

Article

Privacy and Security Mechanisms for B2B Data Sharing: A Conceptual Framework

Wanying Li, Woon Kwan Tse  and Jiaqi Chen

Department of Information Systems, City University of Hong Kong, Hong Kong 999077, China; wanyinli8-c@my.cityu.edu.hk (W.L.); jchen2244-c@my.cityu.edu.hk (J.C.)

* Correspondence: iswktse@cityu.edu.hk

Abstract: In the age of digitalization, business-to-business (B2B) data sharing is becoming increasingly important, enabling organizations to collaborate and make informed decisions as well as simplifying operations and hopefully creating a cost-effective virtual value chain. This is crucial to the success of modern businesses, especially global business. However, this approach also comes with significant privacy and security challenges, thus requiring robust mechanisms to protect sensitive information. After analyzing the evolving status of B2B data sharing, the purpose of this study is to provide insights into the design of theoretical framework solutions for the field. This study adopts technologies including encryption, access control, data anonymization, and audit trails, with the common goal of striking a balance between facilitating data sharing and protecting data confidentiality as well as data integrity. In addition, emerging technologies such as homomorphic encryption, blockchain, and their applicability as well as advantages in the B2B data sharing environment are explored. The results of this study offer a new approach to managing complex data sharing between organizations, providing a strategic mix of traditional and innovative solutions to promote secure and efficient digital collaboration.

Keywords: data sharing; blockchain; B2B architecture; privacy; security



Citation: Li, W.; Tse, W.K.; Chen, J. Privacy and Security Mechanisms for B2B Data Sharing: A Conceptual Framework. *Information* **2024**, *15*, 308. <https://doi.org/10.3390/info15060308>

Academic Editors: Jose de Vasconcelos, Hugo Barbosa and Carla Cordeiro

Received: 4 May 2024
Revised: 22 May 2024
Accepted: 23 May 2024
Published: 26 May 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In the era of big data, enterprises' data have become a strategic resource to promote global economic progress. The exchange and sharing of data for collaboration, informed decision-making, and sustainable operation are becoming increasingly important. Business-to-business (B2B) data sharing enables organizations to leverage the vast amount of information available, gaining valuable insights and driving innovation. Shared access to business data lays the foundation for service processes for companies within the same industry and even across industries and countries. According to a study by Gartner [1], sharing data externally by data and analytics leaders results in three times the measurable economic benefits compared to those who do not share data. In addition, from the value chain perspective, data sharing in this model has significant benefits, such as helping companies expand their business, improve operational efficiency, and improve customer service.

However, despite the profitable practice, there are still plenty of valid reasons for companies to keep their data closed and avoid sharing the data with third parties, with concerns around privacy, security, access control, and data governance. Traditional methods lack robust security, exposing sensitive business data to cyber threats and unauthorized access while posing significant privacy risks. Existing practices are inadequate in efficiently managing the diverse data storage architecture and increasing data volumes, leading to storage, retrieval, and overall data management difficulties. These inefficiencies hinder business operations and are compounded by a lack of transparency and traceability in auditing data transactions. Moreover, the rigidity and complexity of access control within traditional systems fail to meet the dynamic needs of businesses, and there are substantial

challenges in ensuring the integrity and consistency of data, especially in environments with frequent updates. Compliance with data sharing and privacy standards like General Data Protection Regulation (GDPR) adds another layer of complexity, requiring substantial resources and effort from businesses [2].

Recognizing the urgent need for privacy protection technologies and enhanced security protocols, this study aims to explore the privacy concepts and security mechanisms of B2B data sharing. Our main contribution is to propose a reference architecture for data sharing technologies that can be widely adopted based on enterprise infrastructure. It identifies fundamental security requirements and discusses potential solutions to achieve these goals while highlighting open challenges in this domain. The purpose of this paper is to present a conceptual framework through which we analyze and discuss our approach before conducting field tests. This study is expected to bridge the gap between current practices and emerging challenges in B2B data sharing, offering a comprehensive and viable solution that addresses current issues and lays a foundation for future research in this domain.

The rest of this paper is structured as follows: Section 2 presents the related work. Section 3 is about the significance of this research. Section 4 is the introduction to the theoretical basis of the paper. Sections 5–7 introduce the proposed data sharing system framework and analyze its characteristics. Finally, Section 8 summarizes and prospects the paper.

2. Related Work

The current landscape of B2B data sharing among organizations is complex and challenging. One of the primary challenges in B2B data sharing revolves around ensuring data security and privacy while maintaining accessibility and usability. Blockchain's integrity, immutability, decentralization, and verifiability are key strengths for data sharing in multi-party scenarios [3]. Some studies have explored the utilization of blockchain technology as a potential solution. J. Chi et al. [4] proposed a blockchain-based data sharing scheme, which combines identity authentication, Hyperledger Fabric, and community detection algorithms, effectively ensuring security and efficiency in industrial IoT data sharing. Xuan et al. [5] propose a dynamic incentive model for data sharing using blockchain with smart contracts, enabling trust and automated transactions between large numbers of users. A novel approach to data sharing was introduced by Al-Zahrani [6] in the form of a subscription-based model that utilizes the advanced features of blockchain technology. The proposed Data as a Service (DaaS) model has the potential to offer significant benefits to businesses seeking secure and efficient methods of data sharing. However, blockchain's limitations in scalability and its difficulty in storing vast amounts of data have been acknowledged as significant drawbacks [7].

To address these issues, alternative approaches such as off-chain storage solutions have been proposed. Cheng Xu et al. [8] propose SlimChain, a novel blockchain system that scales transactions through off-chain storage and parallel processing. Kete Wang et al. [9] propose a three-tier system architecture for efficient and transparent data sharing, storing hashing and response records on the blockchain and original data in an off-chain database, a concept that partially inspired our framework's architecture.

The migration of data to the cloud has become a common practice in recent times due to the advanced cloud infrastructure, which offers higher accessibility, lower response time, and more cost efficiency [10]. Despite the significant benefits of data sharing in cloud computing, the outsourcing of data deprives users of direct control, which gives rise to security concerns and poses challenges [11,12]. Kotha et al. [13] explored various problems and challenges of data sharing in cloud environments, highlighting the need for robust encryption and access control mechanisms. Song et al. [14] further examined and compared current encryption and key management techniques in cloud storage, providing a foundation for the encryption strategy employed in our framework.

Access control in B2B frameworks has been thoroughly investigated in works such as those by Gai et al. [15], who proposed a blockchain-based access control scheme using role-based access control (RBAC) specifically tailored for lightweight data sharing among different organizations. Xu et al. [16] proposed a blockchain-based secure data sharing platform with fine-grained access control (BSDS-FA) to prevent data leakage. Yang et al. [17] present a ciphertext-policy attribute-based conditional proxy re-encryption (CPRE) scheme, which allows efficient user revocation and resource-constrained devices to access cloud data. The granular access control in our framework takes cues from these studies, aiming to provide fine-grained permissions management.

Various advanced technologies have been used to facilitate secure data sharing. Proxy re-encryption as a tool for secure data sharing has been gaining traction. The survey by Qin et al. [18] laid the groundwork for using proxy re-encryption in secure data sharing, which has been adapted in our framework to streamline access delegation. Additionally, the application of homomorphic encryption in data sharing has been explored by researchers like Zhu et al. [19], whose insights into practical homomorphic encryption applications have influenced the privacy-preserving aspects of our framework.

Despite the extensive research on data exchange mechanisms, there is still a significant gap in the literature regarding a comprehensive theoretical framework that covers data storage, security, privacy protection, etc. Many existing studies only focus on specific aspects of B2B data sharing and do not consider the need for an integrated approach that addresses the interconnected nature of these components. Therefore, there is an urgent need for a theoretical framework that provides a comprehensive solution to the multifaceted challenges inherent in B2B data sharing, facilitating the development of more robust and effective solutions.

3. Research Significance

In this study, after analyzing the need to have a secure mechanism in B2B data sharing, our most significant contribution is to propose a bullet-proof three-layer B2B data sharing framework while maintaining a reasonable degree of accessibility. It is based on blockchain technology, offering a comprehensive solution to the diverse challenges in B2B data sharing. This framework comprises a data storage layer, a privacy preservation layer, and an access control layer. This layered approach is designed to provide a comprehensive solution that balances the need for data accessibility with security and privacy requirements. The framework is grounded in blockchain technology, integrating emerging technologies such as distributed hash table, homomorphic encryption, and proxy re-encryption. The proposed conceptual framework has the following features:

- Scalability and adaptability—a scalable and adaptable framework is essential with increasing data volumes. Blockchain combined with off-chain storage solutions caters to this need, enabling efficient management of large datasets without compromising the blockchain's performance [20].
- Flexibility in storage architectures—cloud storage offers advantages such as lower cost, metered service, scalable, and ubiquitous access, but raises concerns about data integrity and privacy [21]. Our framework offers a variety of off-chain storage options, allowing businesses to decide on their storage architectures flexibly based on their needs and security considerations.
- Efficient data location and retrieval—the integration of distributed hash tables (DHTs) ensures efficient and secure access to data across distributed environments, addressing the challenges of data storage and retrieval in a diverse storage landscape [22].
- Privacy preservation—utilizing encryption and privacy-enhancing technologies like data masking ensures that sensitive data remain confidential, catering to the increasing privacy concerns in data sharing [23].
- Efficient access rights control—the framework's capability to manage access rights efficiently without requiring extensive re-encryption processes enhances its practicality and efficiency, particularly in dynamic business environments.

- Compliance and auditability—the blockchain component of the framework provides a transparent and auditable record of all data sharing requests and access events, enhancing accountability and regulatory compliance [24].

4. Preliminaries

4.1. Data Encryption Technology

4.1.1. Additive Homomorphic Encryption

Homomorphic encryption is a cryptographic technique that enables computations on encrypted data without decryption. It allows computation on encrypted data, yielding the same results as unencrypted data, and is faster and consumes less memory space compared to traditional methods [25]. In other words, it will enable operations to be performed on data while still in encrypted form. In 1999, French cryptographer Paillier [26] published the Paillier algorithm at Eurocrypt, one of the most prestigious academic conferences in cryptography. This became the initial source of additive homomorphic encryption algorithms. The algorithm implemented in this system is rooted in the cryptographic technology that harnesses computational complexity theory to solve mathematical problems. It is the only additive homomorphic encryption algorithm designated by the ISO homomorphic encryption international standard [27]. It offers a range of desirable properties, including the ability to perform secure computations on encrypted data while maintaining the privacy and integrity of the data. It can satisfy the following properties:

$$E(x) \oplus E(y) = E(x + y) \quad (1)$$

- Randomly select two large prime numbers of equal length p, q .
 n and λ are defined by $n = pq$, $\lambda = \text{lcm}(p-1, q-1)$, where $\text{lcm}(x, y)$ denotes the least common multiple of x and y .
- Randomly select integer $g \in Z_{n^2}^*$, $L(x) = \frac{x-1}{n}$ is defined, calculate $\mu = (L(g^\lambda \bmod n^2))^{-1} \bmod n$, and generate keys as

$$\begin{cases} \text{public-key} = (n, g) \\ \text{private-key} = (\lambda, \mu) \end{cases} \quad (2)$$

- To perform encryption, input plaintext information m , $0 \leq m \leq n$, select random numbers $r \in Z_n^*$, $\text{gcd}(r, n) = 1$, and compute the encrypted ciphertext:

$$c = g^m r^n \bmod n^2 \quad (3)$$

- To perform decryption, compute the plaintext message:

$$m = L(c^\lambda \bmod n^2) \cdot \mu \bmod n \quad (4)$$

This is particularly useful for protecting data privacy in scenarios where data need to be handled in a secure and confidential manner (e.g., in cloud computing or secure data analytics) without exposing the plaintext.

Homomorphic encryption provides a significant advantage over traditional encryption methods. Unlike traditional encryption, which requires decryption before further operations, homomorphic encryption allows computations to be directly executed on encrypted data, eliminating the need for repeated encryption decryption [28]. It not only strengthens data security by reducing exposure to potential breaches during computation but also facilitates secure and privacy-preserving data processing in distributed environments, such as cloud computing and outsourced data analysis. Additionally, homomorphic encryption enables secure collaboration on sensitive data across different parties without compromising confidentiality, making it a powerful tool for advancing privacy-preserving technologies in various domains, including healthcare, finance, and enterprise data sharing [29].

4.1.2. Proxy Re-Encryption

Proxy re-encryption (PRE) is a cryptographic technique allowing an entity (called a proxy) to re-encrypt encrypted data from one key to another without accessing plaintext information. Proxy re-encryption was first introduced by Matt Blaze et al. [30] at the 1998 Eurocrypt conference. Their work laid the groundwork for the development of encryption schemes that allow a proxy to transform ciphertexts from one key to another, without learning anything about the underlying plaintext.

This technique has applications in various ways, especially in protecting data privacy and enabling secure data sharing and cloud storage. Such a process allows the data owner to delegate to an agent and have the agent pass the data between them while keeping the data encrypted. The critical steps in proxy re-encryption are as follows:

- Key generation: Each entity has its own public key(pk) and private key(sk). The proxy that performs the re-encryption does not need to decrypt the data but can directly transform ciphertexts from being decryptable. For example, the process whereby entity A generates a re-encryption key using the public key of itself and the private key of B can be represented as follows:

$$rk_{A \rightarrow B} = f(sk_A, pk_B) \quad (5)$$

- Encryption: the data owner encrypts the data m using the agent's public key and passes the ciphertext to the agent.

$$c_A = Enc_{pk_A}(m) \quad (6)$$

- Proxy re-encryption: The proxy re-encrypts the data to another key using its private key, without knowing the plaintext. In this way, the proxy obtains a new ciphertext and keeps the data encrypted.

$$c_B = ReEnc_{rk_{A \rightarrow B}}(c_A) \quad (7)$$

- Decryption: The recipient of the data decrypts the agent's re-encrypted ciphertext using its private key to obtain the final plaintext. In the previous example, B can use its private key to decrypt

$$c_B : m = Dec_{sk_B}(c_B) \quad (8)$$

The design and application of proxy re-encryption relies on specific scenarios and requirements to ensure privacy protection and security in data sharing and delegation contexts.

4.1.3. Data Perturbation

Data perturbation is a privacy-preserving technique that aims to protect the privacy of sensitive information while maintaining the validity and usefulness of the data by transforming or perturbing the original data with a certain degree of modification.

The concept of data perturbation dates back to methods developed in statistical disclosure control (SDC), which were introduced to protect privacy in statistical databases. Early forms of data perturbation were simple and aimed at adding "noise" to data to prevent the identification of individuals from published datasets. Dalenius introduced various methods to assess and mitigate the risk of disclosing information in statistical databases [31]. After that, more sophisticated methods were developed. The methods include data swapping and adding random noise aimed at protecting individual data while allowing for statistical analysis [32].

This technique is now usually applied in scenarios where sensitive data need to be shared or processed to prevent unauthorized access or disclosure. Commonly used data masking techniques include methods such as desensitization, where a portion of the information in sensitive data is replaced with fuzzy or fictitious values.

Dwork [33] introduced the concept of differential privacy, which revolutionized the field of data privacy. It often uses perturbation techniques, such as adding Laplacian and Gaussian noise to the data, offering strong guarantees against various types of inference attacks. Some other methods include noise addition, generalization, erasure, synthetic data generation, etc.

4.2. Data Storage Technology

4.2.1. Blockchain

Blockchain is a distributed and immutable digital ledger technology. The initial research on this technology came from Nakamoto's [34] study on Bitcoin transactions, A Peer-to-Peer Electronic Cash System, which is also the first paper to introduce Bitcoin. It is often associated with cryptocurrencies such as Bitcoin, but its applications go far beyond that. Essentially, blockchain is a decentralized database that records transactions on a computer network securely and tamper-proof. When data users cannot fully trust centralized institutions to store and process data, blockchain has more potential to guarantee data access, control, and security. Over time, it has been realized that this technology is applicable to digital currencies and can have far-reaching implications in many other areas. The technology can be applied in various areas, including supply chain management, voting systems, and protection of digital assets to ensure consistency in transmission protocols, data ownership, and other elements.

Blockchain is usually classified as a public blockchain, private blockchain, or permitted blockchain, depending on the participants:

- Public blockchain: Bitcoin and Ethereum are two famous examples of a public blockchain whose data are completely open and can be accessed and queried by anyone.
- Private blockchain: private blockchains are typically used within an organization as they restrict access to the blockchain, usually to authorized users or entities.
- Permissioned blockchain: In comparison, the permissioned blockchain is somewhere in between, managed by a group of organizations or entities for more flexible, secure, and customizable applications. It is suitable for scenarios where multiple companies collaborate. This hybrid approach enables interaction with other entities while meeting privacy and authorization needs. Permissioned blockchains can use more streamlined consensus mechanisms such as Practical Byzantine Fault Tolerance (PBFT), Raft, or Hybrid Consensus.

4.2.2. Distributed Hash Table

A distributed hash table (DHT), also known as the Kademlia algorithm, is a type of distributed computing system. The concept of distributed hash tables was introduced by four seminal papers that were published in quick succession in the early 2000s, which proposed the architectures of Chord, CAN, Pastry, and Tapestry [35–38]. Each of these works contributed to the foundational understanding and development of DHTs used in decentralized applications today. For example, a recent study shows that it can be combined with encryption technology to provide secure data storage and retrieval and be used to manage decentralized identities and access control [39].

A DHT aims to enable data to be stored and retrieved efficiently. In terms of data sharing, using a DHT to build distributed data storage systems can ensure that data are evenly distributed among participants, improving data availability and fault tolerance. A DHT is based on the concept of hash function, which distributes data to multiple nodes and locates and retrieves the stored data through hash values. Each node is responsible for maintaining a portion of the data and works together through network protocols to present the entire system with a consistent, distributed hash table structure. The basic theory of a DHT includes the following key concepts:

Hash functions and hash rings: a hash function is a mathematical function that maps an input to a fixed-length output. For a DHT, a hash function is typically used to map the

identity of the data to a unique hash value, which is then mapped to a ring space, called a hash ring. The hash value determines which node in the DHT the data should be stored on, and each node occupies a position in the hash ring. The mathematical representation of the hash function is

$$H(x) : x \rightarrow [0, 2^n - 1] \quad (9)$$

where n is the number of bits in the hash output and the SHA-1 hash function is used in the Kademlia DHT.

Data storage and retrieval: To store or retrieve data in the DHT, the hash value of the data identifier is first calculated by the hash function. Then, the nearest node to this hash value is found on the hash ring. The data are stored on this node or retrieved from this node. This approach makes the data evenly distributed among the nodes. The mathematical representation can be to find the node that satisfies the following conditions, where k is the identifier of the node:

$$Node = \min(|H(k) - H(id)|) \quad (10)$$

Node communication protocol: some kind of communication protocol is required between the nodes so that they can work together to store and retrieve data. A basic message format is $Message = \{Type, SourceID, TargetID, Payload\}$.

Overall, a DHT also considers features such as distributed consistency and fault tolerance, which helps to achieve the efficient storage and retrieval of data in a distributed environment.

5. Framework Overview

In this work, we propose a three-layer blockchain-based framework for B2B data sharing. As illustrated in Figure 1, the structure comprises a data storage layer for secure data handling, a privacy preservation layer ensuring data privacy through advanced encryption, and an access control layer for dynamic permission management, offering a comprehensive solution for secure and efficient business data exchange.

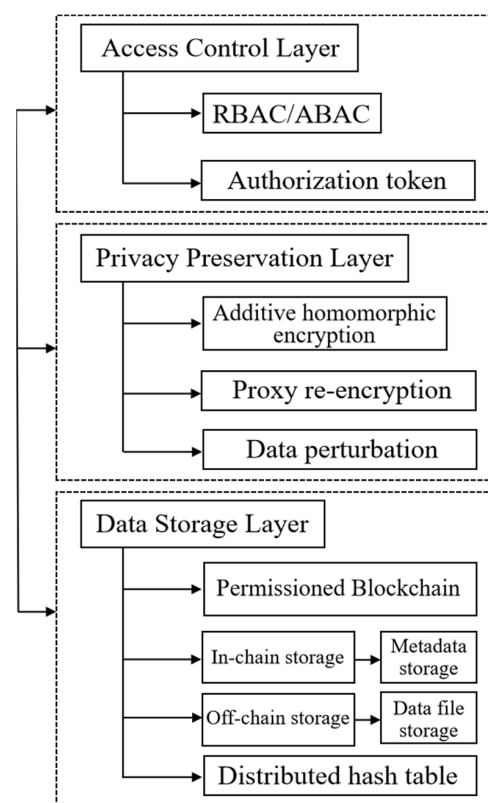


Figure 1. Three-layer B2B data sharing framework. RBAC and ABAC in the figure are abbreviations for role-based access control and attribute-based access control.

- Access Control Layer

In the access control layer, the data owner has the flexibility to control and adjust user access to data, including options to revoke or reassign access rights. This layer employs different access control mechanisms for local and cloud storage. For local and cloud storage, we employ a traditional method such as role-based access control (RBAC). For cloud storage, we employ a complex approach for access control, utilizing an authorization token for each data user.

- Privacy Preservation Layer

The privacy preservation layer incorporates integrating additive homomorphic encryption, data perturbation, and proxy re-encryption mechanisms. The workflow of the privacy preservation layer primarily involves two critical stages: key generation and data encryption.

In the key generation stage, the data owner creates unique public and private key pairs for themselves as well as for each authorized user using asymmetric encryption algorithms, and securely transmits the secret keys to the corresponding users. Furthermore, a proxy re-encryption key will be created for each authorized user, which will be used later.

In the data encryption stage, we employ different encryption approaches for local and cloud storage. For local storage, the privacy layer emphasizes secure data storage on the device and secure transmission within the local network, employing traditional encryption methods that combine symmetric encryption with asymmetric encryption key exchange. In the context of cloud storage, the focus shifts more towards the security and management of data within the cloud environment, considering the unique characteristics and challenges posed by cloud services. Our solution ensures that data remain secure and properly managed, regardless of whether they are stored locally or in the cloud.

- Data Storage Layer

In our B2B data sharing platform design, we adopted a hybrid approach to data storage, combining a permissioned blockchain with off-chain databases. To ensure data integrity and transparency, the permissioned blockchain stores metadata, while business data are offloaded to off-chain storage systems. We use distributed hash tables (DHTs) for content addressing, facilitating the efficient location and retrieval of files across diverse storage formats.

6. Data Sharing Process

In this section, we will present the complete data sharing process under the framework proposed in Section 5. The complete process is as follows (shown as Figure 2):



Figure 2. Data sharing process.

Step 1: Data Storage:

Data are strategically stored based on their nature and sensitivity. Essential information such as metadata are stored on-chain for transparency and auditability. Bulk business data, which are typically voluminous, are stored off-chain in various storage systems like databases and cloud storage solutions for privacy and efficiency. More details will be provided in Section 7.

Step 2: Data Encryption:

Before the data are uploaded, they are encrypted using additive homomorphic encryption. This allows certain calculations to be performed on the encrypted data without needing to decrypt them. To further enhance privacy, data perturbation is applied to the

datasets intended for sharing. This modifies the data slightly to mask the actual properties and values of the original data while maintaining overall statistical characteristics.

Step 3: Access Control:

The data owner can have fine-grained control over the sharing of data, which includes granting access, revoking it, and reassigning it. In the proposed framework, the data owner can revoke access for specific users and assign different access rights, without the need for key redistribution or data re-encryption.

Step 4: Data Access:

When a data request is made, the distributed hash table (DHT) is utilized to locate the requested file across the diverse storage landscape. The system then verifies the requester's authorization to access the data. If authorized, proxy re-encryption is employed to transform the data, which are encrypted for the owner's key, into a format that the requester's key can decrypt, facilitating secure data sharing without exposing the actual data.

Step 5: Transaction Audit:

Every action on the platform, including data sharing requests and access events, is logged on the blockchain for auditability.

7. Analysis and Discussion

In this chapter, we will provide a comprehensive explanation of the three-layer B2B data sharing framework that was proposed in Section 6. We will critically analyze each of its components and mechanisms to demonstrate its effectiveness. The following sections will break down these layers, present a detailed analysis of their functionalities and interdependencies, and illustrate how, collectively, they form a cohesive and potent solution to the complex challenges of secure and efficient data sharing in business-to-business contexts.

7.1. Data Storage Layer

In the data storage layer, we use a hybrid approach of a permissioned blockchain with off-chain databases, where metadata are stored on blockchain for transparency, and business data are stored off-chain for efficiency. A DHT is used for efficient file location and retrieval. In this section, we will discuss how data are strategically stored based on their nature and sensitivity and explain the significance of a hybrid storage approach in terms of transparency, auditability, privacy, and efficiency.

- **Permissioned Blockchain**

Blockchain technology offers two distinct types: open, permission-less systems like Bitcoin or Ethereum; and controlled, permissioned networks such as the Hyperledger Project from The Linux Foundation. In a permission-less or public blockchain, the identities of the participants in the system remain anonymous. It implies that anyone can join or leave the blockchain network at any time, increasing the risk of security breaches in the network. This is quite dangerous for B2B data sharing. Nevertheless, in a permissioned or private blockchain, only a known and identifiable set of participants are explicitly admitted to the blockchain network. It reduces the presence of malicious actors within the network. Consequently, only authenticated and authorized actors can participate in the network, which enhances the security of the system, as required by enterprise applications. This is important to the accessibility of B2B data sharing. Based on the above arguments, we choose to base our framework on a permissioned blockchain for controlled access, enhanced security, and regulatory compliance.

- **In-chain Storage**

The main purpose of in-chain storage is to record all critical data operations, including data creation, modification, and access events, as well as the hash value of the data themselves. It takes advantage of the fundamental feature of blockchain, which is that data cannot be changed once they are recorded, ensuring the credibility and consistency of the information.

When new data are created and uploaded to the platform, their metadata (e.g., file size, format, and time of creation) and the hash value of the data are recorded on the

blockchain. The hash value is generated by applying a cryptographic hash function to the data content, which provides a unique digital fingerprint of the data. This hash value plays a key role in the data lifecycle and is used to verify that the data have not been tampered with. If the data are modified, the specific details of the modification (including the modification time and the identity of the modifier) and the hash value after modification will be recorded. In this way, any change to the data will leave an immutable trace, guaranteeing the historical traceability of the data. This is important to the data integrity of B2B data sharing. Whenever data are accessed, the time of access, the identity of the visitor, and the nature of the access (e.g., viewed or downloaded) are recorded. This provides transparency of data usage and facilitates monitoring and auditing. A standard metadata will have the following structure (Figure 3):

```
Record{
  TransactionID: Unique identifier of the transaction,
  OperationType: Type of operation, 'Create', 'Modify', or 'Access'
  Timestamp: Time when the operation was performed,
  Metadata: {
    Identifier: Unique identifier of the dataset,
    Size: Size of the dataset,
    Format: Format of the dataset (e.g., CSV, JSON),
    Owner: Identity of the entity uploading the dataset
  }
  DataHash: Cryptographic hash of the dataset,
  Operator: Identity of the entity uploading the dataset,
  OperationDetails: Specific details of the operation (e.g., 'view', 'download')
}
```

Figure 3. Standard metadata structure.

- Off-chain Storage

In our B2B data sharing framework, off-chain storage is a key part of managing large and complex business data, especially when it comes to traditional databases and cloud storage solutions. These storage options are suitable for real-world business data, such as large datasets and bulky files, which are often not suitable for storage on the blockchain due to scalability limitations. Thus, off-chain storage provides the necessary capacity and flexibility to efficiently handle large datasets while addressing the limitations of the blockchain in handling high volume data storage. The combination of using in-chain and off-chain storage is to balance the need of robustness deployed by in-chain storage while balancing the scalability concern of voluminous B2B data deployed by off-chain storage.

Cloud storage is a popular choice for many businesses facing growing data volumes and large storage requirements. It reduces the cost of database construction and maintenance and facilitates data exchange and circulation. However, for companies highly sensitive about data privacy, storing substantial data with centralized cloud service providers (CSPs) can pose significant risks. Storing large amounts of data with semi-trusted third parties could lead to misuse and disclosure of proprietary information.

Therefore, for real-time or sensitive data, businesses often prefer local storage on their own devices, exposing access interfaces to the sharing systems. This approach provides companies with more direct control over their data, reducing the risks associated with third-party data breaches. Local storage ensures the immediacy and sensitivity of the data while allowing for safe sharing and access when needed. In addition, off-chain storage systems offer flexibility to comply with various regulatory requirements, such as GDPR, especially when sharing data that could potentially identify individuals. They can be

customized to align with diverse compliance standards, an essential aspect for businesses operating across different legal landscapes.

Our framework offers a variety of off-chain storage choices, allowing businesses to flexibly decide on their data storage method based on their needs and security considerations. Whether choosing cloud storage for its cost-effectiveness and convenience or opting for local storage for greater security and control, our framework supports effective and secure data sharing while ensuring data safety and privacy.

- Distributed hash tables

In the data storage layer, the DHT serves to locate datasets across different business entities. To illustrate how the DHT functions in the framework, consider an example of supply chain management. In such a scenario, the implementation of a DHT facilitates a decentralized approach to sharing data among various entities such as manufacturers, suppliers, distributors, and retailers. Each of these businesses operates as a node within the DHT network, representing their own unique databases. These nodes could be servers or data centers belonging to different organizations participating in the data sharing process. In addition to storing actual data, these nodes keep the DHT to hold and share information about where and how specific data can be accessed.

For each business's database, a unique key is generated using a hash function. This key is typically derived from identifiable attributes of the business, i.e., identifier for the dataset mentioned above. When a business decides to share specific data, like inventory levels or production schedules, it does not directly store these data in the DHT. Instead, it registers an entry in the DHT under the generated key, with the entry containing metadata or pointers indicating how and where these data can be accessed, such as an API endpoint or a database query interface. A DHT entry may look like Figure 4:

Key:ef6523a34d8e4a018b14a7abf1c79f59a8c4ef15'
Value:
- Database Access Point: 'http://db.tech6523.com'
- API Endpoint: 'https://api.tech6523.com/inventory'
- Metadata: 'Inventory data, last updated on 2023-11-18'

Figure 4. A DHT entry sample.

When a user wants to find a database, they use the hash function on the dataset's identifier to generate the key and query the DHT. The DHT responds by providing the information about where the database is located.

It should be noticed that every node in the distributed network has a part of the DHT, rather than the entire table duplicated in all nodes. When there is a request for the corresponding value for the key, a peer receives the request and checks for the key in its own table. If it is available, the value will be returned or else the request will be passed on to the peers until the value is found. This is important in B2B data sharing because the use of a DHT can prevent the problem of putting all eggs in one basket.

7.2. Privacy Preservation Layer

The privacy preservation layer incorporates integrating additive homomorphic encryption, data perturbation, and proxy re-encryption mechanisms. The workflow of the privacy preservation layer primarily involves two critical stages: key generation and data encryption.

7.2.1. Key Generation

Based on asymmetric encryption algorithms, the data owner generates unique public and private key pairs for each authorized user. The first one is the master key pair for

the data owner (pk_a, sk_a) , where pk_a is kept secret. The second key public/private pair (pk_b, sk_b) is created for each authorized user. Then, the data owner securely sends the corresponding key pair to the authorized user, which can be realized by a temporary key exchange protocol or existing public/private key pair of both. Both the data owner's public key pk_a and the authorized user's public key pk_b are considered public. Furthermore, for each authorized user, the data owner creates a proxy re-encryption key $rk_{a \rightarrow b}$, which is used to secure data sharing between different users without decryption and re-encryption. The key is generated using sk_a and pk_b , that is

$$rk_{a \rightarrow b} = \text{Generate Re EncryptionKey}(sk_a, pk_b) \quad (11)$$

7.2.2. Data Encryption

There are different encryption approaches for local storage and cloud storage based on their different storage environments.

- Local storage:

For local storage, the privacy layer focuses on the secure storage of data on the device and the secure transmission within the local network. Compared to cloud storage, local storage is less exposed to external threats such as data breaches happening in cloud service providers. Therefore, we employ traditional encryption methods combining symmetric encryption with the asymmetric encryption key exchange method. This mechanism is well established and therefore will not be further discussed here.

- Cloud storage:

For cloud storage, the privacy layer focuses more on the security and management of data in the cloud environment, while considering the characteristics and challenges of cloud services.

If you consider the cloud service provider as a semi-trusted third party, there are potential security issues. One of the main security concerns in cloud computing is how the data are being used by a third-party cloud service provider. Combining access control with re-encryption is a viable solution for protecting data from unauthorized access and cloud breaches [40]. In this framework, we adopt homomorphic encryption and data perturbation to protect sensitive data stored on the cloud from unauthorized access.

Before the data are uploaded to the cloud platform for storage, the data owner first implements data masking on each data record, replacing, disrupting, or partially hiding sensitive information to ensure that the original content of the data has been protected when they leave the local environment. Then, they are further encrypted using homomorphic encryption. For each authorized user, corresponding authorization tokens are generated. These tokens determine the user's level of access to the encrypted data. More details about the authorization process will be provided at the access control layer. Algorithm 1 illustrates the encryption process, which involves data masking and encryption:

Algorithm 1 Data Masking and Encryption

```

1: procedure MASKANDENCRYPT ( $d, pk_a$ )
2:    $n \leftarrow$  number of attributes in  $d$ 
3:    $m \leftarrow$  selectRandomNonNegativeNumber()
4:    $r \leftarrow$  generateRandomIntegerList( $n + m$ )
5:   for  $i \leftarrow 1$  to  $n + m$  do
6:      $d'_i \leftarrow d_i + r_i$ 
7:      $E_{pk_a}(d'_i) \leftarrow \text{EncryptAlgorithm}(d'_i, pk_a)$ 
8:   end for
9:   return  $\{E_{pk_a}(d'_1), E_{pk_a}(d'_2), \dots, E_{pk_a}(d'_{n+m})\}$ 
10: end procedure

```

Assume that the data owner's data consists of multiple records, and each data record d contains n attributes. Each record can be represented as $\{d_1, d_2, \dots, d_n\}$, where d_i is the value of the i^{th} attribute. In the process, we need to mask the attribute first. To protect sensitive data, the owner can mask the attributes and their values by selecting n random integers, denoted $\{r_1, r_2, \dots, r_{n+m}\}$. These integers are chosen from a specific number field. The value of m is an additional security parameter to mask the actual number of attributes of the data record, which is also randomly selected and may vary from record to record based on the security requirements of the data owner for the given application. For each data record d_i , data owner combines it with the correspond random number r_i , and then encrypted it using the public key pk_a , generating $E_{pk_a}(d_i + r_i)$.

The core of this process is to increase the privacy of the data through random numbers, so that the privacy of the data content is still protected even if the data are stored on an external cloud service. Through this mechanism, even if someone accesses these data, they will not be able to determine the original values of the attributes. Additionally, they will not be able to determine the total number of attributes that existed in the original data record. This enhances the protection of not only the raw attribute values but also the data structure information. This is important to the privacy aspect of B2B data sharing.

7.3. Access Control Layer

In the access control layer, the data owner has the flexibility to control and adjust user access to data. This includes the option to revoke a user's access rights or to reassign them in the future. Additionally, the use of data masks ensures that even if a user's access is revoked, the data they previously accessed cannot reveal any confidential information.

It is similar to the privacy preservation layer, as different access control mechanisms are employed for local storage and cloud storage. It is under direct physical control and therefore we can use traditional access control mechanisms such as role-based access control (RBAC). For finer-grained control, attribute-based access control (ABAC) can be used, where user attributes and environmental factors can be considered.

However, the situation becomes complicated for cloud storage because of privacy, security, and regulation issues. Data must be encrypted before they are uploaded to the cloud service provider, which is generally a secure practice, but it may introduce certain challenges such as a performance overhead and access control complexity. This is quite normal in B2B data sharing. Before the authorized user obtains the data, the data owner has to download and decrypt the encrypted data from the CSP first and re-encrypt the data using the data user's private key and finally upload to the CSP again. It can introduce a performance overhead, especially in decryption and encryption for large datasets. Moreover, managing who can access which pieces of data can become complex, especially in a dynamic B2B environment where access needs may frequently change. This requires a robust access control mechanism that integrates well with the encryption system.

To address these challenges, we adopt a proxy re-encryption mechanism. We generate authorization tokens for the encrypted data $E_{pk_a}(d + r)$ to manage the access control. Consider four stages: access grant, data access, access revocation, and access reassign.

7.3.1. Access Grant

If data owner wants to grant access to the data user for a set of attributes S (where S is a subset of d), an authorization token T_b^d is generated for the user corresponding to the attributes, that is

$$T_b^d = \left\{ \text{DataUser}, rk_{a \rightarrow b}, \left\{ E_{pk_b}(\alpha_1), \dots, E_{pk_b}(\alpha_{n+m}) \right\} \right\} \quad (12)$$

where

$$\alpha_i = \begin{cases} -r_i & \text{if } 1 \leq i \leq n \wedge d_i \in S \\ -d'_i & \text{otherwise} \end{cases} \quad (13)$$

$rk_{a \rightarrow b}$ is the proxy re-encryption key created in the privacy preservation layer. d'_i is calculated in Algorithm 1. For $1 \leq i \leq n$, $d'_i = d_i + r_i$; for $n < i \leq n + m$, $d'_i = r_i$.

If the data owner does not want the data user to access d , which means $S = \emptyset$, she does not generate any authorization token, effectively setting the data user's token $T_b^d \leftarrow null$.

The data owner creates a list of authorization tokens that specifies which users can access which parts of the data. This list, denoted as T^d , includes tokens for accessing either the entire record or specific parts of it, depending on the permissions granted. If a user's token for a particular data record is null, indicating no access, it is not included in the list. Finally, the data owner uploads the authorization tokens T^d and the corresponding encrypted data $E_{pk_a}(d')$ records to the cloud.

7.3.2. Data Access

When the data user wants to access the data, he/she will send a data request to the cloud. When receiving a data request from the data user, the cloud first checks if there is an entry for the data user in an authorization token list T^d . If no entry is found, the process is aborted. If the data user is found to be authorized, the cloud proceeds with a multi-step re-encryption and decryption procedure. This process utilizes the proxy re-encryption key $rk_{a \rightarrow b}$ to convert $E_{pk_a}(d')$. The cloud then performs an additive homomorphic calculation on the encrypted data as follows:

$$E_{pk_b}(d'_i + \alpha_i) \leftarrow E_{pk_b}(d'_i) +_h E_{pk_b}(\alpha_i), \quad 1 \leq i \leq n + m \quad (14)$$

Noted that $E_{pk_b}(\alpha_i)$ is from the user's authorization token T_b^d , and $+_h$ is the additive homomorphic calculation. The result will be sent to data user, and then he/she can perform $D_{pk_b}(E_{pk_b}(d'_i + \alpha_i))$ to get $d'_i + \alpha_i$ with private key sk_b , which is received securely from data owner in Privacy Preservation Layer. Algorithm 2 shows the data access process using homomorphic encryption, which is performed by the cloud.

Algorithm 2 Data Access with Homomorphic Encryption

```

1: procedure DATAACCESS  $(E_{pk_a}(d'), rk_{a \rightarrow b}, T_d^b)$ 
2:   Check if an authorization token  $T_d^b$  exists for the data user
3:   if No entry for data user in  $T_d$  then
4:     return Access Denied
5:   else
6:     for  $i \leftarrow 1$  to  $n + m$  do
7:        $E_{pk_b}(d'_i) \leftarrow PRE.ReEnc(E_{pk_a}, rk_{a \rightarrow b})$  where  $\alpha_i$  is from  $T_d^b$ 
8:        $E_{pk_b}(d'_i + \alpha_i) \leftarrow E_{pk_b}(d'_i) +_h E_{pk_b}(\alpha_i)$ 
9:     end for
10:    Send the result to data user
11:  end if
12: end procedure

```

The data user's decryption will yield the correct data only for the attributes he is authorized to access. The calculation of $d'_i + \alpha_i$ obtains d_i only if the data user has access to the i^{th} attribute of d . Unauthorized attributes will result in a value of 0 upon decryption. The additive homomorphic property ensures that the user can compute sums of the encrypted values without access to the private keys that encrypted them.

7.3.3. Access Revocation

The data owner can revoke a user's access to a data record d by removing the authorization token T_b^d corresponding to $E_{pk_b}(d')$. Considering that the cloud as a semi-trusted third party may not follow the rules of the protocol, we use a smart contracts mechanism

to enforce the execution of the token revocation. Such access revocation is to ensure the proper rights for accessing the assigned data in B2B data sharing.

7.3.4. Access Reassign

When the data owner decides to reassign a user's access to a data record d^1 , after a previous access revocation, he/she generates a new authorization token that specifies the set of data attributes the data user is allowed to access. This set can either be the same as the user had access to before the access rights were revoked or a different set of attributes within the data record. The data owner then sends this token to the cloud, directing it to update the authorization list T^d associated with the data record d^1 . It is similar to access revocation, with the aim of restoring proper data access rights in B2B data sharing.

8. Conclusions and Future Work

Data sharing between businesses can create value for all relevant parties. Each business needs to be clear when considering the framework for the selected data sharing, which is that although sharing will bring challenges to all parties, such as data privacy and access security issues, the expected benefits from sharing need to exceed the construction costs and risks of the sharing mechanism. This study provides a comprehensive solution to the privacy and security challenges of data exchange between businesses in the digital age by establishing a reliable three-layer B2B data sharing framework based on blockchain. Firstly, an in-depth analysis of the development trend of B2B data sharing is conducted, during which the basic structure and additional security and privacy requirements of data sharing are identified, emphasizing the urgency of privacy protection technologies and enhanced security protocols. On this basis, a framework including a data storage layer, privacy protection layer, and access control layer is designed to ensure secure and efficient business data exchange. The framework uses advanced encryption technology, ensures data privacy through compliance, and achieves fine control of data through dynamic permission management. The application of this innovative technology is expected to provide additional value to businesses in addition to ensuring the data sharing process.

Through in-depth research, it is believed that this framework can balance the confidentiality of data while achieving data sharing, providing a viable solution for businesses. Emerging technologies, such as homomorphic encryption and blockchain, their applications, and advantages in the field of B2B data sharing, are also explored. This paper deepens the understanding of privacy and security mechanisms in B2B data sharing, providing valuable insights for businesses participating in the digitalization process.

Overall, the blockchain-based B2B data sharing framework is expected to become an important support tool in the digital transformation of enterprises in the future, promoting more secure and efficient information circulation between industries. To improve and expand this data sharing framework, further research and practical implementation will be encouraged using various emerging technologies. In future, we plan to test the framework with real-time data to evaluate its performance based on the processing time and computation cost. We will conduct a comparative study using different encryption algorithms under different storage architectures and analyze the results as an extension of the existing work.

Author Contributions: Conceptualization, W.L., W.K.T. and J.C.; Methodology, W.L. and J.C.; Software, W.L. and J.C.; Validation, W.L. and J.C.; Formal analysis, W.L. and J.C.; Investigation, W.L. and J.C.; Resources, W.L. and J.C.; Data curation, W.L. and J.C.; Writing—original draft, W.L. and J.C.; Writing—review & editing, W.L., W.K.T. and J.C.; Visualization, W.L.; Supervision, W.K.T.; Project administration, W.K.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

- Goasduff, L. *Data Sharing is a Business Necessity to Accelerate Digital Business*; The Gartner Group: Stamford, CT, USA, 2020; Volume 11.
- Lee, W.S.; John, A.; Hsu, H.C.; Hsiung, P.A. SPChain: A Smart and Private Blockchain-Enabled Framework for Combining GDPR-Compliant Digital Assets Management With AI Models. *IEEE Access* **2022**, *10*, 130424–130443. [\[CrossRef\]](#)
- Shen, M.; Zhu, L.; Xu, K. Blockchain and Data Sharing. In *Blockchain: Empowering Secure Data Sharing*; Shen, M., Zhu, L., Xu, K., Eds.; Springer: Singapore, 2020; pp. 15–27.
- Chi, J.; Li, Y.; Huang, J.; Liu, J.; Jin, Y.; Chen, C.; Qiu, T. A secure and efficient data sharing scheme based on blockchain in industrial Internet of Things. *J. Netw. Comput. Appl.* **2020**, *167*, 102710. [\[CrossRef\]](#)
- Xuan, S.; Zheng, L.; Chung, I.; Wang, W.; Man, D.; Du, X.; Yang, W.; Guizani, M. An incentive mechanism for data sharing based on blockchain with smart contracts. *Comput. Electr. Eng.* **2020**, *83*, 106587. [\[CrossRef\]](#)
- Al-Zahrani, F.A. Subscription-Based Data-Sharing Model Using Blockchain and Data as a Service. *IEEE Access* **2020**, *8*, 115966–115981. [\[CrossRef\]](#)
- Wei, Q.; Shen, Z. Improving Blockchain Scalability from Storage Perspective. In Proceedings of the ACM Turing Award Celebration Conference—China 2023, Wuhan, China, 28–30 July 2023; pp. 112–113.
- Xu, C.; Zhang, C.; Xu, J.; Pei, J. SlimChain: Scaling blockchain transactions through off-chain storage and parallel processing. *Proc. VLDB Endow.* **2021**, *14*, 2314–2326. [\[CrossRef\]](#)
- Wang, K.; Yan, Y.; Guo, S.; Wei, X.; Shao, S. On-Chain and Off-Chain Collaborative Management System Based on Consortium Blockchain. In Proceedings of the Advances in Artificial Intelligence and Security, Cham, Switzerland, 19–23 July 2021; pp. 172–187.
- Mansouri, Y.; Toosi, A.N.; Buyya, R. Data Storage Management in Cloud Environments: Taxonomy, Survey, and Future Directions. *ACM Comput. Surv.* **2017**, *50*, 91. [\[CrossRef\]](#)
- Popovic, K.; Hocenski, Z. *Cloud Computing Security Issues and Challenges*; IEEE: Piscataway, NJ, USA, 2010; pp. 344–349.
- Ren, K.; Wang, C.; Wang, Q. Security Challenges for the Public Cloud. *IEEE Internet Comput.* **2012**, *16*, 69–73. [\[CrossRef\]](#)
- Kotha, S.K.; Rani, M.S.; Subedi, B.; Chunduru, A.; Karrothu, A.; Neupane, B.; Sathishkumar, V.E. A Comprehensive Review on Secure Data Sharing in Cloud Environment. *Wirel. Pers. Commun.* **2022**, *127*, 2161–2188. [\[CrossRef\]](#)
- Song, C.; Park, Y.; Gao, J.; Nanduri, S.K.; Zegers, W. Favored Encryption Techniques for Cloud Storage. In Proceedings of the 2015 IEEE First International Conference on Big Data Computing Service and Applications, Redwood City, CA, USA, 30 March–2 April 2015; pp. 267–274.
- Gai, K.; She, Y.; Zhu, L.; Choo, K.-K.R.; Wan, Z. A Blockchain-Based Access Control Scheme for Zero Trust Cross—Organizational Data Sharing. *ACM Trans. Internet Technol.* **2023**, *23*, 38. [\[CrossRef\]](#)
- Xu, H.; He, Q.; Li, X.; Jiang, B.; Qin, K. BDSS-FA: A Blockchain-Based Data Security Sharing Platform With Fine-Grained Access Control. *IEEE Access* **2020**, *8*, 87552–87561. [\[CrossRef\]](#)
- Yang, Y.; Zhu, H.; Lu, H.; Weng, J.; Zhang, Y.; Choo, K.-K.R. Cloud based data sharing with fine-grained proxy re-encryption. *Pervasive Mob. Comput.* **2016**, *28*, 122–134. [\[CrossRef\]](#)
- Qin, Z.; Xiong, H.; Wu, S.; Batamuliza, J. A Survey of Proxy Re-Encryption for Secure Data Sharing in Cloud Computing. *IEEE Trans. Serv. Comput.* **2016**, *1*. [\[CrossRef\]](#)
- Zhu, L.; Song, S.; Peng, S.; Wang, W.; Hu, S.; Lan, W. The Blockchain and Homomorphic Encryption Data Sharing Method in Privacy-Preserving Computing. In Proceedings of the 2022 IEEE/ACIS 7th International Conference on Big Data, Cloud Computing, and Data Science (BCD), Danang, Vietnam, 4–6 August 2022; pp. 84–87.
- Li, R.; Song, T.; Mei, B.; Li, H.; Cheng, X.; Sun, L. Blockchain for Large-Scale Internet of Things Data Storage and Protection. *IEEE Trans. Serv. Comput.* **2019**, *12*, 762–771. [\[CrossRef\]](#)
- Salim, A.; Tiwari, R.K.; Tripathi, S. An Efficient Public Auditing Scheme for Cloud Storage with Secure Access Control and Resistance Against DOS Attack by Iniquitous TPA. *Wirel. Pers. Commun.* **2021**, *117*, 2929–2954. [\[CrossRef\]](#)
- Li, B.; Wu, H.; He, X.; Wang, B.; Xu, E. Survey of Storage Scalability in Blockchain Systems. *Comput. Sci.* **2023**, *50*, 318–333.
- Zhu, L.; Gao, F.; Shen, M.; Li, Y.; Zheng, B.; Mao, H.; Wu, Z. Survey on privacy preserving techniques for blockchain technology. *J. Comput. Res. Dev.* **2017**, *54*, 2170–2186.
- Hammoud, O.; Tarkhanov, I.A. A Novel Blockchain-Integrated Distributed Data Storage Model with Built-in Load Balancing. In Proceedings of the 2022 IEEE 16th International Conference on Application of Information and Communication Technologies (AICT), Washington, DC, USA, 12–14 October 2022; pp. 1–6.
- Asante, G.; Ben, J.; Asante, M.; Dagadu, J. A Symmetric, Probabilistic, Non-Circuit Based Fully Homomorphic Encryption Scheme. *Int. J. Comput. Netw. Appl.* **2022**, *9*, 160–168. [\[CrossRef\]](#)
- Paillier, P. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Proceedings of the Advances in Cryptology—EUROCRYPT ’99, Berlin, Heidelberg, 2–6 May 1999; pp. 223–238.
- ISO/IEC 18033-6:2019; IT Security Techniques—Encryption Algorithms—Part 6: Homomorphic Encryption. ISO: Geneva, Switzerland, 2019.

28. Martins, P.; Sousa, L.; Mariano, A. A Survey on Fully Homomorphic Encryption: An Engineering Perspective. *ACM Comput. Surv.* **2017**, *50*, 83. [CrossRef]
29. Khedr, A.; Gulak, G. SecureMed: Secure Medical Computation Using GPU-Accelerated Homomorphic Encryption Scheme. *IEEE J. Biomed. Health Inform.* **2018**, *22*, 597–606. [CrossRef]
30. Blaze, M.; Bleumer, G.; Strauss, M. Divertible protocols and atomic proxy cryptography. In Proceedings of the Advances in Cryptology—EUROCRYPT’98, Berlin, Heidelberg, 31 May–4 June 1998; pp. 127–144.
31. Dalenius, T. Towards a methodology for statistical disclosure control. *Stat. Tidskr.* **1977**, *15*, 429–444.
32. Spruill, N.L. The Confidentiality and Analytic Usefulness of Masked Business Microdata. 2002. Available online: http://www.asasrms.org/Proceedings/papers/1983_114.pdf (accessed on 3 May 2024).
33. Dwork, C. Differential Privacy. In *Encyclopedia of Cryptography and Security*; van Tilborg, H.C.A., Jajodia, S., Eds.; Springer US: Boston, MA, USA, 2011; pp. 338–340.
34. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. *Bitcoin* **2008**, *4*, 15. Available online: <https://bitcoin.org/bitcoin.pdf> (accessed on 3 May 2024).
35. Stoica, I.; Morris, R.; Karger, D.; Kaashoek, M.F.; Balakrishnan, H. Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM Comput. Commun. Rev.* **2001**, *31*, 149–160. [CrossRef]
36. Ratnasamy, S.; Francis, P.; Handley, M.; Karp, R.; Shenker, S. A scalable content-addressable network. *SIGCOMM Comput. Commun. Rev.* **2001**, *31*, 161–172. [CrossRef]
37. Rowstron, A.; Druschel, P. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In Proceedings of the Middleware 2001, Berlin, Heidelberg, 12–16 November 2001; pp. 329–350.
38. Zhao, B.Y.; Kubiawicz, J.D.; Joseph, A.D. *Tapestry: An Infrastructure for Fault-tolerant Wide-Area Location and Routing*; University of California at Berkeley: Berkeley, CA, USA, 2001.
39. Raj, T.F.M.; Vallathan, G.; Perumal, E.; Sudhakar, P.A.J. Future and Research Perspectives of Spatiotemporal Data Management Methods. In *Spatiotemporal Data Analytics and Modeling: Techniques and Applications*; A, J., Abimannan, S., El-Alfy, E.S.M., Chang, Y.S., Eds.; Springer Nature Singapore: Singapore, 2024; pp. 235–245.
40. Samanthula, B.K.; Howser, G.; Elmehdwi, Y.; Madria, S. An efficient and secure data sharing framework using homomorphic encryption in the cloud. In Proceedings of the 1st International Workshop on Cloud Intelligence, Istanbul, Turkey, 31 August 2012. [CrossRef]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.