computers

Article

A Secure and Verifiable Blockchain-Based Framework for Personal Data Validation

Junyan Yu, Ximing Li and Yubin Guo *

College of Mathematics and Informatics College of Software Engineering, South China Agricultural University, Guangzhou 510642, China; 2090593733@stu.scau.edu.cn (J.Y.); liximing@scau.edu.cn (X.L.)
* Correspondence: guoyubin@scau.edu.cn; Tel.: +86-20-13802403342

Abstract: The online services provided by the Service Provider (SP) have brought significant convenience to people's lives. Nowadays, people have grown accustomed to obtaining diverse services via the Internet. However, some SP utilize or even tamper with personal data without the awareness or authorization of the Data Provider (DP), a practice that seriously undermines the authenticity of the DP's authorization and the integrity of personal data. To address this issue, we propose a Verifiable Authorization Information Management Scheme (VAIMS). During the authorization process, the authorization information and personal data fingerprints will be uploaded to the blockchain for permanent record, and then the SP will store such authorization information and personal data. The DP generates corresponding authorization fingerprints based on the authorization information and stores them independently. Through the authorization information and authorization fingerprints on the chain, the DP can verify the authenticity of the authorization information stored by the SP at any time. Meanwhile, by leveraging the personal data fingerprint on the blockchain, the DP can check whether the personal data stored by the SP has been tampered with. Additionally, the scheme incorporates database technology to accelerate data query. We implemented a VAIMS prototype on Etherum, and experiments demonstrate that the scheme is effective.

Keywords: blockchain; fingerprint; smart contract; verifiable

check for updates

Citation: Yu, J.; Li, X.; Guo, Y. A Secure and Verifiable Blockchain-Based Framework for Personal Data Validation. *Computers* **2024**, *13*, 240. https://doi.org/10.3390/computers13090240

Academic Editors: Raman Singh and Shantanu Pal

Received: 1 August 2024
Revised: 12 September 2024
Accepted: 19 September 2024
Published: 23 September 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

With the advancement of the Internet and communication technologies, data has emerged as a crucial economic asset [1]. "Personal data" is defined as information that can be employed to identify a natural person, such as a name, home address, email address, ID number, location data, or IP address. This data is susceptible to being leaked or tampered with during the sharing, processing, and usage of data. Particularly, in the current scenario of the proliferation of Service Providers (SPs), there is a corresponding rise in the number of cases where personal data is illegally used. For example, Papageorgiou et al. [2] carried out a study assessing the data security and privacy features of 20 popular m-health apps, disclosing that many of these apps collected the personal sensitive information of Data Providers (DPs) without obtaining their consent. For instance, a portion of apps activates the microphone without the permission of DPs to record. Similarly, Trung T. N. et al. discovered that 30% of Android apps share personal information with third parties without obtaining the consent of DPs [3]. Additionally, in some applications, even if DPs opt to refuse tracking, some websites will improperly store invalid tracking consent evidence and continue to track the browser activities of DPs [4].

The General Data Protection Regulation (GDPR), enacted by the European Union (EU), offers a clear and practical legal framework as well as scenario for the protection of personal data. The GDPR defined three distinct roles in data processing: personal data owners, data controllers, and data processors. In the majority of instances, the role of data controller and data processor are the same. In this paper, "DP" will represent personal

data owners, while "SP" will include data controllers and data processors. Figure 1 depicts the current process for authorizing and utilizing personal data. Initially, the DP selects a service and submits a request to the SP. Subsequently, the SP discloses its privacy policy and seeks consent based on the requested service. The DP gives consent and the necessary personal data, which the SP then stores along with the consent. Finally, the SP provides the corresponding services. Nevertheless, within this framework, the DP's personal data and authorization information are entirely stored and controlled by the SP. The SP has complete control over both the DP's data and their authorization information. As a result, the SP has the ability to tamper with personal data or authorization information, and to 'ake fair use' of the data within a tampered authorized scope at will, all without the DP's knowledge. It is impossible to determine the truthfulness of the authorized information or the integrity of the personal data.

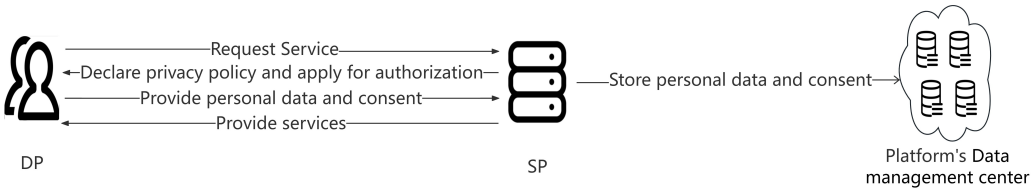


Figure 1. Authorization and use process of personal data.

This paper proposed a Verifiable Authorization Information Management Scheme (VAIMS) to tackle the abovementioned issues, with the following key contributions:

1. We leverage blockchain technology to store authorization information and data fingerprints, and employ smart contracts to implement authorization, ensuring transparency in the authorization process. This prevents the SP from forging DP authorization and guarantees the authenticity of authorization. Additionally, blockchain technology ensures that the DP's authorization information is not tampered with in any form.

2. We utilize data fingerprints to safeguard DP data from being modified. By verifying the consistency of personal data stored on the SP through data fingerprints, we ensure the integrity of personal data.

3. We introduce a database to enhance query efficiency. We store authorization information from the blockchain and provide it to DP for querying. By verifying the query content with the authorization information generated from the blockchain, we can ensure that the query content is in line with the blockchain. We implemented a prototype using Ethereum and MySQL to evaluate the proposed solution and its efficiency. The experimental results demonstrate that VAIMS effectively supports various queries and ensures the authenticity of data.

2. Literature Review

2.1. Tamper-Proof Authorization Scheme

Nguyen B T et al. [5] introduced a data usage license issued by an intermediate manager in the blockchain as a prerequisite for using data, and recorded all data activities in a distributed ledger. However, the intermediate manager may be malicious and cannot guarantee security. Davari et al. [6] proposed a Blockchain-based compliance verification model that ensures that only parties authorized based on DP' consent can access DP' data. And all activities are logged in an immutable distributed ledger. The verification content before being stored on blockchain contains personal data, which may result in personal data being obtained by the companies without relevant records in the blockchain. Ricardo Neisse et al. [7] deployed publicly auditable smart contracts in the blockchain to record data usage policies and source tracking information. However, when revoking consent, the contract must be destroyed. Using a single contract for just one authorization can lead to system resources wastage. Xueping et al. [8] proposed a decentralized and trusted cloud data provenance architecture using blockchain technology which enabled the transparency of data accountability in the cloud, and helped to enhance the privacy and availability

of the provenance data. But the designed architecture is for keeping the creation and operations of a cloud data object secure and tamper-proof, and have no consideration on whether the data object is used in an authorized way. Guy Zyskind et al. [9] implemented access control by combining blockchain with off-chain storage. DP do not need to trust any third party and always know who is collecting data about them and how this data is used. Calani et al. utilize blockchain technology for the storage and maintenance of DP consent, employing smart contracts to rectify and revoke any consent provided by DP [10]. Mahindrakar has developed an automated verification system for ensuring compliance with authorizations using semantic web and Ethereum blockchain, capable of enforcing authorization compliance when data is shared with third parties [11]. Camilo et al. have conducted a proof-of-concept utilizing the Hyperledger Fabric framework to devise a model for consent management [12]. This model, facilitated by blockchain technology, furnishes tools for interaction among data subjects, data controllers, and data processors while upholding the rights of data subjects. Chiu et al. have proposed a scheme for secure data sharing that employs smart contracts as access control lists [13].

The above approach resolves the issue of using personal data within the authorized scope. Nevertheless, there are numerous matters that require attention. Their security is inadequate and typically only addresses either personal data tampering or consent tampering separately. On the other hand, efficiency has always been a bottleneck in the implementation of the scheme based on blockchain. DP cannot query their consent on the blockchain efficiently, which makes it challenging for them to ascertain whether the data is utilized within the authorized scope.

2.2. Efficient Query Scheme

In traditional blockchain schemes, each node usually maintains some state information, such as the earliest origin of the blockchain, the Bitcoin system [14], where nodes save the current balance status for each address. In this way, nodes can obtain the balance immediately without searching for records on the blockchain. The predominant blockchain underlying storage system, encompassing Bitcoin, Ethereum, and Hyperledger, utilizes Key-Value databases typified by LevelDB for data storage [15]. Key-Value databases make trade-offs in read performance to enhance write performance and exclusively cater to elementary key-value queries such as block and transaction inquiries. While the query performance is commendable, DP are unable to retrieve target data based on specific semantics due to the inadequacy of query semantics which poses a primary impediment in enhancing query performance. Ethereum [16] and Hyperledger [17] have extended the blockchain structure design for applications other than cryptocurrencies. It adds smart contracts that encode arbitrary and Turing-complete on top of the blockchain. Smart contracts store contract status on the blockchain and modify status by calling pre-coded transactions. Using smart contracts to store some state information can improve query efficiency.

In addition, there are other researchers [18,19] who are improving the experience and security of blockchain query services. However, limited by the design of the blockchain, while providing high security, its efficiency and query flexibility are far from meeting the needs of existing SP. There is still a need to solve how to provide rich and efficient query services for different needs.LineageChain [20] implements a fine-grained, secure and efficient provenance system for blockchains and provides a novel skip list index designed for supporting efficient provenance query processing. However, due to the structure of the blockchain itself, the space cost of the system has increased significantly. At the same time, there are databases such as BigchainDB [21] which leverages blockchains as a storage layer and introduces a database layer on top that extends blockchains can improve the performance and scalability when using blockchains for data sharing. EtherQL [22] is an efficient query layer based on Ethereum, provides efficient query primitives for analyzing blockchain data that can be integrated with other applications, and has high flexibility. However, both require nodes to trust the connected peers. These schemes cannot yet

provide proof of the authenticity of the query content and ensure that their output content is consistent with the query content in the blockchain.

Under the precondition of enhancing query efficiency, the majority of existing schemes are targeted at transactions in the blockchain. Some schemes have introduced smart contracts that can incorporate a certain quantity of authorization information. However, using additional space to store these contents in the blockchain to enhance query efficiency is overly "extravagant" in the case of scarce storage resources in the blockchain. Simultaneously, it is also intolerable to disregard the authenticity of query results for the sake of improving query efficiency.

3. Verifiable Authorization Information Management Scheme

This section presents VAIMS, which is devised to guarantee the authenticity of DP authorization and the integrity of personal data that is stored and utilized by the SP. It also furnishes evidence of malicious behavior when it occurs. Finally, this solution provides a trusted and efficient approach for querying. VAIMS consists of a blockchain module and a verifiable database module. The blockchain module stores authorization information and data fingerprints, while the smart contract governs the authorization process and precludes the impersonation of DP. The verifiable database module, integrated within the SP, stores authorization information on the chain and enables DP to conduct verifiable queries. As illustrated in Figure 2, DP apply for services to the SP, which then sends an authorization request to the DP. Subsequently, the SP dispatches a smart contract related to authorization to the customer, recorded on the blockchain. The specific operations are as follows:

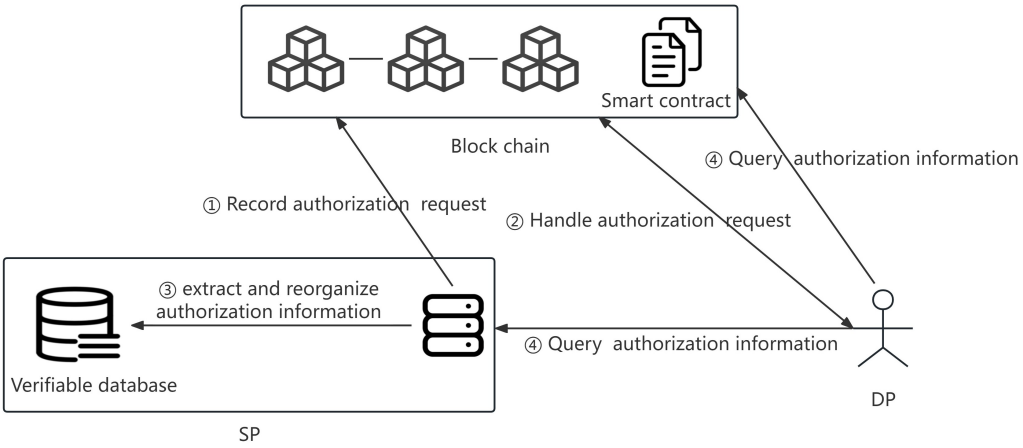


Figure 2. Overview of blockchain-based personal data authorisation management scheme.

1. The SP generates and transmits an authorization request transaction to the smart contract, utilizing the DP's service request as a basis.

2. The DP verifies the authorization request transaction. In the case of agreement, the DP submits the authorization consent transaction and the data fingerprint derived from personal data to the smart contract. If the DP disagrees with the authorization transaction, an authorization reject transaction is sent to the smart contract, thereby terminating the authorization process.

3. The SP retrieves authorization information (e.g., scope, purpose, time, etc.) from the smart contract and stores it in the database, while the DP can extract the corresponding authorization information to generate and store the authorization fingerprint.

4. Upon completion of the authorization process, the DP can securely transmit personal data to the SP, enabling the provision of services to the DP.

Upon the conclusion of Step 4, this paper validates the completion of both the authorization and the proof of authorization components, and the SP proceeds with the operation of services as stipulated in the existing authorization scheme. DP have the ability to retrieve their consent and personal data from either the blockchain or the SP. In the

event of retrieval from the SP, the stored authorization fingerprint and data fingerprint are utilized to verify the integrity of the consent and personal data, thus ensuring that they have not been tampered with. Furthermore, by comparing the provided personal data with the data fingerprint, the SP can verify if the DP has submitted a counterfeit version.

In our scheme, we propose combining blockchain and databases to offer a secure, verifiable, and efficient query scheme. Blockchain's cryptographic nature guarantees immutability, ensuring that once data is recorded, it cannot be altered without consensus. This provides a high level of security against data tampering and fraud, making it a reliable safeguard in ecosystems where data integrity is crucial. On the other hand, databases are optimized for fast data retrieval and processing, capable of handling large volumes of data and complex queries efficiently. Although blockchain has scalability challenges, offloading some data storage and processing tasks to databases can improve the overall scalability of the ecosystem.

3.1. Verifiable Database

The database is synchronized with the blockchain. In Figure 3, The table comprises attributes such as addressFrom, addressTo, authorizationMessage, blockHash, transactionHash, and blockNumber. The meaning of each attribute is described in Table 1.

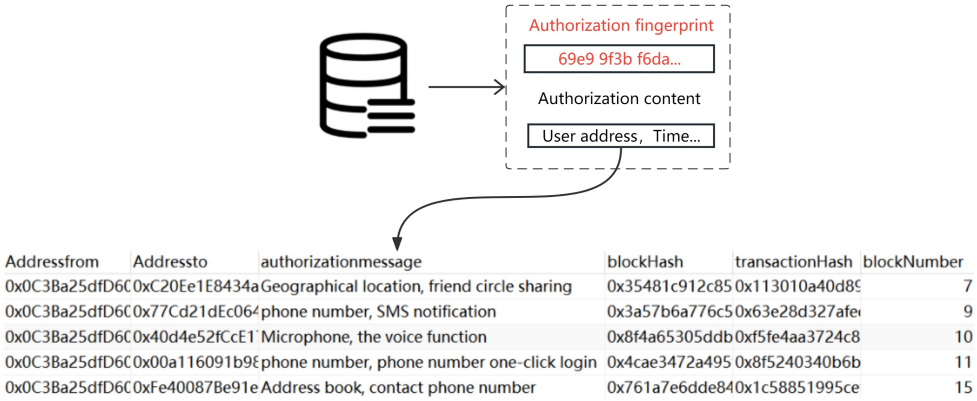


Figure 3. Off-chain verifiable database.

Table 1. Database properties correspond to stored content.

Attributes	Description
addressFrom	The address initiating the authorization request
addressTo	The DP's address
authorizationMessage	Authorization information(authorization permission and authorization purpose)
blockHash	The hash of the block where the permission operation is
transactionHash	The transaction hash of the authorization operation
blockNumber	The block height where the authorization operation is located

To guarantee consistency between the blockchain and the database, prevent the tampering of consents stored in the database, and offer reliable query services for SP DP, it is essential to store the authorization fingerprint for each DP. This step is vital in verifying the authenticity of database records and preventing deception on the SP.

3.2. Authorization Smart Contract

This subsection governs the authorization process through the use of a smart contract. The authorization process is as follows:

1. The SP creates an authorization request smart contract or utilizes a pre-existing one, fills in the relevant authorization information, and notifies the DP.

2. The DP examine the smart contract, assess its authorization information, and decide whether to grant consent.

3. If the DP consents to the authorization, the data fingerprint of the authorization data and the consent information are transmitted to the smart contract. This action initiates a contract event, prompting the contract to record the authorization, and stores information such as the authorization details and the data fingerprint.

4. If the DP rejects the authorization, the smart contract is terminated. It does not store anything and clears the authorization information for this process.

Since we have adopted a data authorization method based on the semaphore mechanism proposed by E.W. Dijkstra [23], the entire authorization process can only be accomplished when there is communication and trust between the SP and DP. We define a semaphore in the Authorization Smart Contract as an integer variable utilized to track the number of activities and ensure the proper execution of subsequent logic. The semaphore can only be incremented during the authorization process. If the semaphore does not match the operation's serial number, the Authorization Smart Contract cannot execute. Additionally, each operation and its execution result are submitted to the immutable smart contract log. This log acts as evidence to identify any malicious behavior, thereby enhancing the overall security of the authorization process.

3.3. Data Validation

3.3.1. Authorization Fingerprint

VQL recommends the construction of a unique authorization fingerprint for the database to ensure content consistency with the blockchain [24]. However, the VQL scheme involves generating fingerprints for both the overall database and the daily database. This approach requires DP to frequently verify and update their fingerprints, and complicates the access of DP consent across multiple databases. Consequently, we find it difficult to remember and query the fingerprint. To address this issue, this study proposes the construction of a distinctive authorization fingerprint for each DP's consent in the database. Rather than generating blocks over time, this method treats the DP's consent as the object. By generating authorization fingerprints from individual data, more flexibility is provided, eliminating the need for frequent updating and querying of new authorization fingerprints by individual DP. Verification of tampering in the database is achieved by comparing the hash value of the stored consent with the authorization fingerprint. This ensures data integrity and security.

3.3.2. Data Fingerprint

The generation and verification methods for data fingerprints are similar to those of authorization fingerprints. Data fingerprints act as a protective measure against the tampering of personal data. Before submitting personal data, data fingerprints are sent to smart contracts along with authorized consent transactions. Only the data fingerprint is stored on the blockchain, while the personal data remains undisclosed. This approach strengthens the protection of personal privacy. When personal data is tampered with, the data fingerprint stored in the blockchain serves as incontrovertible evidence, verifying the authenticity and integrity of the data.

3.3.3. Fingerprint Iteration

The system extracts authorization information from newly generated blocks within the blockchain and stores it as new authorization content in the database. As transactions and blocks are continuously generated, the SP's database needs timely updates to support DP query services. When a new block contains a DP's consent, the corresponding DP's authorization fingerprint needs to be updated, as outlined in Algorithm 1. Firstly, the fingerprint verification is compared with the hash value computed from the current DP's consent. Then, the new consent is stored in the database. Simultaneously, the hash value, which serves as the new fingerprint, is computed by hashing the concatenation of the verified consent (before updating) and the extracted consent from the new block.

Algorithm 1: Consent update

Input: BLK: Block in the blockchain; AF: $User_i$'s Authorization Fingerprint;
Consent: All of $user_i$'s consent; nConsent: The new block of $user_i$'s consent;
Output: nDF: $User_i$'s new Authorization Fingerprint;
Begin
  If BLK is new then
    Construct a new nConsent from BLK ;
    HASH (Consent);
    If HASH (Consents) = AF then
      Storage nConsent into DB;
      nAF = HASH (Consent + nConsent);
      Return nAF;
    Else
      Rollback;
    End if
  End if

3.3.4. Authorization Information Verification

After obtaining the DP's consent, the DP is required to undergo an authentication process, as shown in Algorithm 2. By comparing the hash value of the DP's consent in the database with the DP's consent in the blockchain or Authorization fingerprint, it is feasible to verify the consistency between the data in the database and in the blockchain. In most cases, DP can simply compare the newest local authorization fingerprint with the hash value generated from the consent in the database for verification. The hash value of the DP's consent in the blockchain needs to be calculated only when initializing or recovering of the local authorization fingerprint. To determine whether personal data has been tampered with, one can compare the corresponding data fingerprint from the blockchain with the hash value calculated from the personal data of the SP. If the authorization fingerprint verification fails, it indicates that the SP has modified the DP's consent in the database. Similarly, if the data fingerprint verification fails, it signifies that the SP has tampered with the personal data. In the event of verification failure, the consent, authorization fingerprint, and data fingerprint stored in the blockchain serve as evidence of the SP's tampering behavior. Armed with such evidence, DP can choose to accuse the SP or request the correction. Alternatively, The SP can prove that the personal data is submitted by DP.

Algorithm 2: Consent verification

Input: BLK: Block in the blockchain; DB: database; $ConsentD_i$: $User_i$'s consent constructed from database;
$ConsentB_i$: $User_i$'s consent constructed from BLK; AF: $User_i$'s Authorization Fingerprint;
Output: $ResultConsenti_i$: $User_i$'s consent verification result;
Begin
  If User = $User_i$ then
    Query ConsentDi from DB;
    Construct $ConsentB_i$ from BLKs;
  End if
  If HASH($ConsentD_i$)= HASH($ConsentB_i$) or AF= HASH($ConsentB_i$) then
    $ResultConsent_i$ = True;
  Else
    $ResultConsent_i$ = False;
  End if
  Return $ResultConsent_i$;
End

3.4. Data Authenticity Analysis

As DP receive query results from SP, as long as the database at the service provider for query is consistent with the underlying blockchain, the authenticity of the queried data is guaranteed. Therefore, we conduct data authenticity analysis from three aspects: user autonomy, database integrity, and verifiability of query results.

3.4.1. User Autonomy

In our scheme, the data verification of the SP is achieved by users through the verification methods provided by the scheme. Undoubtedly, this may consume a certain amount of computing resources and storage space. Consequently, the DP needs to have a reasonable motivation to verify the data. In the scheme presented in this paper, the data fingerprint is generated based on the personal authorization information of DP. Each DP only needs to maintain its own data fingerprint. Generally, there will be no other nodes maintaining data fingerprints for DP. The data fingerprint is designed to ensure the interests of users. To ensure that they are not deceived by the service provider, users will usually take the initiative to maintain the data fingerprint in a timely manner. This reflects the high degree of autonomy of users. They are not passively relying on external forces but actively safeguarding their own rights and interests by keeping a close eye on and maintaining their data fingerprints. They understand that by doing so, they can have better control over their data and interactions with the service provider, ensuring the integrity and security of their personal information and transactions.

3.4.2. Database Integrity

The consistency between the database provided by SP and the underlying blockchain data is accomplished through a data validation scheme. Whenever a new authorization is constructed, DP can either execute the scheme on their own or employ other nodes to execute it for backing up data and publishing fingerprints. At the same time, DP or nodes can establish a database of authorization information based on their own blockchain data and in accordance with the same rules. Subsequently, they can calculate their data fingerprints using a predefined hash function. If this data fingerprint is identical to the one provided by SP, the correctness of the query result can be verified. Moreover, since the authorization information will be written to the blockchain after verification, the integrity information is unalterable.

3.4.3. Verifiability of Query Results

After ensuring the integrity of the SP database through the verifiability of query results, the query results received by the DP should also be consistent with the SP's database. This scheme employs two methods to achieve the verifiability of query results: user verification in the database verification scheme and the simplified query result verification scheme. For user database verification in the database verification scheme, DP needs to download all blockchain data and check its consistency and conduct authenticity analysis just like other nodes. It is noteworthy that when DP requests data from other nodes, we will first connect to the anchor node in the blockchain network. This means that the other nodes we query are considered absolutely reliable. Therefore, the case of malicious nodes is insignificant and beyond the scope of this paper. In the simplified query result verification scheme, DP can utilize their own maintained data fingerprints. Since the data fingerprints are calculated locally by DP, the authenticity of the user's own authorization information in the database can be guaranteed. After verifying the authenticity of the database, all the user's authorization information in the service provider's database is taken as a new whole and queried on this basis. Finally, DP can effectively query the content. It is worth mentioning that the user's data fingerprint may be calculated locally. When DP verification fails, it does not mean that the service provider is dishonest or that the user is maliciously slandering the service provider. This may be due to problems that occur during the DP's maintenance of the data fingerprint.

4. Implementation Evaluation and Discussion

To test the feasibility and performance of our proposed system and algorithms, we implemented a prototype on Ganache, a personal blockchain network for Ether development and testing, along with MySQL. Our solution employs JavaScript to implement a user API (APIs can be integrated into a service layer to provide services) and service provider API by invoking the underlying blockchain API and the MySQL API. The user API offers various queries and database validations, including query interfaces for blocks, transactions, and validation interfaces. Meanwhile, the service provider supports query functions for gathering records from the blockchain through the blockchain API, such as data request interfaces for blocks, transactions, and the global state of the blockchain. The above operations and fingerprint generation are carried out using [specific method/tool]. We utilize the popular database MySQL as the data store for the service provider. We chose MySQL because it can support efficient querying of both generic data and rich data like arbitrary forms of transactions and smart contracts. Additionally, being a popular database, using it for scenario efficiency testing makes the experimental results more feasible. In order to evaluate the system performance without interference from network communication, we set up the experimental platform on a server equipped with an i7-11800H CPU and 16GB RAM. Considering different scenarios, we tested the query result validation time as well as various data query services, including throughput, block query, and transaction query.

4.1. Verification Time

In the experiments conducted in this paper, the Authorization fingerprint is generated using the SHA-256 algorithm, which exhibits linear time complexity, O(n), in relation to the size of the input. This implies that the time required to process input data is directly proportional to the length of the DP's data. As described in the previous chapter, DP must validate their transactions whether querying all transactions or specific ones by inputting the latest version of the Authorization fingerprint and completing the verification before any query. Should any modifications occur in the DP's authorization information, DP can readily detect inconsistencies between the computed SHA-256 value based on the authorization information and the stored Authorization fingerprint, thus identifying inaccuracies in the SP's database and avoiding potential misguidance from incorrect SP information. In Figure 4, In the experiments conducted in this paper, the Authorization fingerprint is generated using the SHA-256 algorithm, which exhibits linear time complexity, O(n), in relation to the size of the input. This implies that the time required to process input data is directly proportional to the length of the DP's data. As described in the previous chapter, DP must validate their transactions whether querying all transactions or specific ones by inputting the latest version of the Authorization fingerprint and completing the verification before any query. Should any modifications occur in the DP's authorization information, DP can readily detect inconsistencies between the computed SHA-256 value based on the authorization information and the stored Authorization fingerprint, thus identifying inaccuracies in the SP's database and avoiding potential misguidance from incorrect SP information.

In the subsequent experimental results of this paper, the results related to the query time are all derived from Formula (1). To obtain a more accurate query time, we use the average query time as the result. Therefore, we query randomly selected block lists or transaction lists.

$$Average\ Query\ Time = \frac{Total\ Time\ for\ Query\ all\ Transactions}{Number\ of\ transactions} \quad (1)$$

In this section, we analyzed the execution times of various operations during a query process. In our scheme, a complete query process includes verification time and query time. As shown in Figure 4, when querying the specified transaction through the database designed in this article's scheme, as the number of authorization information increases, The query time is not constant but has a relatively small fluctuation.the execution time

of the query specified transaction operation remains at around 2 ms. This is because the authorization information stored in the database needs to be consistent with that in the blockchain, and usually no modification is required. Database can be established to maintain a fast and stable query speed. The verification time increases linearly with the increase in the number of authorization information. However, considering that VAIMS enhances the security of authorization, and the verification time depends on the number of authorization information of the user rather than the overall number of authorizations. It is difficult for the user's valid authorization at a single service provider to reach a relatively high number. And when the DP needs to access a large amount of authorization data, the proportion of verification time in the total duration will decrease. Therefore, the relative increase in verification time is within an acceptable range.

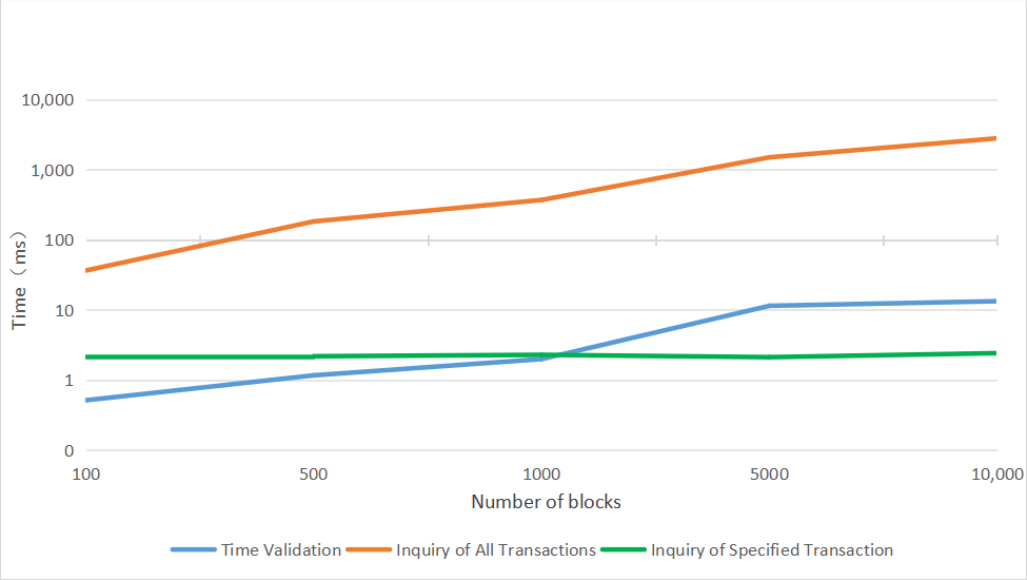


Figure 4. Verification Time and query time.

4.2. Throughput and Delay

We initially evaluated the throughput performance of the proposed system, VAIMS. We compared the throughput of VAIMS with other schemes that support queries. In Figure 5, we made a comparison between VAIMS and the implementation related to record retrieval discussed in [5]. The formula for calculating the throughput is as follows Formula (2).

$$Throughput = \frac{Amount\ of\ transactions}{Time\ duration} \quad (2)$$

Compared with the method proposed in this paper, the method in [5] has lower latency under low-load conditions. However, as the workload increases, the performance advantages of our method in various aspects of query execution become apparent. Due to the inclusion of a verification part in our scheme, which is a necessary time consumption in the query process, our scheme has higher latency than [5] in terms of delay under low-load situations. Our method exhibits stronger robustness in high-load environments. This is because we do not need to initiate transaction requests to the blockchain for querying. The introduction of a database brings a huge improvement in query efficiency to the scheme and enhances its throughput. The reason why blockchain systems cannot handle high workloads is that their own systems have local processing bottlenecks or transaction operations on the blockchain require sufficient peer responses. As can be seen from the figure, under a workload of 1000-tps, the throughput of our scheme can reach up to 986 tps at most. In addition to what is shown in the figure, we continued to test the throughput of our scheme as the workload continues to increase. When reaching 10,000-tps, VAIMS can still maintain a throughput of 8912 tps. When reaching 100,000-tps, the efficiency of VAIMS drops significantly and can

only be maintained at 42,936 tps. In the research of Xueping L et al. in [8], the experimental results show that querying 10 records with a total size of 1.004 KB takes an average of 221 ms. In contrast, when VAIMS queries 10 records with a total of 13.984 KB, the recorded average time is 243 ms. This finding indicates that even if the amount of data queried increases, VAIMS can maintain a similar efficiency to that in [8].

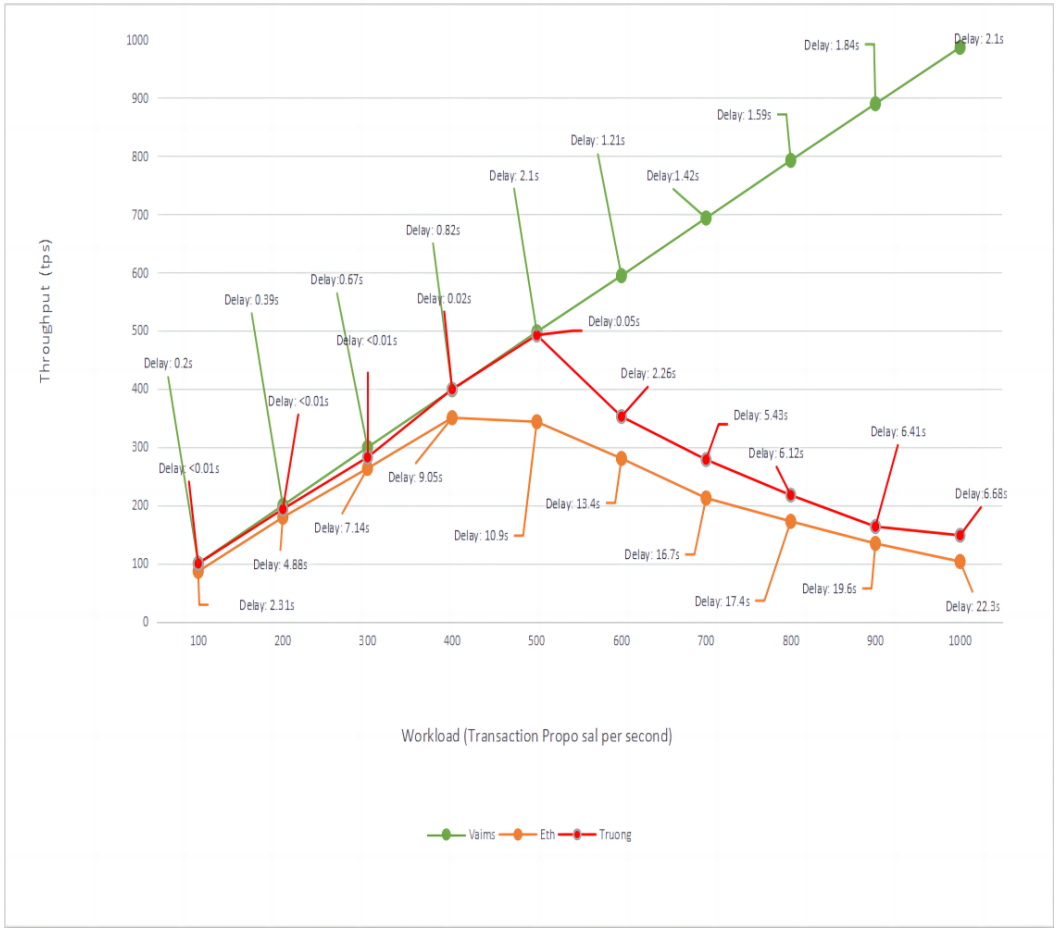


Figure 5. System performance under diverse workloads.

4.3. Total Transaction Query

We first evaluated the performance of the VAIMS system for querying authorized transactions. We compared the efficiency difference between Ethernet and the VAIMS query system we developed. In Figure 6, we tested different numbers of transactions, including 1000, 5000, 10,000, 50,000, 75,000, and 100,000 transactions. We also evaluated the impact on transaction query efficiency when the authorization success rate is set to 50% and 100%. As the number of transactions queried increases, the query time for Ethernet shows a significant increase. In contrast, the VAIMS system maintains a relatively low query time. This is because in blockchain querying, one must start from the genesis block and query along the chain until reaching the current block height to ensure no content is omitted. However, after introducing the database in VAIMS, related aspects need not be considered. Moreover, for users, only the information of successful authorizations needs to be stored. Invalid authorizations or rejected authorizations are not stored in the database. Storing only valid content further improves query efficiency. In the blockchain, whether the authorization is successful or not, the corresponding block will be added to the chain. As the chain length increases, the query efficiency of blockchain-based methods will decrease, further widening the gap in query time between VAIMS and Ethereum.

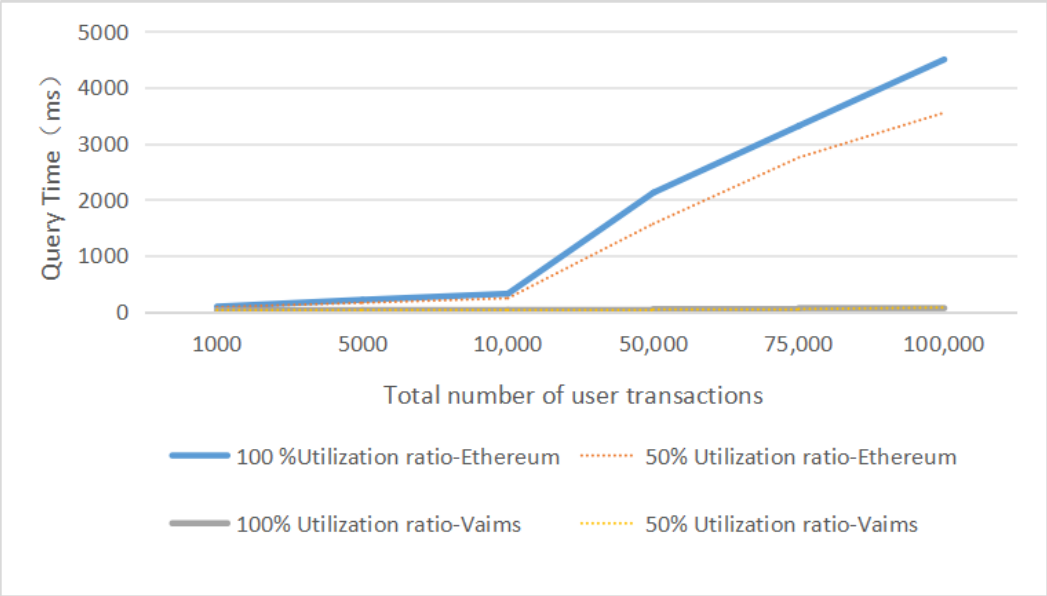


Figure 6. Performance Comparison of VAIMS and Ethernet in Querying Authorized Transactions.

4.4. Block Query

The blockchain is responsible for storing various DP-generated transactions in blocks. To demonstrate the query efficiency of our system, we initially compared the query time per block of different systems, such as Ethereum, and our proposed scheme. We conducted block query experiments for scenarios with block numbers ranging from 2000 to 200,000. Figure 7 illustrates the comparison of block query times between Ethereum and our system. In this study, we randomly selected a list of blocks to query and recorded the execution time of these queries. Ethereum experiences a significant increase in query time as the number of blocks increases, whereas our scheme maintains a relatively low level of query time complexity. While querying information in a specific block using Ethereum's API requires traversal from the first block to the target block, resulting in a substantial query time, our proposed system achieves faster queries by optimizing database data storage. Consequently, our system saves considerable query time.

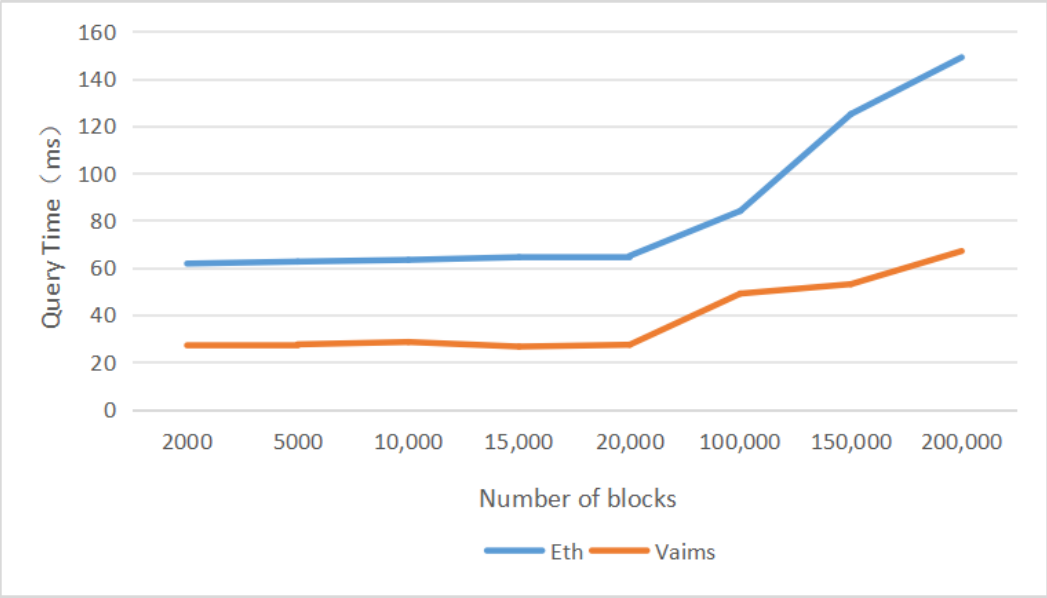


Figure 7. Block query time comparsion between Ethereum and VAIMS.

4.5. Transaction Query

In our proposed blockchain system, various types of authorization details are stored in blocks, including the acquisition of location information, DP contact information, and recording permissions. These authorization details negotiated in the contract are extracted from the blockchain and reconstructed in the database. We conducted experiments to measure the query time for individual transactions. Figure 8 illustrates a comparison of transaction query times between Ethereum and the Vaims scheme. In this study, we randomly selected a list of transactions to query and recorded the time taken to complete these queries. The experiments evaluated the query efficiency performance under different numbers of transactions, ranging from 1000, 2500, 5000, 7500, 10,000, 50,000, 75,000 and 100,000 transactions. Considering the existing magnitude of transactions, we calculated the query time for randomly accessing transactions. The results indicate that Ethereum's average query time increases with the number of transactions, while the Vaims scheme is less affected by the number of transactions, maintaining a relatively low query time. On the other hand, Ethereum experiences a significant increase in query time as more transactions are queried.

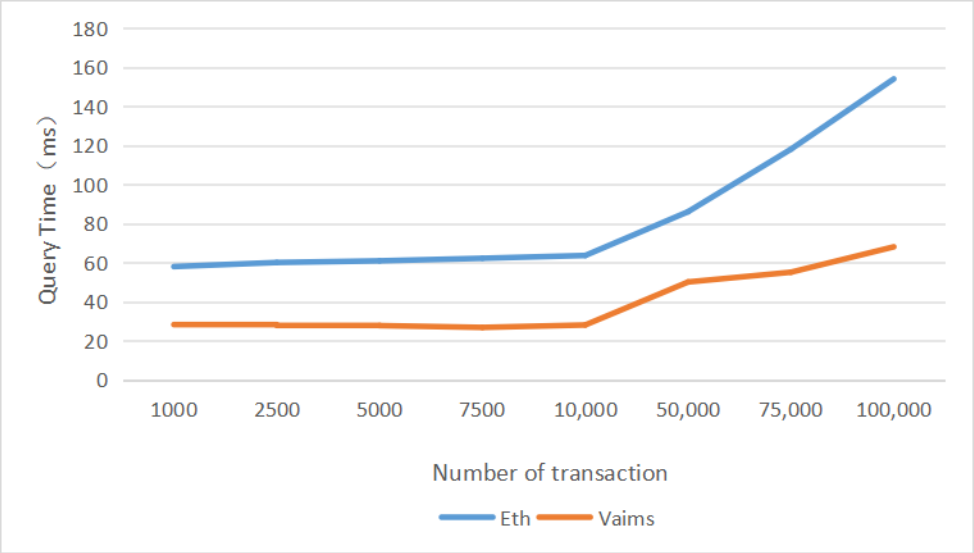


Figure 8. Transaction query time comparsion between Ethereum and VAIMS.

4.6. Smart Contract Cost

In the implemented smart contract, the actual cost is defined as follows: Gas usage multiplied by Gas price equals Ether. In our case, no specific gas price is involved; we only calculate the gas usage, which depends on the employed blockchain. The following table summarizes the costs of all executed operations. In the current Ethereum, the average transaction consumes approximately 21,000 units of gas. Smart contracts with complex operations typically consume more than ordinary transactions. In Table 2 below, it can be noted that more gas is consumed when creating the authorization request and consenting for the first time, as some variables are initially created in these two operations. The storage of authorization information in our contract operation also incurs some gas consumption.

Table 2. Smart Contract Cost Analysis.

Operation	Gas Used
create-contract	732,000
first create-request	50,503
first request-access	40,690
create-request	30,603
request-access	23,590
request-reject	10,550

4.7. Experiment Summary and Outlook

In this experiment, Ganache was deliberately chosen to simulate the blockchain environment and facilitate subsequent tests. The prime motivation for this selection lies in the fact that Ganache offers a visually intuitive interface, which significantly eases our direct perception and understanding of the blockchain status. During the testing stage, it empowers us to monitor freshly added transactions and undertake smart contract debugging at any specific moment.But this is not necessary.If you wish to replace this section you can use Hardhat for testing, which is recommended by Truffle Suite, the developer of Ganache.

Ethereum was selected as the blockchain component within the scheme, primarily because our aim was to leverage the public chain paradigm. The public chain proves highly suitable as the underlying framework for the context of distrusting service providers delineated in this article. Renowned for its wide user base, Ethereum supports multiple Turing-complete programming languages, such as Solidity, which effectively aids in the implementation of the smart contract aspect. Consequently, our proposed scheme is not restricted to being exclusively implemented on Ethereum.It is crucial to note that the public chain does encounter challenges, including low efficiency and Gas consumption. Hence, the adoption of the consortium chain emerges as a viable alternative, as it holds the potential to enhance the authorization efficiency within the scheme and curtail energy consumption.

To substantiate the feasibility of the scheme's efficiency, MySQL, a prevalently utilized open-source database, was selected. Regarding the calculation of query efficiency, no optimizations were executed at the database level in this article to augment the query efficiency of the proposed scheme.

In this experimental endeavor, popular tools were deliberately chosen for the prototype design to showcase the usability in terms of the scheme's efficiency. When the requisite scenarios vary, the corresponding blockchain and database components can be substituted to achieve targeted and substantive efficiency enhancements.

5. Conclusions

This paper introduces a verification scheme for managing authorisation information, which achieves transparency, traceability and non-tampering of authorisation operations through the application of smart contracts and blockchain technology, ensuring the security of the authorisation process and the authenticity of the authorisation. The introduction of fingerprint technology and the storage of data fingerprints in the blockchain can constrain the SP behaviour to ensure the integrity of personal data. Through the combination of authorisation fingerprint and database technology, a trusted query is provided, which reduces the necessity of blockchain access, and finally the scheme of this paper is proved through experiments to improve the query efficiency.

6. Future Work

Although we have effectively implemented the right to be informed as mandated by the General Data Protection Regulation (GDPR). Moreover, by utilizing smart contracts, we have constructed a transparent authorization process between users and service providers, accomplishing authorization record-keeping. Additionally, through verifiable queries, we have enhanced user query efficiency while maintaining the security of the scheme intact. Nevertheless, there remains a need for improvement and expansion of the scheme's scope. In the future, we will strive to realize users' data control rights and the right to be forgotten. We will refine the authorization process by leveraging decentralized technologies and cryptographic techniques. This will involve addressing issues such as data sharing that safeguards user privacy, enhancing scheme scalability, and ensuring flexible data control rights. By addressing these challenges, our research aims to make a significant contribution to the continuous development of personal information security through decentralized technologies. These schemes are expected to provide valuable insights for countries in their efforts to protect personal information.

Author Contributions: Conceptualization, J.Y.; Software, J.Y. and X.L.; Writing—original draft, J.Y. and Y.G.; Writing—review & editing, X.L. and Y.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Natural Science Foundation of China (Project No. 62272174, "Research on lattice-based asymmetric ciphertext retrieval technology", Principal Investigator: Huang Qiong, Affiliation: College of Mathematics and Informatics (Software), South China Agricultural University) and the Key Research and Development Project of Guangxi Science and Technology Plan (Project No. GIIP2309, "Low-light image character recognition for intelligent elderly care services").

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding authors.

Acknowledgments: Parts of this manuscript have used Google Translator or Grammarly for translating from Chinese and for improving the resulting text, before being read and finalized by human hand by the authors.

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

Abbreviations

The following abbreviations are used in this manuscript:

SP	Service Provider
DP	Data Provider

References

1. Hao, L.; Min, Z.; Dengguo, F.; Zhu, H. Research on Access Control of Big Data. *Chin. J. Comput.* **2017**, *40*, 72–91.
2. Papageorgiou, A.; Strigkos, M.; Politou, E.; Alepis, E.; Solanas, A.; Patsakis, C. Security and privacy analysis of mobile health applications: The alarming state of practice. *IEEE Access* **2018**, *6*, 9390–9403. [CrossRef]
3. Nguyen, T.T.; Backes, M.; Marnau, N.; Stock, B. Share First, Ask Later (or Never?) Studying Violations of GDPR's Explicit Consent in Android Apps. In Proceedings of the 30th USENIX Security Symposium (USENIX Security 21), Vancouver, BC, Canada, 11–13 August 2021; pp. 3667–3684.
4. Sakamoto, T.; Matsunaga, M. After GDPR, still tracking or not? Understanding opt-out states for online behavioral advertising. In Proceedings of the 2019 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 19–23 May 2019; pp. 92–99.
5. Truong, N.B.; Sun, K.; Lee, G.M.; Guo, Y. Gdpr-compliant personal data management: A blockchain-based solution. *IEEE Trans. Inf. Forensics Secur.* **2019**, *15*, 1746–1761. [CrossRef]
6. Davari, M.; Bertino, E. Access control model extensions to support data privacy protection based on GDPR. In Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 9–12 December 2019; pp. 4017–4024.
7. Neisse, R.; Steri, G.; Nai-Fovino, I. A blockchain-based approach for data accountability and provenance tracking. In Proceedings of the 12th International Conference on Availability, Reliability and Security, Reggio Calabria, Italy, 29 August–1 September 2017; pp. 1–10.
8. Liang, X.; Shetty, S.; Tosh, D.; Kamhoua, C.; Kwiat, K.; Njilla, L. Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In Proceedings of the 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), Madrid, Spain, 14–17 May 2017; pp. 468–477.
9. Zyskind, G.; Nathan, O. Decentralizing privacy: Using blockchain to protect personal data. In Proceedings of the 2015 IEEE Security and Privacy Workshops, San Jose, CA, USA, 21–22 May 2015; pp. 180–184.
10. Calani, M.; Denaro, G.; Leporati, A. Exploiting the Blockchain to Guarantee GDPR Compliance while Consents Evolve under Data Owners' Control. In Proceedings of the ITASEC, Online, 7–9 April 2021; pp. 331–343.
11. Mahindrakar, A.; Joshi, K.P. Automating GDPR compliance using policy integrated blockchain. In Proceedings of the 2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), Baltimore, MD, USA, 25–27 May 2020; pp. 86–93.
12. Camilo, J. Blockchain-Based Consent Manager for GDPR Compliance. In *Open Identity Summit 2019*; Lecture Notes in Informatics (LNI); Gesellschaft für Informatik: Bonn, Germany, 2019.
13. Chiu, W.Y.; Meng, W.; Jensen, C.D. My data, my control: A secure data sharing and access scheme over blockchain. *J. Inf. Secur. Appl.* **2021**, *63*, 103020. [CrossRef]
14. Garay, J.; Kiayias, A.; Leonardos, N. The bitcoin backbone protocol: Analysis and applications. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, 26–30 April 2015; pp. 281–310.

15. Shao, Q.F.; Jin, C.Q.; Zhang, Z.; Qian, W.N.; Zhou, A.Y. Blockchain: Architecture and research progress. *Chin. J. Comput.* **2018**, *41*, 969–988.
16. Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
17. Peregrina-Pérez, M.J.; Lagares-Galán, J.; Boubeta-Puig, J. Hyperledger Fabric blockchain platform. In *Distributed Computing to Blockchain*; Elsevier: Amsterdam, The Netherlands, 2023; pp. 283–295.
18. Xu, C.; Zhang, C.; Xu, J. vchain: Enabling verifiable boolean range queries over blockchain databases. In Proceedings of the 2019 International Conference on Management of Data, Amsterdam, The Netherlands, 30 June–5 July 2019; pp. 141–158.
19. Muzammal, M.; Qu, Q.; Nasrulin, B. Renovating blockchain with distributed databases: An open source system. *Future Gener. Comput. Syst.* **2019**, *90*, 105–117. [CrossRef]
20. Ruan, P.; Chen, G.; Dinh, T.T.A.; Lin, Q.; Ooi, B.C.; Zhang, M. Fine-grained, secure and efficient data provenance on blockchain systems. *Proc. VLDB Endow.* **2019**, *12*, 975–988. [CrossRef]
21. El-Hindi, M.; Binnig, C.; Arasu, A.; Kossmann, D.; Ramamurthy, R. BlockchainDB: A shared database on blockchains. *Proc. VLDB Endow.* **2019**, *12*, 1597–1609. [CrossRef]
22. Li, Y.; Zheng, K.; Yan, Y.; Liu, Q.; Zhou, X. EtherQL: A query layer for blockchain system. In Proceedings of the Database Systems for Advanced Applications: 22nd International Conference, DASFAA 2017, Suzhou, China, 27–30 March 2017; Proceedings, Part II 22; pp. 556–567.
23. Stark, E.W. Semaphore primitives and starvation-free mutual exclusion. *J. ACM (JACM)* **1982**, *29*, 1049–1072. [CrossRef]
24. Peng, Z.; Wu, H.; Xiao, B.; Guo, S. VQL: Providing query efficiency and data authenticity in blockchain systems. In Proceedings of the 2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW), Macao, China, 8–12 April 2019; pp. 1–6.

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.