(GC(\mathcal{E}'(v_i, c_j))) = k \tag{5.1}$$

First, we prove the $\Rightarrow$ direction. The condition that $\exists \mathcal{M}, \phi_i^{\mathcal{M}} \neq c_j$ means the intention of v_i is not c_j in $\mathcal{M}$. According to definition 14, we know that for all $b_u \in \mathcal{B}$, if $C_u = c_j$, then $\mathcal{M}(b_u) \neq v_i$. As a result, $\mathcal{M}$ is a ballot assignment of $\mathcal{E}'(v_i, c_j)$ as well. According to Theorem 10, there is a corresponding maximum flow solution of $GC(\mathcal{E}'(v_i, c_j))$ in $\mathcal{E}'(v_i, c_j)$. Furthermore, the size of the maximum flow is $|B|$ according to Theorem 9, which completes the proof.

Then, we prove the $\Leftarrow$ direction. The condition that $MF(GC(\mathcal{E}'(v_i, c_j))) = k$ means there is a maximum flow of size k in the graph $GC(\mathcal{E}'(v_i, c_j))$. According to Theorem 10, there is a corresponding ballot assignment $\mathcal{M}$ for the e-voting system $\mathcal{E}'(v_i, c_j)$. It is clear that $\mathcal{M}$ is also a ballot assignment of $\mathcal{E}$ because $\mathcal{E}'(v_i, c_j)$ is pruned from $\mathcal{E}$. Moreover, $\phi_i^{\mathcal{M}} \neq c_j$ because v_i is deleted from the mixins of all the ballots whose candidates are c_j, which completes the proof. □

Theorem 12. *Algorithm 10 solves problem 1.*

Proof. At first, Algorithm 10 computes the maximum flow δ of $\mathcal{E}$. Note that δ equals to k for sure. Secondly, Algorithm 10 enumerates voter v_i and candidate c_j on line 2 and 3. Thirdly, $\mathcal{E}'(v_i, c_j)$ is computed from line 4 to 9 and the maximum flow δ' of $\mathcal{E}'(v_i, c_j)$ is computed on line 10. If δ' does not equal to δ, then Algorithm 10 asserts that the anonymity is compromised on line 12. The correctness of the assertion is justified as follows. The condition that $\delta \neq \delta'$ can be expressed as:

$$\exists v_i, c_j, MF(GC(\mathcal{E}'(v_i, c_j))) \neq k \tag{5.2}$$

From equation 5.1 we can get:

$$\exists v_i, c_j, (\exists \mathcal{M}, \phi_i^{\mathcal{M}} \neq c_j) \Rightarrow MF(GC(\mathcal{E}'(v_i, c_j))) = k \tag{5.3}$$

We can get formula as follows when applying syllogism on equation 5.2 and 5.3:

$$\exists v_i, c_j, \forall \mathcal{M}, \phi_i^{\mathcal{M}} = c_j \tag{5.4}$$

According to definition 15, equation 5.4 indicates that the anonymity is compromised; in particular, the intention of v_i is c_j in all ballot assignments of $\mathcal{E}$.

Finally, when Algorithm 10 proceeds to line 16, it means:

$$\forall v_i, c_j, MF(GC(\mathcal{E}')) = k \tag{5.5}$$

We apply syllogism on equation 5.1 and 5.5 and can get:

$$\forall v_i, c_j, \exists \mathcal{M}, \phi_i^{\mathcal{M}} \neq c_j \tag{5.6}$$

Equation 5.6 indicates the preservation of anonymity, which completes the proof. □

5.2.4 Mixin Selection

The anonymity in an e-voting system can be compromised because the mixins of the ballots are not appropriately selected. In this section, we define the mixin selection problem and present dynamic ring signature to solve it.

Definition 16. *Given an e-voting system $\mathcal{E} = \{\mathcal{V}, \mathcal{C}, \mathcal{B}\}$, an **raw ballot** $\bar{b} = (\bar{V}, \bar{C})$ consists of a voter $\bar{V}$ and a candidate $\bar{C}$, in which $\bar{V} \in \mathcal{V}$ and $\bar{C} \in \mathcal{C}$.*

A raw ballot is a ballot whose mixins are not yet determined. The problem of mixin selection concerns how to determine the mixins of a raw ballot to prevent the anonymity of a given e-voting system from being compromised.

Problem 2. ***Mixin selection:*** *given an e-voting system $\mathcal{E} = \{\mathcal{V}, \mathcal{C}, \mathcal{B}\}$ and a raw ballot $\bar{b} = (\bar{V}, \bar{C})$, determine the mixins $V' \cup \{\bar{V}\}$ of $\bar{b}$ so that the anonymity of the new e-voting system, i.e., $\mathcal{E}' = \{\mathcal{V}, \mathcal{C}, \mathcal{B} \cup \{(V' \cup \{\bar{V}\}, \bar{C})\}\}$, is not compromised, or there are no such mixins.*

The problem of mixin selection is easy to solve as follows: if $\mathcal{V}$ can preserve the anonymity in the new e-voting system, then output $\mathcal{V}$; otherwise, assert that there is no answer. Such an naive algorithm is based on the intuition that more mixins can *enhance* the anonymity *more significantly*, which is formally stated as Theorem 13 and Corollary1.

Theorem 13. *If $V \subset \mathcal{V}$ is an answer to the problem of mixin selection, then $V \cup \{v_i\}$ is also an answer where $v_i \in \mathcal{V}$ is an arbitrary voter.*

Proof. Notate the e-voting system $\{\mathcal{V}, \mathcal{C}, \mathcal{B} \cup \{(V, \bar{C})\}\}$ as $\mathcal{E}_1$. Notate the e-voting system $\{\mathcal{V}, \mathcal{C}, \mathcal{B} \cup \{(V \cup \{v_i\}, \bar{C})\}\}$ as $\mathcal{E}_2$. Because V is an answer to the problem of mixin selection, the anonymity of $\mathcal{E}_1$ is preserved. As a result, the function call of MFAV($\mathcal{E}_1$) returns $\perp$.

For an arbitrary pair of voter and candidate, notate the maximum flows calculated on line 10 in MFAV($\mathcal{E}_1$) and MFAV($\mathcal{E}_2$) as δ'_1 and δ'_2 respectively. Because V is a subset of $V \cup \{v_i\}$, the edges of GC($\mathcal{E}_1$) is a subset of the edges of GC($\mathcal{E}_2$) according to Algorithm 9. Therefore, $\delta'_1 < \delta'_2$. Because the function call of MFAV($\mathcal{E}_1$) returns $\perp$, the condition of anonymity compromise on line 11, i.e., $\delta > \delta'_1$ will always be false. Consequently, $\delta > \delta'_2$ will always be false as well. We can assert that MFAV($\mathcal{E}_2$) will return $\perp$ and the anonymity of $\mathcal{E}_2$ is preserved, which completes the proof. □

Corollary 1. *If $\mathcal{V}$ is not an answer to the problem of mixin selection, then there is no answer.*

Proof. Assume for contradiction that there is an answer V to the problem of mixin selection given that $\mathcal{V}$ is not an answer. We know V must be a subset of $\mathcal{V}$ because $\mathcal{V}$ is the universe. According to Theorem 13, $\mathcal{V}$ is also an answer to the problem of mixin selection, which leads to a contradiction. As a result, the assumption does not hold, which completes the proof. □

Although the problem of mixin selection can be solved by simply checking whether $\mathcal{V}$ is an answer, however, it is not wise to select $\mathcal{V}$ as the mixins if it is an answer. This is because it is time-consuming to use such a large amount of mixins to generate the ring signature and for future signature verification. As a result, we aim to find an answer of size as minimum as possible. Such a problem is formally stated as follows.

Problem 3. ***Minxin selection revisited (MSR)****: given an e-voting system* $\mathcal{E} = \{\mathcal{V}, \mathcal{C}, \mathcal{B}\}$ *and a raw ballot* $\bar{b} = (\bar{V}, \bar{C})$*, determine the mixins* $V' \cup \{\bar{V}\}$ *of* $\bar{b}$ *of* ***minimum size*** *so that the anonymity of the new e-voting system, i.e.,* $\mathcal{E}' = \{\mathcal{V}, \mathcal{C}, \mathcal{B} \cup \{(V' \cup \{\bar{V}\}, \bar{C})\}\}$*, is not compromised, or there are no such mixins.*

The problem MSR is not easy to solve because the number of possible mixin sets is up to 2^{n-1}. That is, it takes exponential time if brute force algorithm is employed. In this chapter, we propose HeruMS as shown in Algorithm 11, a heuristic algorithm with polynomial time complexity for the problem MSR.

Algorithm 11 is based on the intuition to include those voters that frequently appear in ballots of different candidates. In particular, we first calculate the appearance time of each voter from line 6 to 14, which serves as the heuristic value. Then, the voters are enumerated according to their heuristic values in descending order from line 16 to 23. Inside the enumeration, we try to add a voter into the mixin set and invoke Algorithm 10 to verify whether current mixins can preserve anonymity. If so, an answer to the problem MSR is found. If all the voters are added into mixin set while the anonymity is still compromised, we can assert that there are no such mixins according to Corollary1.

Theorem 14. *Algorithm 11 solves problem 2.*

Proof. In Algorithm 11, $V \cup \{\bar{V}\}$ equals $\mathcal{V}$ if it proceeds to line 23. According to Corollary1, there is no answer if $\mathcal{V}$ is not an answer to problem 2, which completes the proof. □

Algorithm 11 HeurMS: mixin selection

Input: $\mathcal{E}$: an e-voting system; $\bar{b}$: a raw ballot

Output: V: mixins for $\bar{b}$ to preserve anonymity, or $\perp$ if there are no such mixins

1: $V \leftarrow \emptyset$
2: $\mathcal{W} \leftarrow$ a function from voters $\mathcal{V}$ to integers $\mathbb{Z}$
3: **for** $i \leftarrow 1$ **to** n **do**
4: $\quad \mathcal{W}(v_i) \leftarrow 0$
5: **end for**
6: **for** $i \leftarrow 1$ **to** k **do**
7: $\quad$ **for** $j \leftarrow 1$ **to** l_i **do**
8: $\quad\quad$ **if** $C_i = \bar{C}$ **then**
9: $\quad\quad\quad \mathcal{W}(v_i^j) \leftarrow \mathcal{W}(v_i^j) - 1$
10: $\quad\quad$ **else**
11: $\quad\quad\quad \mathcal{W}(v_i^j) \leftarrow \mathcal{W}(v_i^j) + 1$
12: $\quad\quad$ **end if**
13: $\quad$ **end for**
14: **end for**
15: Sort $\mathcal{V}$ according to $\mathcal{W}(\mathcal{V})$ in descending order
16: **for** $i \leftarrow 1$ **to** n **do**
17: $\quad$ **if** $v_i \neq \bar{V}$ **then**
18: $\quad\quad V \leftarrow V \cup \{v_i\}$
19: $\quad\quad$ **if** $\mathrm{MFAV}(\{\mathcal{V}, \mathcal{C}, \mathcal{B} \cup \{(V \cup \{\bar{V}\}, \bar{C})\}\}) = \perp$ **then**
20: $\quad\quad\quad$ **return** $V \cup \{\bar{V}\}$
21: $\quad\quad$ **end if**
22: $\quad$ **end if**
23: **end for**
24: **return** $\perp$

5.3 Analysis & Experiments

In this section, we first analyze the time complexity of Algorithm 10 and Algorithm 11 in subsection 5.3.1. Then, we demonstrate the severe de-anonymization issue of the traditional approach by showing the number of compromised voters with different numbers of random mixins in subsection 5.3.2. Furthermore, the average number of mixins and time consumption of Algorithm 11 with different numbers of ballots and candidates are evaluated in subsection 5.3.3 and 5.3.4 respectively. Finally, subsection 5.3.5 validates the correctness of Algorithm 11 in the setting that multiple ballots are submitted concurrently. To summarize, the traditional approach that selects mixins randomly suffers from the de-anonymization issue severely, while dynamic ring signature can address such an issue with a small number of mixins in a time-efficient way, even when the multiple ballots are submitted concurrently.

5.3.1 Time Complexity Analysis

In this subsection, we analyze the time complexity of Algorithm 10 and 11, and show that both of them are polynomial to the number of voters, ballots, and candidates.

Theorem 15. *Algorithm 10 and Algorithm 11 takes* $O(nmk(n + k + \sum_{i=1}^{k} l_i))$ *and* $O(n^2mk(n + k + \sum_{i=1}^{k} l_i))$ *respectively.*

Proof. First, for a given e-voting system, there are $n + k + 2$ vertices and $n + k + \sum_{i=1}^{k} l_i$ edges in the constructed graph $\mathcal{G}$ using Algorithm 9. Then, Algorithm 10 enumerates each voter and each candidate and invokes a maximum flow algorithm on the corresponding constructed graph. We employ Ford-Fulkerson algorithm [34], whose complexity is $O(E \cdot f)$ in a graph with E edges and maximum flow size of f, as the maximum flow algorithm. Because the maximum size of the maximum flow for the constructed graphs is k, each invocation of maximum flow algorithm takes time $O((n + k + \sum_{i=1}^{k} l_i) \cdot k)$. Inside the enumeration, Algorithm 10 also enumerates each

ballots, which takes time $O(k)$. To summarize, Algorithm 10 takes $O(nmk(n+k+\sum_{i=1}^{k} l_i))$ time, which completes the proof.

Algorithm 11 enumerates each voter and invokes Algorithm 10. Therefore, the time complexity of Algorithm 11 is $O(n^2mk(n+k+\sum_{i=1}^{k} l_i))$, which completes the proof. □

We further simplify the time complexity expressions as follows. The number of ballots is no more than the number of voters, i.e, $k = O(n)$. As a result, Algorithm 10 takes $O(nmk(n+\sum_{i=1}^{k} l_i))$ time. Later on, we will show that the number of mixins for each ballot is small, i.e., $l_i = O(1)$. Therefore, Algorithm 10 takes $O(n^2mk)$ time optimistically. Similarly, Algorithm 11 takes $O(n^3mk)$ time optimistically.

Corollary 2. *If the number of mixins for each ballot is considered as a constant, i.e., $l_i = O(1)$ for all l_i, then, Algorithm 10 and Algorithm 11 takes $O(n^2mk)$ and $O(n^2mk)$, respectively.*

5.3.2 Number of Compromised Voters v.s. Number of Mixins for Traditional Approaches

In this subsection, we conduct experiments to examine to which degree the anonymity is compromised by the traditional linkable ring signature. We fix the number of voters to be 100 and evaluate how the number of voters, ballots, and candidates affects the number of compromised voters. For each parameter setting, we generate ballots with random choices of candidates and mixins and count the number of compromised voters according to Algorithm 10. Such experiments are repeated for 100 times, and the average value is displayed.

In Figure 5.4, we fix the number of voters and candidates to be 100 and 2, respectively, vary the number of mixins from 2 to 17 with a step of 1, and vary the number of ballots from 60 to 100 with a step of 10. The anonymity of up to 94.39% of the voters

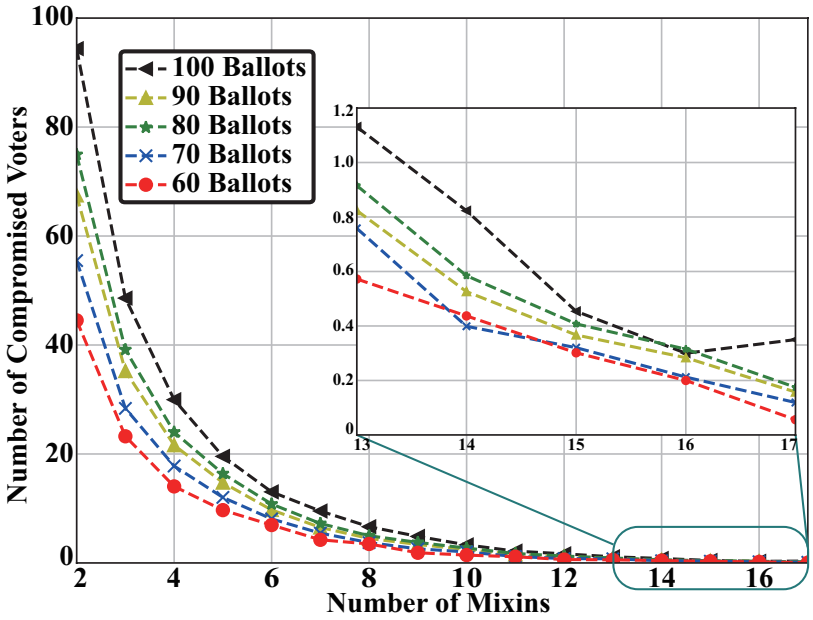

Figure 5.4: Number of compromised voters v.s. number of ballots and mixins

is compromised when the number of ballots and mixins are 100 and 2, respectively. Indeed, the number of compromised users decreases significantly when the number of mixins increases from 2 to 17. Moreover, a smaller number of ballots indicates a smaller number of compromised voters. However, there are still some compromised voters even there are as many as 17 mixins. That is, 17 mixins are not enough to guarantee anonymity, and we can hardly tell whether it can guarantee anonymity for sure with more mixins. Note that it requires more computational power to generate and verify a signature with more mixins. As a comparison, the average number of mixins in Monero transactions is 3, 5, and 11 since September 2016, September 2017, and now respectively [118]. Therefore, it is of high demand for a provably correct anonymity validation algorithm and an intelligent mixin selection algorithm.

In Figure 5.5, we fix the number of voters and ballots to be 100 and 80, respectively, vary the number of mixins from 2 to 17, and vary the number of candidates from 2 to 6. As we can see, the number of compromised users does not change too much when the number of candidates increases from 3 to 6. Hence, the influence of the number of

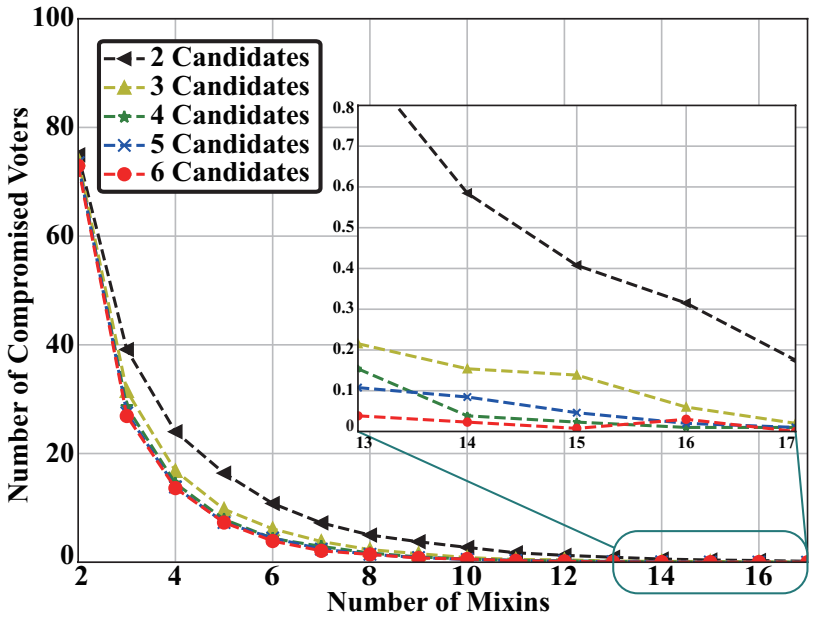

Figure 5.5: Number of compromised voters v.s. number of candidates and mixins

candidates to the number of compromised voters is limited. Last but not least, there are still some compromised voters even there are up to 17 mixins and 6 candidates.

5.3.3 Number of Mixins for Dynamic Ring Signature

In this subsection, we conduct experiments to evaluate the number of mixins generated by dynamic ring signature. On the one hand, we demonstrate how the number of candidates and ballots influences the number of required mixins. For each parameter setting, we generate ballots with randoms choice of candidates, and generate the mixins according to Algorithm 11. Such an experiment is repeated for 100 times, and the average value is displayed. One the other hand, with a fixed number of 2 candidates and 80 ballots, we run the experiment for 200 times and show the histogram of the number of mixins.

Figure 5.6 shows how the average number of mixins in dynamic ring signature varies for different numbers of candidates and ballots. We can see that when the number

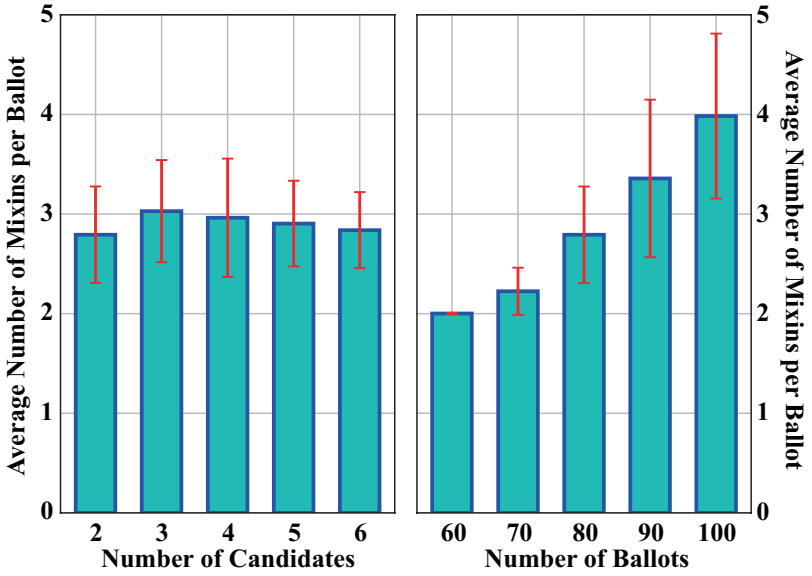

Figure 5.6: Number of mixins per ballot v.s. number of ballots and candidates

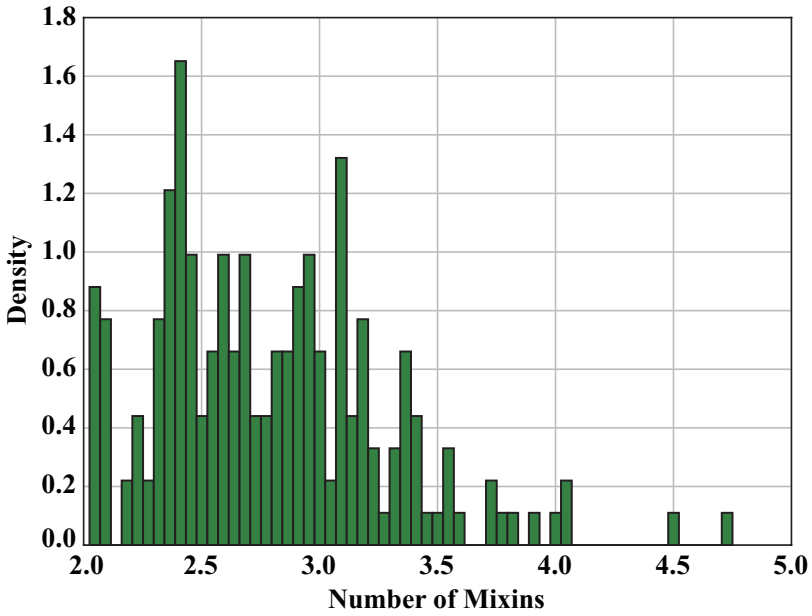

Figure 5.7: Histogram of number of mixin

of ballots is 80, the number of generated mixins remains nearly unchanged and is around 3 as the number of candidate increase. We can conclude that dynamic ring signature is superior to the traditional approach by reducing the number of mixins from 15 to 3. As the number of ballots increases from 60 to 100 and we fix the number of candidates to be 2, we can see that the number of required mixins slightly increases from 2 to 4 approximately. We can draw the conclusion that the number of required mixins for dynamic ring signature is stable regardless of the number of ballots. Figure 5.7 shows the histogram of the number of required mixins when the number of candidates is 2, and the number of ballots is 80. The figure shows that the number of mixins is less than 4 for almost all the time, which indicates the stability of dynamic ring signature.

5.3.4 Time Consumption for Dynamic Ring Signature

In this subsection, we conduct experiments to evaluate the time efficiency of dynamic ring signature. On one hand, we demonstrate how the time consumption is influenced by the number of candidates and ballots. For each parameter setting, we generate ballots with random choices of candidates, and generate the mixins according to Algorithm 11. Such an experiment is repeated for 100 times, and the average value is displayed. One the other hand, with a fixed number of 2 candidates and 80 ballots, we run the experiment for 200 times and show the histogram of the time consumption.

Figure 5.8 shows how the time consumption for each ballot with different numbers of candidates and ballots. When the number of ballots is 80, the average time to generate each ballot remains nearly unchanged and is less than 5 milliseconds as the number of candidate increase. We conclude that dynamic ring signature is time-efficient regardless of the number of candidates. As the number of ballots increases from 60 to 100, we can see that the average time consumption of each ballot increases from 1 to 24 milliseconds approximately. Such a phenomenon is regular because the

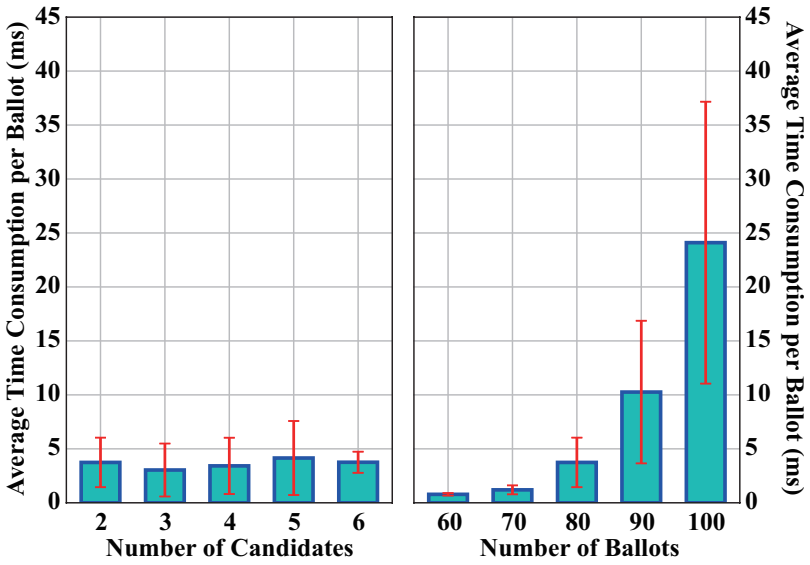

Figure 5.8: Time consumption per ballot v.s. number of ballots and candidates

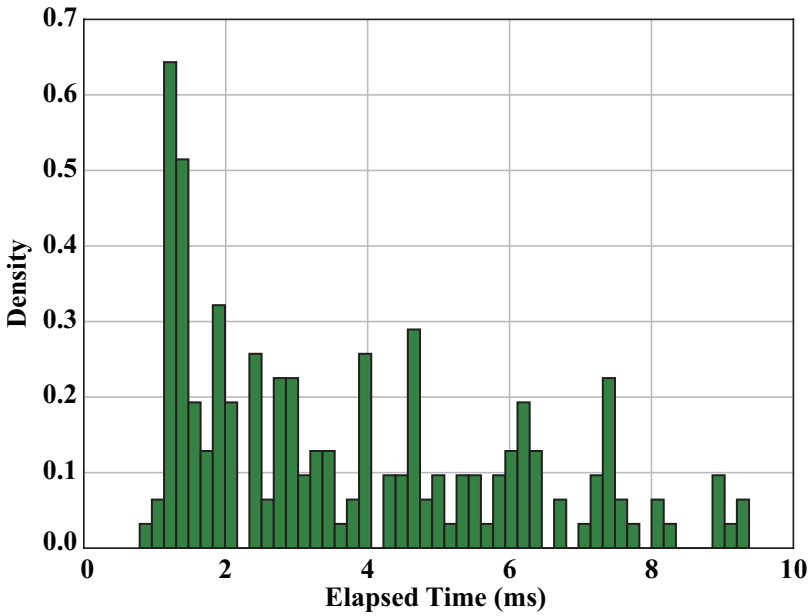

Figure 5.9: Histogram of time consumption

constructed graph is more complicated when there are more ballots. Figure 5.9 shows the histogram of the time consumption when the number of candidates is 2, and the number of ballots is 80. The figure shows that the time consumption for each ballot is less than 10 milliseconds for all the cases, which indicates the stability in terms of time efficiency of dynamic ring signature.

5.3.5 Concurrent Ballot Submission

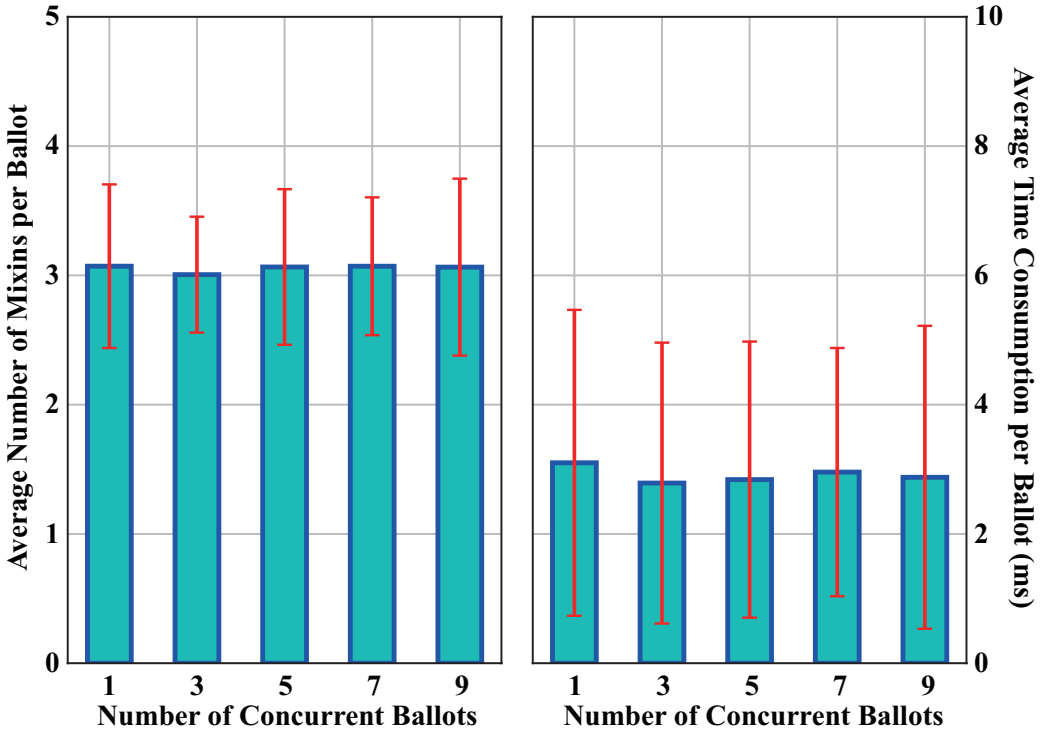

Figure 5.10: Average number of mixins and time consumption per ballot v.s. number of concurrent ballots

Algorithm 11 processes a single raw ballot for a given e-voting system. Furthermore, the e-voting system will be renewed after processing and including the raw ballot. Therefore, it seems Algorithm 11 can only process ballots sequentially. Indeed, the provable anonymity is provided only when the ballots come in sequence. Naturally, it comes to the question of whether Algorithm 11 can process concurrent ballot submis-

sions. Figure 5.10 shows the experimental results when there are 1 to 9 concurrent ballots with a step of 2 ballots. As we can see, the number of required mixins and the time consumption remains nearly unchanged with different numbers of concurrent ballots, which indicates that dynamic ring signature works well even if multiple ballots are submitted simultaneously.

5.4 Related Work

5.4.1 E-voting systems

E-voting systems have been studied for decades and have been employed in many countries for national, statewide and municipal elections. Before the rising of blockchain technology, the research community has been devoted to providing *verifiability*. That is, to guarantee that ballots from the voters have been counted indeed [37]. Some recent works include [36], [91], [60], and [90]. In these systems, The definitions of verifiability differ in many aspects, such as the classes of protocols they capture and the underlying models and assumptions. However, verifiability seems to be a contradiction to anonymization for these approaches [38]. That is, these systems fail to anonymize voters.

Blockchain, a technology for trustless data storage, shows great potential in e-voting because the data on blockchain is auditable in nature [87][54][13]. Here, auditability shares the same meaning as verifiability in traditional e-voting systems. The first application of blockchain technology is Bitcoin, which brings out Bitcoin-based e-voting systems [187][18][178]. Because users are pseudo-anonymous in Bitcoin, these works cannot provide anonymity. Later on, the researchers try to develop BEV based on general blockchain platforms such as Ethereum [116][156] and Hyperledger Fabric [179][92]. Meanwhile, cryptographic techniques such as zero-knowledge proof [142], blind signature [26], secrete sharing [156], and linkable ring signature [123][38] are

employed for anonymity. However, existing BEV systems incur various issues. In BEV with zero-knowledge proof [116], the signatures are computationally heavy to verify. In BEV using blind signature [62] or secrete sharing [156], the identities of the voters can be arbitrarily faked. In BEV with linkable ring signature [179][101], the identities of some voters can be compromised from a set of submitted ballots. To summarize, existing BEV systems fail to consider auditability and anonymity at the same time.

5.4.2 Linkable Ring Signature

Linkable ring signature was first proposed by Liu et al. in 2004 [110]. There are many variants in different types of cryptosystems with different features. In this chapter, we consider public key-based linkable ring signature [123][151][110][109] only. Identity-based [7] and certificate-based [6] linkable ring signatures are not considered because they require a private key generator to issue user keys, which contradicts to the decentralized concept of blockchain.

In particular, public key-based linkable ring signature can be classified into static and dynamic group linkable ring signature. In static group ring signature, the signer is supposed to use mixins within a static group of public keys. On the contrary, dynamic group linkable ring signature, or one-time linkable ring signature [122], can use arbitrary mixins. In either schema, the mixins are randomly selected without consideration of provable anonymity.

5.5 Chapter Summary

In this chapter, we present Roshan, a blockchain-based e-voting system that provides auditability, immutability, and anonymity. Inside Roshan, registration information and ballot information are submitted to blockchain and cannot be tampered. The key

novelty in Roshan lies in dynamic ring signature, an anonymous mechanism to provide provable anonymity. In particular, dynamic ring signature consists of two parts, a maximum flow-based algorithm for anonymity validation and a heuristic algorithm for mixin selection. The experiments indicate that dynamic ring signature uses very few mixins to provide provable anonymity, demonstrating its superiority over traditional ring signature approaches. It is notable that dynamic ring signature is not limited to e-voting and can be used in other applications such as cryptocurrencies.

Chapter 6

Conclusion and Future Directions

In this thesis, we identify the issues in big data sharing especially blockchain-based big data sharing, and address several critical issues. First, we present a comprehensive survey of big data sharing in chapter 2, which helps identification of the challenges and state-of-the-art solutions of big data sharing. Second, in chapter 3, we propose a fairness-based transaction packing algorithm in the consensus layer, which can improve the quality of service when providing blockchain-based big data sharing services. Third, in chapter 5, we propose a dynamic ring signature scheme with provable anonymity for the data sharers and sharees. Finally, in chapter 4, we propose an efficient multi-keyword search algorithm to preserve the privacy when the data sharees are using the data. We believe the comprehensive survey and high-performance solutions presented in this thesis may serve as a preliminary step towards the broad applications of blockchain-based big data sharing and attract extensive attention from both the academia and industries.

In the future, there are two directions concerning the technical and application levels. On the one hand, there are still many challenging issues that remain to be addressed in blockchain-based big data sharing. To name a few, high-performance consensus algorithms, fine-grained access control mechanisms, methods to prevent unautho-

rized big data re-sharing, etc. are demanded to improve the performance and make blockchain-based big data sharing more practical. On the other hand, blockchain-based big data sharing should be deployed in more real-world applications besides supply chain management and healthcare information management. We believe the solutions of blockchain-based big data sharing will be significantly improved and have a wide range of applications in the future.

References

[1] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In *Annual international cryptology conference*, pages 205–222. Springer, 2005.

[2] Paulo Sérgio Almeida, Carlos Baquero, Nuno Preguiça, and David Hutchison. Scalable bloom filters. *Information Processing Letters*, 101(6):255–261, 2007.

[3] Marcelo Arenas, Pablo Barceló, Leonid Libkin, and Filip Murlak. *Foundations of data exchange*. Cambridge University Press, 2014.

[4] Judie Attard, Fabrizio Orlandi, Simon Scerri, and Sören Auer. A systematic review of open government data initiatives. *Government Information Quarterly*, 32(4):399–418, 2015.

[5] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. A survey of attacks on ethereum smart contracts (sok). In *Springer POST*, pages 164–186, 2017.

[6] Man Ho Au, Joseph K Liu, Willy Susilo, and Tsz Hon Yuen. Certificate based (linkable) ring signature. In *International Conference on Information Security Practice and Experience*, pages 79–92. Springer, 2007.

[7] Man Ho Au, Joseph K Liu, Willy Susilo, and Tsz Hon Yuen. Secure id-based linkable and revocable-iff-linked ring signature with constant-size construction. *Theoretical Computer Science*, 469:1–14, 2013.

[8] David E Bakken, R Rarameswaran, Douglas M Blough, Andy A Franz, and Ty J Palmer. Data obfuscation: Anonymity and desensitization of usable data sets. *IEEE Security & Privacy*, 2(6):34–41, 2004.

[9] Carlo Batini, Monica Scannapieco, et al. Data and information quality. *Cham, Switzerland: Springer International Publishing. Google Scholar*, page 43, 2016.

[10] Paul C Bauer, Florian Keusch, and Frauke Kreuter. Trust and cooperative behavior: Evidence from the realm of data-sharing. *PloS one*, 14(8):e0220115, 2019.

[11] Elizabeth A Bell, Lucila Ohno-Machado, and M Adela Grando. Sharing my health data: a survey of data sharing preferences of healthy individuals. In *AMIA Annual Symposium Proceedings*, volume 2014, page 1699. American Medical Informatics Association, 2014.

[12] Susan Bell, Josh Benaloh, Michael D Byrne, Dana DeBeauvoir, Bryce Eakin, Philip Kortum, Neal McBurnett, Olivier Pereira, Philip B Stark, Dan S Wallach, et al. Star-vote: A secure, transparent, auditable, and reliable voting system. In *2013 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 13)*, 2013.

[13] Emanuele Bellini, Paolo Ceravolo, and Ernesto Damiani. Blockchain-based e-vote-as-a-service. In *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, pages 484–486. IEEE, 2019.

[14] J Benet and N Greco. Filecoin: A decentralized storage network. *Protoc. Labs*, pages 1–36, 2018.

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

[16] Elisa Bertino and Elena Ferrari. Big data security and privacy. In *A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years*, pages 425–439. Springer, 2018.

[17] Elisa Bertino and Ravi Sandhu. Database security-concepts, approaches, and challenges. *IEEE Transactions on Dependable and secure computing*, 2(1):2–19, 2005.

[18] Stefano Bistarelli, Marco Mantilacci, Paolo Santancini, and Francesco Santini. An end-to-end voting-system based on bitcoin. In *Proceedings of the Symposium on Applied Computing*, pages 1836–1841, 2017.

[19] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer, 2001.

[20] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography Conference*, pages 253–273. Springer, 2011.

[21] Raphael Bost. $\sum o\varphi o\varsigma$: Forward secure searchable encryption. In *ACM CCS*, pages 1143–1154, 2016.

[22] Yingyi Bu, Adawaichee Fu, Raymond Chi Wing Wong, Lei Chen, and Jiuyong Li. Privacy preserving serial data publishing by role composition. *Proceedings of the VLDB Endowment*, 1(1):845, 2008.

[23] Chengjun Cai, Jian Weng, Xingliang Yuan, and Cong Wang. Enabling reliable keyword search in encrypted decentralized storage with fairness. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2018.

[24] K Selçuk Candan, Huan Liu, and Reshma Suvarna. Resource description framework: metadata and its applications. *ACM SIGKDD Explorations Newsletter*, 3(1):6–19, 2001.

[25] Ning Cao, Cong Wang, Ming Li, Kui Ren, and Wenjing Lou. Privacy-preserving multi-keyword ranked search over encrypted cloud data. *IEEE Transactions on parallel and distributed systems*, 25(1):222–233, 2013.

[26] David Chaum. Blind signatures for untraceable payments. In *Advances in cryptology*, pages 199–203. Springer, 1983.

[27] Jinchuan Chen and Yunzhi Xue. Bootstrapping a blockchain based ecosystem for big data exchange. In *2017 IEEE international congress on big data (bigdata congress)*, pages 460–463. IEEE, 2017.

[28] Lanxiang Chen, Wai-Kong Lee, Chin-Chen Chang, Kim-Kwang Raymond Choo, and Nan Zhang. Blockchain based searchable encryption for electronic health record sharing. *Future Generation Computer Systems (FGCS)*, 2019.

[29] Min Chen, Shiwen Mao, and Yunhao Liu. Big data: A survey. *Mobile networks and applications*, 19(2):171–209, 2014.

[30] Min Chen, Yongfeng Qian, Jing Chen, Kai Hwang, Shiwen Mao, and Long Hu. Privacy protection and intrusion avoidance for cloudlet-based medical data sharing. *IEEE transactions on Cloud computing*, 2016.

[31] Shimin Chen, Phillip B Gibbons, Suman Nath, et al. Rethinking database algorithms for phase change memory. In *Cidr*, volume 11, page 5th, 2011.

[32] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 41–50. IEEE, 1995.

[33] Francis S Collins and Harold Varmus. A new initiative on precision medicine. *New England Journal of Medicine*, 372(9):793–795, 2015.

[34] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[35] Véronique Cortier, Constantin Cătălin Drăgan, François Dupressoir, Benedikt Schmidt, Pierre-Yves Strub, and Bogdan Warinschi. Machine-checked proofs of privacy for electronic voting protocols. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 993–1008. IEEE, 2017.

[36] Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachene. Election verifiability for helios under weaker trust assumptions. In *European Symposium on Research in Computer Security*, pages 327–344. Springer, 2014.

[37] Véronique Cortier, David Galindo, Ralf Küsters, Johannes Mueller, and Tomasz Truderung. Sok: Verifiability notions for e-voting protocols. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 779–798. IEEE, 2016.

[38] Véronique Cortier and Joseph Lallemand. Voting: You can't have privacy without individual verifiability. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 53–66, 2018.

[39] Chris Culnane, Peter YA Ryan, Steve Schneider, and Vanessa Teague. vvote: a verifiable voting system. *ACM Transactions on Information and System Security (TISSEC)*, 18(1):1–30, 2015.

[40] Chris Culnane and Steve Schneider. A peered bulletin board for robust use in verifiable voting systems. In *2014 IEEE 27th Computer Security Foundations Symposium*, pages 169–183. IEEE, 2014.

[41] Philip M Davis, Bruce V Lewenstein, Daniel H Simon, James G Booth, and Mathew JL Connolly. Open access publishing, article downloads, and citations: randomised controlled trial. *BMj*, 337, 2008.

[42] Wenliang Du and Zhijun Zhan. Building decision tree classifier on private data. In *Proceedings of the IEEE international conference on Privacy, security and data mining*, pages 1–8, 2002.

[43] Cynthia Dwork. Differential privacy. In *International Colloquium on Automata, Languages, and Programming*, 2006.

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

[45] Cynthia Dwork. The differential privacy frontier. In *Theory of Cryptography Conference*, pages 496–502. Springer, 2009.

[46] Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 371–380, 2009.

[47] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N Rothblum, and Salil Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 381–390, 2009.

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

[49] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Commun. ACM*, 2018.

[50] Ruogu Fang, Samira Pouyanfar, Yimin Yang, Shu-Ching Chen, and SS Iyengar. Computational health informatics in the big data age: a survey. *ACM Computing Surveys (CSUR)*, 49(1):1–36, 2016.

[51] Adam R Ferguson, Jessica L Nielson, Melissa H Cragin, Anita E Bandrowski, and Maryann E Martone. Big data from small data: data-sharing in the'long tail'of neuroscience. *Nature neuroscience*, 17(11):1442–1447, 2014.

[52] Benjamin CM Fung, Ke Wang, and S Yu Philip. Anonymizing classification data for privacy preservation. *IEEE transactions on knowledge and data engineering*, 19(5):711–725, 2007.

[53] John Gantz and David Reinsel. Extracting value from chaos. *IDC iview*, 1142(2011):1–12, 2011.

[54] Kanika Garg, Pavi Saraswat, Sachin Bisht, Sahil Kr Aggarwal, Sai Krishna Kothuri, and Sahil Gupta. A comparitive analysis on e-voting system using blockchain. In *2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)*, pages 1–4. IEEE, 2019.

[55] Mouzhi Ge, Hind Bangui, and Barbora Buhnova. Big data for internet of things: A survey. *Future generation computer systems*, 87:601–614, 2018.

[56] Craig Gentry and Dan Boneh. *A fully homomorphic encryption scheme*, volume 20. Stanford university Stanford, 2009.

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

[58] Jeremy Ginsberg, Matthew H Mohebbi, Rajan S Patel, Lynnette Brammer, Mark S Smolinski, and Larry Brilliant. Detecting influenza epidemics using search engine query data. *Nature*, 457(7232):1012–1014, 2009.

[59] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM (JACM)*, 33(4):792–807, 1986.

[60] Gurchetan S Grewal, Mark D Ryan, Liqun Chen, and Michael R Clarkson. Du-vote: Remote electronic voting with untrusted computers. In *2015 IEEE 28th Computer Security Foundations Symposium*, pages 155–169. IEEE, 2015.

[61] Önder Gürcan, Antonella Del Pozzo, and Sara Tucci-Piergiovanni. On the bitcoin limitations to deliver fairness to users. In *Springer OTM*, 2017.

[62] Freya Sheer Hardwick, Apostolos Gioulis, Raja Naeem Akram, and Konstantinos Markantonakis. E-voting with blockchain: An e-voting protocol with decentralisation and voter privacy. In *2018 IEEE International Conference on Blockchain*, pages 1561–1567. IEEE, 2018.

[63] Brian Hayes. Cloud computing. *Communications of the ACM*, 51(7):9–11, 2008.

[64] Kun He, Jing Chen, Ruiying Du, Qianhong Wu, Guoliang Xue, and Xiang Zhang. Deypos: Deduplicatable dynamic proof of storage for multi-user environments. *IEEE Transactions on Computers*, 65(12):3631–3645, 2016.

[65] Jan-Hinnerk Helms. Previewing audio data, February 12 2009. US Patent App. 11/834,680.

[66] Maurice Herlihy. Blockchains from a distributed computing perspective. *Commun. ACM*, 2019.

[67] Bobby Lee Houtkoop, Chris Chambers, Malcolm Macleod, Dorothy VM Bishop, Thomas E Nichols, and Eric-Jan Wagenmakers. Data sharing in psychology: A survey on barriers and preconditions. *Advances in methods and practices in psychological science*, 1(1):70–85, 2018.

[68] Haibo Hu, Jianliang Xu, Chushi Ren, and Byron Choi. Processing private queries over untrusted data cloud through privacy homomorphism. In *2011 IEEE 27th International Conference on Data Engineering*, pages 601–612. IEEE, 2011.

[69] Haibo Hu, Jianliang Xu, Xizhong Xu, Kexin Pei, Byron Choi, and Shuigeng Zhou. Private search on key-value stores with hierarchical indexes. In *2014 IEEE 30th International Conference on Data Engineering*, pages 628–639. IEEE, 2014.

[70] Shengshan Hu, Chengjun Cai, Qian Wang, Cong Wang, Xiangyang Luo, and Kui Ren. Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization. In *IEEE INFOCOM*, 2018.

[71] Michael Hucka, Andrew Finney, Herbert M Sauro, Hamid Bolouri, John C Doyle, Hiroaki Kitano, Adam P Arkin, Benjamin J Bornstein, Dennis Bray, Athel Cornish-Bowden, et al. The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003.

[72] Lee Hutchinson. Solid-state revolution: in-depth on how ssds really work. *Ars Technica*, 2012.

[73] Adam Jacobs. The pathologies of big data. *Commun. ACM*, 52(8):36–44, 2009.

[74] Raj Jain, Dah-Ming Chiu, and William R Hawe. *A quantitative measure of fairness and discrimination for resource allocation in shared computer system*. Eastern Research Lab, 1984.

[75] Shan Jiang, Jiannong Cao, Yang Liu, Jinlin Chen, and Xuefeng Liu. Programming large-scale multi-robot system with timing constraints. In *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–9. IEEE, 2016.

[76] Shan Jiang, Jiannong Cao, Julie A McCann, Yanni Yang, Yang Liu, Xiaoqing Wang, and Yuming Deng. Privacy-preserving and efficient multi-keyword search over encrypted data on blockchain. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 405–410. IEEE, 2019.

[77] Shan Jiang, Jiannong Cao, Jia Wang, Milos Stojmenovic, and Julien Bourgeois. Uniform circle formation by asynchronous robots: A fully-distributed approach. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–9. IEEE, 2017.

[78] Shan Jiang, Jiannong Cao, Hanqing Wu, and Yanni Yang. Fairness-based packing of industrial iot data in permissioned blockchains. *IEEE Transactions on Industrial Informatics*, 2020.

[79] Shan Jiang, Jiannong Cao, Hanqing Wu, Yanni Yang, Mingyu Ma, and Jianfei He. Blochie: a blockchain-based platform for healthcare information exchange. In *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 49–56. IEEE, 2018.

[80] Shan Jiang, Jiannong Cao, Juncen Zhu, and Yinfeng Cao. Polychain: a generic blockchain as a service platform. In *2021 International Conference on Blockchain and Trustworthy Systems (BlockSys)*, pages 1–14. Springer, 2021.

[81] Shan Jiang, Junbin Liang, Jiannong Cao, and Rui Liu. An ensemble-level programming model with real-time support for multi-robot systems. In *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 1–3. IEEE, 2016.

[82] Shan Jiang, Junbin Liang, Jiannong Cao, Jia Wang, Jinlin Chen, and Zhixuan Liang. Decentralized algorithm for repeating pattern formation by multiple robots. In *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 594–601. IEEE, 2019.

[83] Taeho Jung, Xiang-Yang Li, Wenchao Huang, Jianwei Qian, Linlin Chen, Junze Han, Jiahui Hou, and Cheng Su. Accounttrade: Accountable protocols for big data trading against dishonest consumers. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[84] Seny Kamara and Tarik Moataz. Boolean searchable symmetric encryption with worst-case sub-linear complexity. In *Springer EUROCRYPT*, pages 94–124, 2017.

[85] Seny Kamara, Charalampos Papamanthou, and Tom Roeder. Dynamic searchable symmetric encryption. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 965–976, 2012.

[86] Vamsee Kasavajhala. Solid state drive vs. hard disk drive price and performance study. *Proc. Dell Tech. White Paper*, pages 8–9, 2011.

[87] Kashif Mehboob Khan, Junaid Arshad, and Muhammad Mubashir Khan. Investigating performance constraints for blockchain based secure e-voting system. *Future Generation Computer Systems*, 105:13–26, 2020.

[88] Nawsher Khan, Ibrar Yaqoob, Ibrahim Abaker Targio Hashem, Zakira Inayat, Waleed Kamaleldin Mahmoud Ali, Muhammad Alam, Muhammad Shiraz, and Abdullah Gani. Big data: survey, technologies, opportunities, and challenges. *The scientific world journal*, 2014, 2014.

[89] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*. Springer, 2017.

[90] Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. Demos-2: scalable e2e verifiable elections without random oracles. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 352–363, 2015.

[91] Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. End-to-end verifiable elections in the standard model. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 468–498. Springer, 2015.

[92] Denis Kirillov, Vladimir Korkhov, Vadim Petrunin, Mikhail Makarov, Ildar M Khamitov, and Victor Dostov. Implementation of an e-voting scheme using hyperledger fabric permissioned blockchain. In *International Conference on Computational Science and Its Applications*, pages 509–521. Springer, 2019.

[93] Bryan Klimt and Yiming Yang. The enron corpus: A new dataset for email classification research. In *Springer ECML*, pages 217–226, 2004.

[94] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *IEEE Symposium on Security and Privacy*, 2016.

[95] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. Hmac: Keyed-hashing for message authentication. *RFC*, 2104:1–11, 1997.

[96] Nir Kshetri and Jeffrey Voas. Blockchain-enabled e-voting. *IEEE Software*, 35(4):95–99, 2018.

[97] Amrit Kumar, Clément Fischer, Shruti Tople, and Prateek Saxena. A traceability analysis of monero's blockchain. In *European Symposium on Research in Computer Security*, pages 153–173. Springer, 2017.

[98] Jiewu Leng, Douxi Yan, Qiang Liu, Kailin Xu, J. Leon Zhao, Rui Shi, Lijun Wei, Ding Zhang, and Xin Chen. Manuchain: Combining permissioned blockchain with a holistic optimization model as bi-level intelligence for smart manufacturing. *IEEE Trans. on Syst. Man Cybern. Syst.*, 2020.

[99] Meng Li, Donghui Hu, Chhagan Lal, Mauro Conti, and Zijian Zhang. Blockchain-enabled secure energy trading with verifiable fairness in industrial internet of things. *IEEE Trans. on Ind. Informatics*, 2020.

[100] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. Closeness: A new privacy measure for data publishing. *IEEE Transactions on Knowledge and Data Engineering*, 22(7):943–956, 2009.

[101] Peng Li and Junzuo Lai. Lat-voting: Traceable anonymous e-voting on blockchain. In *International Conference on Network and System Security*, pages 234–254. Springer, 2019.

[102] Tiancheng Li, Ninghui Li, Jian Zhang, and Ian Molloy. Slicing: A new approach for privacy preserving data publishing. *IEEE transactions on knowledge and data engineering*, 24(3):561–574, 2010.

[103] Xiaoqi Li, Peng Jiang, Ting Chen, Xiapu Luo, and Qiaoyan Wen. A survey on the security of blockchain systems. *Future Generation Computer Systems (FGCS)*, 2017.

[104] Fan Liang, Wei Yu, Dou An, Qingyu Yang, Xinwen Fu, and Wei Zhao. A survey on big data market: Pricing, trading and protection. *IEEE Access*, 6:15132–15154, 2018.

[105] Dan Lin, Prathima Rao, Elisa Bertino, Ninghui Li, and Jorge Lobo. Exam: a comprehensive environment for the analysis of access control policies. *International Journal of Information Security*, 9(4):253–273, 2010.

[106] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

[107] Jian Liu, Wenting Li, Ghassan O Karame, and N Asokan. Toward fairness of cryptocurrency payments. *IEEE Secur. Priv.*, 2018.

[108] Jian Liu, Wenting Li, Ghassan O Karame, and N Asokan. Scalable byzantine consensus via hardware-assisted secret sharing. *IEEE Trans. on Computers*, 2019.

[109] Joseph K Liu, Man Ho Au, Willy Susilo, and Jianying Zhou. Linkable ring signature with unconditional anonymity. *IEEE Transactions on Knowledge and Data Engineering*, 26(1):157–165, 2013.

[110] Joseph K Liu, Victor K Wei, and Duncan S Wong. Linkable spontaneous anonymous group signature for ad hoc groups. In *Australasian Conference on Information Security and Privacy*, pages 325–335. Springer, 2004.

[111] Joseph K Liu and Duncan S Wong. Linkable ring signatures: Security models and new schemes. In *International Conference on Computational Science and Its Applications*, pages 614–623. Springer, 2005.

[112] Xiulong Liu, Jiuwu Zhang, Shan Jiang, Yanni Yang, Keqiu Li, Jiannong Cao, and Jiangchuan Liu. Accurate localization of tagged objects using mobile rfid-augmented robots. *IEEE Transactions on Mobile Computing*, 2019.

[113] Xueqiao Liu, Guomin Yang, Yi Mu, and Robert Deng. Multi-user verifiable searchable symmetric encryption for cloud storage. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2018.

[114] Ruiqu Ma and Patrick TI Lam. Investigating the barriers faced by stakeholders in open data development: A study on hong kong as a "smart city". *Cities*, 92:36–46, 2019.

[115] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3–es, 2007.

[116] Patrick McCorry, Siamak F Shahandashti, and Feng Hao. A smart contract for boardroom voting with maximum voter privacy. In *International Conference on Financial Cryptography and Data Security*, pages 357–375. Springer, 2017.

[117] Xianrui Meng, Seny Kamara, Kobbi Nissim, and George Kollios. Grecs: Graph encryption for approximate shortest distance queries. In *ACM CCS*, pages 504–517, 2015.

[118] Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, et al. An empirical analysis of traceability in the monero blockchain. *Proceedings on Privacy Enhancing Technologies*, 2018(3):143–163, 2018.

[119] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Bitcoin White Paper*, 2008.

[120] C Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 116–125, 2001.

[121] Qun Ni, Jorge Lobo, Seraphin Calo, Pankaj Rohatgi, and Elisa Bertino. Automating role-based provisioning by learning from examples. In *Proceedings of the 14th ACM symposium on Access control models and technologies*, pages 75–84, 2009.

[122] Shen Noether. Ring signature confidential transactions for monero. *IACR Cryptol. ePrint Arch.*, 2015:1098, 2015.

[123] Shen Noether, Adam Mackenzie, et al. Ring confidential transactions. *Ledger*, 1:1–18, 2016.

[124] Hyeontaek Oh, Sangdon Park, Gyu Myoung Lee, Jun Kyun Choi, and Sungkee Noh. Competitive data trading model with privacy valuation for multiple stakeholders in iot data markets. *IEEE Internet of Things Journal*, 7(4):3623–3639, 2020.

[125] Alina Oprea, Michael K Reiter, Ke Yang, et al. Space-efficient block storage integrity. In *NDSS*, 2005.

[126] Ahmed Oussous, Fatima-Zahra Benjelloun, Ayoub Ait Lahcen, and Samir Belfkih. Big data technologies: A survey. *Journal of King Saud University-Computer and Information Sciences*, 30(4):431–448, 2018.

[127] Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In *ACM PODC*, 2017.

[128] Agostino Pirovano, ANDREA LEONARDO Lacaita, A Benvenuti, F Pellizzer, S Hudgens, and R Bez. Scaling analysis of phase-change memory technology. In *IEEE International Electron Devices Meeting 2003*, pages 29–6. IEEE, 2003.

[129] Heather A Piwowar, Roger S Day, and Douglas B Fridsma. Sharing detailed research data is associated with increased citation rate. *PloS one*, 2(3):e308, 2007.

[130] Samira Pouyanfar, Yimin Yang, Shu-Ching Chen, Mei-Ling Shyu, and SS Iyengar. Multimedia big data analytics: A survey. *ACM Computing Surveys (CSUR)*, 51(1):1–34, 2018.

[131] Zhiguang Qin, Hu Xiong, Shikun Wu, and Jennifer Batamuliza. A survey of proxy re-encryption for secure data sharing in cloud computing. *IEEE Transactions on Services Computing*, 2016.

[132] Junfei Qiu, Qihui Wu, Guoru Ding, Yuhua Xu, and Shuo Feng. A survey of machine learning for big data processing. *EURASIP Journal on Advances in Signal Processing*, 2016(1):67, 2016.

[133] Vinay Rathi, Kristina Dzara, Cary P Gross, Iain Hrynaszkiewicz, Steven Joffe, Harlan M Krumholz, Kelly M Strait, and Joseph S Ross. Sharing of clinical trial data among trialists: a cross sectional survey. *Bmj*, 345, 2012.

[134] Muhammad Habib Ur Rehman, Chee Sun Liew, Assad Abbas, Prem Prakash Jayaraman, Teh Ying Wah, and Samee U. Khan. Big data reduction methods: A survey. *Data Science and Engineering*, 1(4):265–284, 2016.

[135] Hyun Sook Rhee, Willy Susilo, and Hyun-Jeong Kim. Secure searchable public key encryption scheme against keyword guessing attacks. *IEICE Electronics Express*, 6(5):237–243, 2009.

[136] Ronald L Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 552–565. Springer, 2001.

[137] Philippe Rocca-Serra, Marco Brandizi, Eamonn Maguire, Nataliya Sklyar, Chris Taylor, Kimberly Begley, Dawn Field, Stephen Harris, Winston Hide, Oliver Hofmann, et al. Isa software suite: supporting standards-compliant experimental annotation and enabling curation at the community level. *Bioinformatics*, 26(18):2354–2356, 2010.

[138] Ann M Rogerson and Giselle Basanta. Peer-to-peer file sharing and academic integrity in the internet age. *Handbook of academic integrity*, pages 273–285, 2016.

[139] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 457–473. Springer, 2005.

[140] Yuvraj Sahni, Jiannong Cao, and Shan Jiang. Middleware for multi-robot systems. In *Mission-Oriented Sensor Networks and Systems: Art and Science*, pages 633–673. Springer, 2019.

[141] Daniel Sandler, Kyle Derr, and Dan S Wallach. Votebox: A tamper-evident, verifiable electronic voting system. In *USENIX Security Symposium*, volume 4, page 87, 2008.

[142] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous

payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE, 2014.

[143] Uwe Schwiegelshohn and Ramin Yahyapour. Analysis of first-come-first-serve parallel job scheduling. In *ACM SODA*, 1998.

[144] Rashid Sheikh, Durgesh Kumar Mishra, and Beerendra Kumar. Secure multiparty computation: From millionaires problem to anonymizer. *Information Security Journal: A Global Perspective*, 20(1):25–33, 2011.

[145] Elaine Shi, John Bethencourt, TH Hubert Chan, Dawn Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *2007 IEEE Symposium on Security and Privacy (SP'07)*, pages 350–364. IEEE, 2007.

[146] Ida Sim, Michael Stebbins, Barbara E Bierer, Atul J Butte, Jeffrey Drazen, Victor Dzau, Adrian F Hernandez, Harlan M Krumholz, Bernard Lo, Bernard Munos, et al. Time for nih to lead on data sharing. *Science*, 367(6484):1308–1309, 2020.

[147] Dilpreet Singh and Chandan K Reddy. A survey on platforms for big data analytics. *Journal of big data*, 2(1):8, 2015.

[148] Chris Snijders, Uwe Matzat, and Ulf-Dietrich Reips. " big data": big gaps of knowledge in the field of internet science. *International journal of internet science*, 7(1):1–5, 2012.

[149] Dawn Xiaoding Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *IEEE S&P*, pages 44–55, 2000.

[150] Salmin Sultana and Elisa Bertino. A distributed system for the management of fine-grained provenance. *Journal of Database Management (JDM)*, 26(2):32–47, 2015.

[151] Shi-Feng Sun, Man Ho Au, Joseph K Liu, and Tsz Hon Yuen. Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In *European Symposium on Research in Computer Security*, pages 456–474. Springer, 2017.

[152] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[153] Core Techniques. Technologies for advancing big data science & engineering (bigdata). *URL: http://www. nsf. gov/funding/pgm_summ. jsp*, 2012.

[154] Carol Tenopir, Suzie Allard, Kimberly Douglass, Arsev Umur Aydinoglu, Lei Wu, Eleanor Read, Maribeth Manoff, and Mike Frame. Data sharing by scientists: practices and perceptions. *PloS one*, 6(6):e21101, 2011.

[155] Manolis Terrovitis, John Liagouris, Nikos Mamoulis, and Spiros Skiadopoulos. Privacy preservation by disassociation. *arXiv preprint arXiv:1207.0135*, 2012.

[156] Raylin Tso, Zi-Yuan Liu, and Jen-Ho Hsiao. Distributed e-voting and e-bidding systems based on smart contract. *Electronics*, 8(4):422, 2019.

[157] Huseyin Ulusoy, Pietro Colombo, Elena Ferrari, Murat Kantarcioglu, and Erman Pattuk. Guardmr: fine-grained security policy enforcement for mapreduce systems. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, pages 285–296, 2015.

[158] Jaideep Vaidya and Chris Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 639–644, 2002.

[159] Jaideep Vaidya and Chris Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 206–215, 2003.

[160] Willem G Van Panhuis, Proma Paul, Claudia Emerson, John Grefenstette, Richard Wilder, Abraham J Herbst, David Heymann, and Donald S Burke. A systematic review of barriers to data sharing in public health. *BMC public health*, 14(1):1–9, 2014.

[161] Jia Wang, Jiannong Cao, and Shan Jiang. Fault-tolerant pattern formation by multiple robots: a learning approach. In *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, pages 268–269. IEEE, 2017.

[162] Jia Wang, Jiannong Cao, Milos Stojmenovic, Miao Zhao, Jinlin Chen, and Shan Jiang. Pattern-rl: Multi-robot cooperative pattern formation via deep reinforcement learning. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 210–215. IEEE, 2019.

[163] Ke Wang, Benjamin CM Fung, and S Yu Philip. Handicapping attacker's confidence: an alternative to k-anonymization. *Knowledge and Information Systems*, 11(3):345–368, 2007.

[164] Qian Wang, Kui Ren, Shucheng Yu, and Wenjing Lou. Dependable and secure sensor data storage with dynamic integrity assurance. *ACM Transactions on Sensor Networks (TOSN)*, 8(1):1–24, 2011.

[165] Richard Y Wang and Diane M Strong. Beyond accuracy: What data quality means to data consumers. *Journal of management information systems*, 12(4):5–33, 1996.

[166] Samuel D Warren and Louis D Brandeis. The right to privacy. *Harvard law review*, pages 193–220, 1890.

[167] Alan F Westin. Privacy and freedom atheneum. *New York*, 7:431–453, 1967.

[168] Glenn R Wilcock. Performing nearline storage of a file, July 7 2020. US Patent 10,705,764.

[169] Katherine Wolstencroft, Stuart Owen, Olga Krebs, Quyen Nguyen, Natalie J Stanford, Martin Golebiewski, Andreas Weidemann, Meik Bittkowski, Lihua An, David Shockley, et al. Seek: a systems biology data and model management platform. *BMC systems biology*, 9(1):1–12, 2015.

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

[171] Hanqing Wu, Jiannong Cao, Shan Jiang, Ruosong Yang, Yanni Yang, and Jianfei Hey. Tsar: a fully-distributed trustless data sharing platform. In *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 350–355. IEEE, 2018.

[172] Hanqing Wu, Jiannong Cao, Yanni Yang, Cheung Leong Tung, Shan Jiang, Bin Tang, Yang Liu, Xiaoqing Wang, and Yuming Deng. Data management in supply chain using blockchain: Challenges and a case study. In *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–8. IEEE, 2019.

[173] Feng Xia, Wei Wang, Teshome Megersa Bekele, and Huan Liu. Big scholarly data: A survey. *IEEE Transactions on Big Data*, 3(1):18–35, 2017.

[174] Xiaokui Xiao and Yufei Tao. Anatomy: Simple and effective privacy preservation. In *Proceedings of the 32nd international conference on Very large data bases*, pages 139–150. VLDB Endowment, 2006.

[175] Xiaokui Xiao and Yufei Tao. M-invariance: towards privacy preserving republication of dynamic datasets. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 689–700, 2007.

[176] Sophia Yakoubov, Vijay Gadepally, Nabil Schear, Emily Shen, and Arkady Yerukhimovich. A survey of cryptographic approaches to securing big-data analytics in the cloud. In *2014 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–6. IEEE, 2014.

[177] Haina Ye, Xinzhou Cheng, Mingqiang Yuan, Lexi Xu, Jie Gao, and Chen Cheng. A survey of security and privacy in big data. In *2016 16th international symposium on communications and information technologies (iscit)*, pages 268–272. IEEE, 2016.

[178] Haibo Yi. Securing e-voting based on blockchain in p2p network. *EURASIP Journal on Wireless Communications and Networking*, 2019(1):1–9, 2019.

[179] Bin Yu, Joseph K Liu, Amin Sakzad, Surya Nepal, Ron Steinfeld, Paul Rimba, and Man Ho Au. Platform-independent secure blockchain-based voting system. In *International Conference on Information Security*, pages 369–386. Springer, 2018.

[180] Jia Yu, Kui Ren, Cong Wang, and Vijay Varadharajan. Enabling cloud storage auditing with key-exposure resistance. *IEEE Transactions on Information forensics and security*, 10(6):1167–1179, 2015.

[181] Zuoxia Yu, Man Ho Au, Jiangshan Yu, Rupeng Yang, Qiuliang Xu, and Wang Fat Lau. New empirical traceability analysis of cryptonote-style blockchains. In *International Conference on Financial Cryptography and Data Security*, pages 133–149. Springer, 2019.

[182] Li Yue, Huang Junqin, Qin Shengzhi, and Wang Ruijin. Big data model of security sharing based on blockchain. In *2017 3rd International Conference on Big Data Computing and Communications (BIGCOM)*, pages 117–121. IEEE, 2017.

[183] Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. *Similarity search: the metric space approach*, volume 32. Springer Science & Business Media, 2006.

[184] XJ Zhang and XF Meng. Differential privacy protection for data publishing and analysis. *J Comput*, 4:927–949, 2014.

[185] Yinghui Zhang, Robert Deng, Ximeng Liu, and Dong Zheng. Outsourcing service fair payment based on blockchain and its applications in cloud computing. *IEEE Trans. on Serv. Comput.*, 2018.

[186] Yi Zhao, Haiyang Wang, Hui Su, Liang Zhang, Rui Zhang, Dan Wang, and Ke Xu. Understand love of variety in wireless data market under sponsored data plans. *IEEE Journal on Selected Areas in Communications*, 38(4):766–781, 2020.

[187] Zhichao Zhao and T-H Hubert Chan. How to vote privately using bitcoin. In *International Conference on Information and Communications Security*, pages 82–96. Springer, 2015.

[188] Zhenzhe Zheng, Yanqing Peng, Fan Wu, Shaojie Tang, and Guihai Chen. Arete: On designing joint online pricing and reward sharing mechanisms for mobile data markets. *IEEE Transactions on Mobile Computing*, 19(4):769–787, 2019.