International Journal of
Geo-Information

Article

Spatio-Temporal Big Data Collaborative Storage Mechanism Based on Incremental Aggregation Subvector Commitment in On-Chain and Off-Chain Systems

Mingjia Han, Xinyi Yang, Huachang Su, Yekang Zhao, Ding Huang and Yongjun Ren *

School of Computer Science, Nanjing University of Information Science and Technology, Nanjing 210044, China
* Correspondence: renyj100@126.com

Abstract: As mobile internet and Internet of Things technologies rapidly advance, the amount of spatio-temporal big data have surged, and efficient and secure management solutions are urgently needed. Although cloud storage provides convenience, it also brings significant data security challenges. Blockchain technology is an ideal choice for processing large-scale spatio-temporal big data due to its unique security features, but its storage scalability is limited because the data need to be replicated throughout the network. To solve this problem, a common approach is to combine blockchain with off-chain storage to form a hybrid storage blockchain. However, these solutions cannot guarantee the authenticity, integrity, and consistency of on-chain and off-chain data storage, and preprocessing is required in the setup phase to generate public parameters proportional to the data length, which increases the computational burden and reduces transmission efficiency. Therefore, this paper proposes a collaborative storage mechanism for spatio-temporal big data based on incremental aggregation sub-vector commitments, which uses vector commitment binding technology to ensure the secure storage of on-chain and off-chain data. By generating public parameters of fixed length, the computational complexity is reduced and the communication efficiency is improved while improving the security of the system. In addition, we design an aggregation proof protocol that integrates aggregation algorithms and smart contracts to improve the efficiency of data query and verification and ensure the consistency and integrity of spatio-temporal big data storage. Finally, simulation experiments verify the correctness and security of the proposed protocol, providing a solid foundation for the blockchain-based spatio-temporal big data storage system.

Keywords: spatio-temporal big data; subvector commitment; blockchain; smart construct

check for updates

Citation: Han, M.; Yang, X.; Su, H.; Zhao, Y.; Huang, D.; Ren, Y. Spatio-Temporal Big Data Collaborative Storage Mechanism Based on Incremental Aggregation Subvector Commitment in On-Chain and Off-Chain Systems. *ISPRS Int. J. Geo-Inf.* **2024**, *13*, 280. https://doi.org/10.3390/ijgi13080280

Academic Editors: Wolfgang Kainz, Jun Feng, Changqing Luo and Mamoun Alazab

Received: 5 July 2024
Revised: 5 August 2024
Accepted: 7 August 2024
Published: 10 August 2024

1. Introduction

In the era of information explosion, big spatio-temporal data have become an important branch in the field of data science. Big spatio-temporal data refer to those large datasets that possess temporal and spatial attributes. This type of data are usually captured or generated at specific times and locations and can provide detailed information about the timing and geographical position of objects or events. As mobile internet and IoT technologies rapidly advance, the Internet of Things (IoT) and big data technologies, especially the widespread application of the Global Positioning System (GPS), generate a vast amount of data labeled with time and geographical information. These datasets are not only large but also update rapidly. Big spatio-temporal data not only demand greater storage capacity but also pose unprecedented challenges to data security and privacy protection [1]. At present, although cloud storage provides a solution for the massive volumes of big spatio-temporal data, it also introduces several security issues [2]. Firstly, the separation of data ownership and management makes big spatio-temporal data easily accessible to unauthorized third parties and even susceptible to tampering and deletion. Secondly, cloud service providers may cause data leakage due to internal vulnerabilities or external attacks, which is particularly critical for big spatio-temporal data that include time and geographical information [3]. Additionally, cloud storage lacks a transparent data deletion mechanism, making it impossible for users to ensure their data are truly and completely removed upon request, thus raising the risk of data misuse or leakage [4,5]. Consequently, as the volume of big spatio-temporal data increases, traditional data security measures can no longer effectively handle the increasingly complex security threats, urgently requiring new technologies and strategies to ensure the secure management of extensive big spatio-temporal data.

Blockchain is a distributed database technology originally designed as the underlying technology for Bitcoin. It allows data to be stored in blocks, which are cryptographically linked together to form a growing chain. Each block contains a series of transaction records, and the data are kept secure and tamper-proof through complex encryption algorithms. The fundamental attributes of blockchain technology encompass decentralization, openness, immutability, and anonymity [6,7]. After years of development, blockchain technology is not only limited to the initial application of digital currency, but its reach has broadened to encompass diverse fields like financial services, supply chain management, smart contracts, identity verification, and medical record management. It provides new possibilities for solving trust problems, improving efficiency, and reducing costs [8].

Given the exponential rise in big spatio-temporal data, the decentralized nature of blockchain brings advantages in security and transparency but also poses challenges in storage efficiency and scalability [9]. In blockchain, each node must maintain a copy of the entire chain, and with the escalating volume of transactions and data, the data volume of each block also grows. This requires that each participating node have sufficient storage space to maintain the complete blockchain. Therefore, the native design of blockchain is not well-suited for directly handling large-scale, spatio-temporal data.

Currently, on-chain storage and off-chain storage are the focus of blockchain scaling technology research. Compared with on-chain storage, off-chain storage significantly improves the scalability of blockchain from the standpoint of storage capacity. By transferring data from on-chain to off-chain storage of blockchain, it not only improves the scalability but also enhances the utilization efficiency of on-chain storage space. However, off-chain storage solutions still face some challenges. On the one hand, although off-chain storage enhances the scalability of the blockchain, the off-chain storage system is detached from the blockchain, and it can only obtain data from the off-chain storage system without verifying the authenticity of the data [10]. On the other hand, the existing off-chain storage scheme lacks the on-chain and off-chain data consistency and integrity guarantee mechanisms and suffers from the problems of excessively long public parameters, fixed off-chain storage capacity, and difficulty in expansion. The contributions of this paper are as follows: An on-chain and off-chain collaborative storage mechanism based on incremental aggregatable sub-vector commitments is proposed. It improves the on-chain and off-chain collaborative storage scheme built on updatable sub-vector commitments, making it have fixed-length public parameters and improving communication efficiency. It also combines smart contracts to design an aggregation proof protocol to improve query and verification efficiency.

The rest of this paper is structured as follows: In Section 2, we delineate the related research and progress lineage of blockchain storage scalability. In Section 3, we describe the security problems of blockchain on-chain and off-chain cooperative storage of big spatio-temporal data and propose a solution for this paper. Section 4 presents a detailed system overview of spatio-temporal big data collaborative storage mechanisms based on incremental aggregation subvector commitment in on-chain and off-chain systems. Section 5 describes the specific scheme design, scheme construction, and aggregation proof protocol, and finally gives the correctness and security analysis. In Section 6, the paper presents an analysis of the performance of the proposed scheme.

2. Related Works

Chen and Sun et al. [11,12] outlined several approaches to enhance the expandability of blockchain storage systems, including off-chain storage, on-chain storage, and network scaling. The off-chain storage method is realized by migrating the blockchain data to an external storage system, in which the blockchain only retains references (also known as "pointers") to these data and some necessary non-data information instead of all data, which greatly enhances the scalability of the blockchain. The implementation of off-chain storage mainly includes the use of Distributed Hash Table (DHT), Inter-Planetary File System (IPFS) and cloud-based storage solutions.

DHT-based off-chain storage uses distributed hash table technology to achieve decentralized storage and fast retrieval of data using decentralized networks, where each network node holds only a portion of the data to improve storage efficiency and data access speed. Chao-Ran Luo et al. [13] developed vRoute, a data and location decoupling algorithm for DHT-based heterogeneous peer-to-peer (P2P) networks. vRoute is used in the indexing phase of the algorithm, where indexing messages are sent to the neighboring nodes through DHT rules and a reverse routing table is built using Bloom Filter. In the query phase, addressing is done through the forward routing table first, and if an index exists in the reverse routing table, the data continues to be traced based on the index until the data is found or it is confirmed that no match exists. Wu et al. [14], on the other hand, proposed a two-chain DHT blockchain architecture, including a data chain for holding data content and an index chain tasked with recording data indexing and validation details, with all storage nodes collectively creating a DHT-based data storage network, where each index block is connected to the corresponding data block via a DHT key.The LightChain blockchain architecture proposed by Hassanzadeh-Nazarabadi et al. [15] is also based on DHT, where blocks and transaction information are replicated among peer nodes of the DHT network. Each node is responsible for storing only a small amount of block and transaction information randomly assigned to it and accessing the transaction and block copies of other nodes as needed.

Off-chain storage based on IPFS enables distributed storage of data on nodes globally by creating a decentralized file system. This approach uses content-addressing techniques to efficiently retrieve and persistently store data, thus speeding up data access. Li et al. [16] used IPFS technology to store IoT data by storing the data directly on IPFS while recording hash values corresponding to these IPFS files on the blockchain and creating hash-based links between different blocks.Mahmud et al. [17,18] developed a dual blockchain system based on IPFS that allows users to upload transactions to IPFS, which subsequently generates and returns a small-sized representation of the content to the user. The user can then share these content representations with neighboring miners for transaction validation, and successfully validated transactions are added to the block.

Cloud-based off-chain storage technology combines blockchain data and cloud storage services to reduce the load on the blockchain by storing data on a cloud platform while ensuring data accessibility and security [18,19]. Feng et al. [20] introduced a blockchain-based dual verifiable cloud storage strategy that utilizes hash functions with homomorphic properties to allow cloud service providers to integrate the signatures of different users and verify the merged data. Zhou et al. [21] designed a blockchain-based fine-grained cloud data secure deletion scheme that employs attribute-based encryption based on ciphertext policy to allow data owners to precisely control access privileges and ensure the verifiability of data deletion operations. Zhang et al. [22], on the other hand, proposed a blockchain-based data auditing scheme that aims at identifying malicious multi-cloud storage service providers and realizes batch verification of data stored by multiple cloud service providers through homomorphic verifiable tags.

Compared to off-chain storage, on-chain storage employs a decentralized method where data is distributed across various nodes in the network [23], with each node storing a portion of the data based on specific rules. On-chain storage is mainly categorized into two modes: collaborative storage mode and node-light mode. In the collaborative storage mode, multiple nodes work together to store and manage data: each node is tasked with holding a segment of the data and validating the data with other nodes. The main approaches to collaborative storage include techniques such as encoding, clustering, and sharding.

In shard-based collaborative storage, data are divided into multiple small segments, known as shards, and dispersed across various nodes in the network. This approach aims to enhance storage efficiency and system scalability. Wu et al. [24] developed a consortium blockchain framework called BETASCO, which focuses on the sharded processing of smart contracts. The system allocates an independent shard environment for each smart contract and employs distributed hash technology to locate contracts, ensuring that each transaction is accurately delivered to the corresponding smart contract shard. Zheng [25] conducted a study on a sharded consortium blockchain system named Meepo, which utilizes cross-timing and cross-calling techniques to improve operational efficiency across shards. In coding-based collaborative storage, data are partitioned and coded and then distributed and stored on multiple nodes of the network, which enhances the reliability and fault tolerance of the data. Lajam et al. [26] explored how to apply network coding techniques to P2P content distribution systems. Fan et al. [27] developed a latency-aware coded data block allocation algorithm that specifically deals with the problem of selecting the number and location of coded data blocks to be stored.

Cluster-based collaborative storage stores and manages data by forming clusters of multiple nodes. This approach enables nodes to share data storage and validation tasks together, which enhances the system's reliability and scalability [28]. Li et al. [29] developed a multi-node collaborative storage strategy called ICIStrategy, which divides all the participants into a number of clusters, and each cluster holds the network's copy of all the data in the network. Every node in the cluster does not need to maintain a full data duplicate, thereby decreasing the scale of data each node must store, relieving the storage pressure, and lowering the communication overhead within the clusters. Mamun et al. [30] designed a lightweight and scalable consensus protocol and distributed the data blocks in a balanced manner among the clusters through a parallel processing module. The nodes are tasked with verifying the blocks and communicating with the distributed consensus protocol through the Message Passing Interface (MPI) communicator to exchange consensus information with the distributed consensus manager and their respective coordinators. The validated data blocks will be stored in the node's local memory and other nodes to enhance the overall integrity and reliability of the data.

The light node model allows light nodes to perform transaction verification without having to download data from the entire blockchain [31]. This is achieved by synchronizing only block header information and employing Simplified Payment Verification (SPV), which allows light nodes to confirm the validity of a transaction without the need for large amounts of storage space. Wu et al. [32] designed a blockchain system consisting entirely of light nodes, where it is assumed that any two nodes are able to perform data transfer either directly or via broadcast. In order to reduce the network load, they used a Byzantine fault-tolerant algorithm for block validation, which sends new block information to only a few designated validation nodes without broadcasting it to the entire network.

Network scaling utilizes relay networks and the Open System Interconnection Reference Model (OSI) for Internet Communication to improve the blockchain's scalability without modifying the blockchain's upper-layer architecture. A relay network is a transmission network whose main function is to achieve faster data transmission speed by accelerating the block transmission process. Conversely, the OSI model, enhances the administration and transmission of blockchain data by refining the underlying data transmission protocol, thereby boosting the overall expandability of the system. Thai et al. [33] developed a blockchain data transmission protocol based on a named data network that takes full advantage of the highly efficient transmission capabilities of the named data network for transaction and block delivery and activates the in-network caching and the built-in multicast function, which conveniently realizes the distribution of content.

3. Problem Statement

Spatio-temporal big data is a complex and rich data set that contains a large amount of information related to time and space and has important application value in many fields. However, the storage of spatio-temporal big data faces the following specific challenges: First, the data volume is huge, including a large amount of geographic information, sensor data, and real-time data, resulting in low storage and processing efficiency; second, insufficient scalability, rapid data growth, and difficulty in meeting high scalability storage requirements; in addition, real-time processing and access are required, but the synchronization and verification speed of on-chain storage is slow, making it difficult to meet real-time requirements; at the same time, data privacy and security issues cannot be ignored, and the public storage of sensitive information will bring risks of privacy leakage. In addition, spatio-temporal big data types are diverse, including text, images, videos, and sensor data, and the blockchain storage format is single, making it difficult to effectively process these data; furthermore, historical data management requires long-term data preservation for analysis and backtracking, but storing all historical data on the chain will increase the burden and affect performance; in addition, legal compliance requirements are complex, and the immutability and openness of the blockchain may conflict with data privacy protection and deletion requirements; finally, cross-system data sharing and interoperability require data consistency and traceability, but it is difficult to achieve efficient cross-system data sharing and verification by relying solely on on-chain storage. Therefore, the on-chain and off-chain collaborative storage systems came into being, which store key transaction records and verification information on the chain to ensure data security and integrity and store large-scale non-critical data on the chain to reduce costs and improve efficiency, thereby comprehensively utilizing the advantages of cloud storage and blockchain to achieve efficient, flexible, and secure spatio-temporal big data management and application.

Blockchain's on-chain and off-chain cooperative storage combine both on-chain and off-chain data storage. Blockchain's on-chain data storage cost is relatively high, and by storing part of the data off-chain, it can significantly reduce the transaction cost and storage cost. Storing data off-chain means that most data is stored outside the blockchain, while only necessary metadata and hash values are stored on-chain for verifying the integrity of off-chain data. This approach reduces the volume of data on the blockchain, lowering transaction costs and storage requirements while maintaining the security and reliability of on-chain data. Off-chain storage typically uses traditional databases or distributed storage systems, while on-chain data is secured using blockchain technology to ensure data immutability and transparency. However, the synchronization of on-chain and off-chain data may lead to data inconsistency problems, and additional mechanisms are needed to guarantee data integrity and consistency. Existing on-chain-off-chain collaborative storage schemes require preprocessing of the stored data during the setup phase, and the length of the generated public parameters is linearly related to the length of the submitted data, which not only increases the computational burden, but also leads to a significant limitation of the transmission efficiency when the data volume is too large, affecting the communication efficiency of the system.

To address the aforementioned issues, this paper proposes a spatio-temporal big data collaborative storage mechanism based on incremental aggregation subvector commitment in on-chain and off-chain systems, effectively ensuring the consistency and integrity of data stored on-chain and off-chain in blockchain systems.

4. Preliminaries

4.1. Blockchain Technology

Blockchain technology is a distributed database system that allows multiple participants to jointly maintain a reliable, immutable data record without the need for a central authority. The core of the technology lies in its data structure: a series of "blocks" arranged in chronological order and connected by cryptographic means, each of which contains a certain number of transaction records. When new transactions occur and are verified by network participants (nodes), they will be added to a new block, and the block will be linked to the existing blockchain.

The blockchain data structure is a method of storing data in a chain. It consists of multiple interconnected "blocks" forming a continuous chain. Each block contains a set of transaction data and two main parts: a block header and a block body. The block header contains information such as the hash value of the previous block, a timestamp, a difficulty target, and a random number called a "nonce", which together ensure the security and data consistency of the blockchain. The block body stores the actual transaction information. Through this structure, the blockchain achieves immutability and decentralized storage of data. Therefore, blockchain has many advantages:

(1) Decentralization: Blockchain technology eliminates the need for a centralized management entity and increases the transparency and security of the system.

(2) Immutability: Once data is recorded in a block and added to the chain, it cannot be changed or deleted, ensuring the integrity and accuracy of the data.

(3) Transparency: All network participants can view transactions and history records on the blockchain, but user identities remain anonymous or pseudo-anonymous.

(4) Security: Blockchain uses encryption technology and consensus mechanisms to ensure the security of transactions and the overall security of the network.

4.2. Incrementally Aggregatable Subvector Commitments

Here, we give an implementation of incrementally aggregatable subvector commitments based on the RSA assumption within groups of unknown order. Incremental aggregatable subvectors commit to having parameters of constant size, unrelated to the vector length, and similarly, the verification time is constant. In order for the scheme to have constant-sized parameters, the prover is made to calculate S_i and incorporate it in the opening at position i. To prevent the adversary from generating the wrong S_i, we store the public parameter $U_n = g^{\prod_{i\in[n]} e_i}$ as an accumulator over all positions. The prover can correctly verify that S_i passes through $S_i^{e_n} = U_n$ as an accumulator over all positions. The prover can correctly verify that S_i passes through $S_i^{e_n} = U_n$ in a constant time.

The incrementally aggregatable subvector commitment defined here consists of the following algorithm: ($VC.Setup$, $VC.Specialize$, $VC.Com$, $VC.Open$, $VC.Verify$, $VC.Disagg$, $VC.Agg$).

The setup algorithm $VC.Setup(1^\lambda, 1, n) \rightarrow crs$ produces a group of hidden order $G \leftarrow Ggen(1^\lambda)$, a generator $g \leftarrow \mathcal{R}\ G$. Furthermore, it defines a deterministic collision function, $PrimeGen$, which maps integers to prime numbers.

$VC.Specialize(crs, n) \rightarrow crs_n$: the specialize algorithm calculates $n(l+1)$-bit primes $e_1, \ldots, e_n,\quad e_i \leftarrow PrimeGen(i)$ through $PrimeGen$ algorithm, where $i \in [n], U_n = g^{e_{[n]}}$, returns $crs_n \leftarrow (crs, U_n)$. $U_n$can be designated as an accumulator for the set $[n]$.

$VC.Com(crs, \vec{v}) \rightarrow (C, aux)$: the commitment algorithm computes $S_i \leftarrow g^{e_{[n]\setminus\{i\}}}$ for all $i \in [n], C \leftarrow S_1^{v_1} \ldots S_n^{v_n}$ and $aux \leftarrow (v_1, \ldots, v_n)$.

$VC.Open(crs, I, \vec{y}, aux) \rightarrow \pi_{\mathrm{I}}$: the open algorithm generates for each $j \in [n]\setminus I, S_j^{1/e_I} \leftarrow g^{e_{[n](I\cup\{i\})}}$ and $S_I \leftarrow g^{e_{[n]I}}$ and then returns $\pi_I := (S_I, A_I)$;

$$\Lambda_I \leftarrow \prod_{j=1, j\notin I}^{n} \left(S_j^{1/e_I}\right)^{y_j} = \left(\prod_{j=1, j\notin I}^{n} S_j^{y_j}\right)^{1/e_I} \tag{1}$$

$VC.Verify(crs, C, I, \vec{y}, \pi_I) \rightarrow b \in \{0, 1\}$: the verify algorithm parse $\pi_I := (S_I, A_I)$, and computes for all $i \in I, S_i = S_j^{e_{I\setminus\{i\}}} = U_n^{1/e_I}$. Computing the following equation returns 1 if it holds, and 0 if not.

$$S_I^{e_I} = U_n \wedge C = \Lambda_I^{e_I} \prod_{i\in I} S_i^{y_i} \tag{2}$$

$VC.Disagg(crs, I, \overrightarrow{v_I}, \pi_I, K) \rightarrow \pi_K$: the disaggregation algorithm parse $\pi_I := (S_I, A_I)$, firstly. Then computes S_K from $S_I, S_K \leftarrow S_I^{e_{IK}}$, for each $j \in I \setminus K$, calculate $\quad X_j = S_K^{1/e_j}$, by computing $X_j = S_I^{e_{I(K\cup\{j\})}}$. Finally, it returns the proof $\pi_K := (S_K, \Lambda_K)$, where

$$\Lambda_K \leftarrow \Lambda_I^{e_I \backslash K} \cdot \prod_{i \in I \backslash K} X_j^{v_j} \tag{3}$$

$VC.Agg(crs, (I, \overrightarrow{v}_I, \pi_I), (I, \overrightarrow{v}_J, \pi_J)) \rightarrow \pi_K$: the aggregation algorithm parse $\pi_I := (S_I, \Lambda_I)$, and similarly $\pi_J := (S_J, \Lambda_J)$. let $K = I \cup J$, $I \cap J = \varnothing$. First, compute $S_K \leftarrow ShamirTrick(S_I, S_J)$. Then, for every $j \in J$, computes $\phi_j \leftarrow S_K^{e_{J \backslash \{j\}}} = S_I^{1/e_j}$, and $\psi_i \leftarrow S_K^{e_{I \backslash \{i\}}} = S_J^{1/e_j}$. Then, computes

$$\rho_I \leftarrow \frac{\Lambda_I}{\prod_{j \in J} \phi_j^{v_j}} \tag{4}$$

$$\sigma_J \leftarrow \frac{\Lambda_J}{\prod_{i \in I} \psi_i^{v_i}} \tag{5}$$

Ultimately, return $\pi_K := (S_K, \Lambda_K)$, where $\Lambda_K \leftarrow ShamirTrick(\rho_I, \sigma_J, e_I, e_J)$.

Use the *ShamirTrick* function to compute the roots of a group element g when it has an x-root and a y-root within a group of unknown order and x and y are coprime to each other. Given $\rho_x = g^{\frac{1}{x}}, \quad \rho_y = g^{\frac{1}{y}}, ax + by = 1, a, b,$ and x, y can be computed by using the extended Euclidean algorithm. Afterwards, $g^{\frac{1}{xy}} = g^{\frac{ax+by}{xy}} = g^{\frac{a}{y} + \frac{b}{x}} = \rho_y^a \rho_x^b$. Algorithm 1 is utilized to compute the aggregated auxiliary proof information S_I.

Algorithm 1 Shamir's Trick

```
Require: $\rho_x, \rho_y, x, y$.
Ensure: Accept or reject.
1: if $p_x^x \neq p_y^y$ then return $\perp$
2: end if
3: Through extended Euclidean algorithm to generate $ax + by = d = \gcd(x, y)$ by $a, b, d$ s.t.
4: if $d \neq 1$ then return $\perp$
5: end if
6: return $p_x^x \neq p_y^y$
```

5. Spatio-Temporal Big Data Collaborative Storage Mechanism

This chapter presents our proposed scheme in detail. The specific sections are summarized as follows: Section 5.1 introduces the system model of on-chain and off-chain storage of blockchain; Section 5.2 explains the collaborative storage mechanism of spatio-temporal big data in text form; Section 5.3 describes the specific scheme; Section 5.4 discusses the aggregation proof protocol; and Section 5.5 conducts corresponding security analysis.

5.1. Overview of On-Chain and Off-Chain Collaborative Storage System

The system model diagram of this solution is shown in Figure 1. It is mainly segmented into six layers, including the storage layer, data layer, network layer, consensus layer, contract layer, and application layer. Their specific introduction is as follows:

Storage Layer: Responsible for storing data securely and persistently in the network. This layer is implemented through distributed ledger technology, which encrypts transaction data and stores it decentralizedly on multiple nodes throughout the network. This decentralized storage increases the security and tamper-resistance of the data because to alter the information stored on the blockchain, it is necessary to change, the data on most of the nodes within the network simultaneously, which is almost impractical to achieve in practice. The storage layer is the foundation of the core architecture of blockchain technology, ensuring the tamper-proof and permanently recorded nature of the entire system's data. In addition to the storage of blockchain nodes, this paper also incorporates blockchain off-chain storage into the storage layer to enhance the scalability of the blockchain.

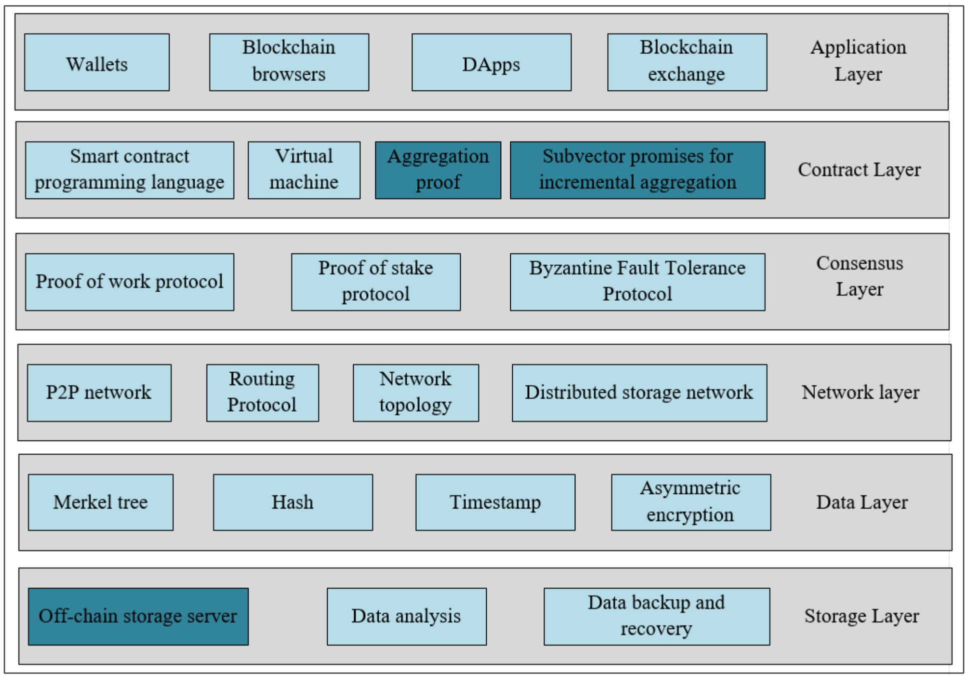

Figure 1. Storage system architecture in on-chain and off-chain system.

Data Layer: The blockchain constitutes a chain-like data structure consisting of a series of connected blocks, each of which internally records the hash value and transaction details of the previous block. The block header carries meta-information such as version numbers, timestamps, random numbers, and the Merkle tree root hash of the transaction. Using the Merkle tree structure, it is possible to efficiently confirm whether a transaction is present within a block.

Network Layer: The network layer is a collection of technologies and protocols that ensure effective communication and data synchronization between blockchain nodes. It covers the discovery and connection mechanisms of nodes, routing protocols, and network topology to ensure that information can be transmitted securely and efficiently in a decentralized environment. Nodes, as the basic units of the network, find optimal paths to exchange blocks and transaction data through specific routing protocols, while the topology of the network ensures the network's resilience and scalability. In addition, the network layer supports the implementation of a distributed storage network, which increases redundancy and improves data security and availability by decentralizing data storage on multiple nodes.

Consensus Layer: Blockchain consensus mechanism is the mechanism tasked with agreeing on a certain version of a transaction record (i.e., the state of the blockchain) among all participants in the network. The consensus layer ensures not only the consistency and immutability of data but also the safety and reliability of the decentralized network. The consensus layer uses different algorithms to achieve this goal, which include proof of work, proof of stake, delegated proof of stake, and Byzantine fault tolerance. Through the consensus mechanism, the blockchain network can ensure that all nodes agree on the correctness and order of data without a central authority.

Contract Layer: The smart contract layer is the programming level that enables automatic enforcement of contract terms on blockchain technology. It is a critical component responsible for handling the creation, deployment, and execution of smart contracts. Smart contracts are automated programs running on the blockchain that can automatically execute predefined operations when predetermined conditions are met.

Application Layer: The application layer of the blockchain provides a direct interface for users to interact with and covers wallets, blockchain browsers, and decentralized applications (DApps). Wallets allow users to manage digital assets, blockchain browsers provide transactions and block data queries, and DApps provide various decentralized services on the blockchain, such as finance and entertainment, through smart contracts, enhancing transparency and security.

5.2. Spatio-Temporal Big Data Collaborative Storage Scheme

The spatio-temporal big data collaborative storage scheme proposed in this paper includes the following phases: Initial setup, Query phase, Verification phase, and Update phase. The work of each phase is described in detail below. The scheme designed in this chapter is shown in Figure 2.

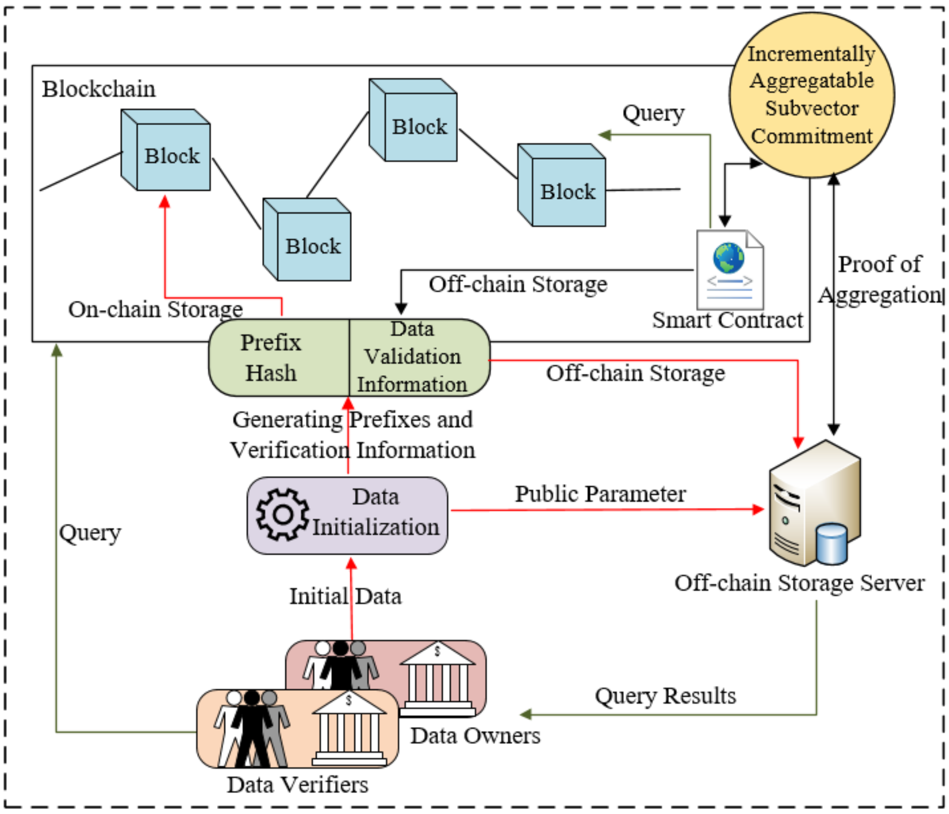

Figure 2. Spatio-temporal big data collaborative storage scheme.

Initial setup: the data owner selects a security parameter, outputs the public parameter, initial summary and initial state through the probabilistic bootstrap algorithm, verifies its legitimacy using a smart contract, and after passing the verification, stores the initial summary, and the initial state in the blockchain and the off-chain storage server, respectively, and sends the initial data together to the off-chain storage server to be stored in the off-chain storage server.

Query phase: the data verifier submits a request to query the subscript collection. The off-chain storage server queries the data in the database according to the requested subscript collection, submits the proofs to the blockchain, performs aggregation using smart contracts, and returns the generated aggregated proofs and stored data to the data verifier after the aggregation is completed.

Verification phase: the data verifier based on the proof and data returned from the off-chain storage server, the initial digest stored in the blockchain. Run the validation algorithm; if it passes the validation, then it indicates that the data is legitimate.

Update phase: the data owner first calculates the corresponding auxiliary information and signature locally based on the data to be updated, and then sends the auxiliary information to the blockchain for upload, the off-chain storage server receives the updated data uploaded by the data owner, the blockchain runs a smart contract to verify the information and the signature, and after the verification passes, the signature is stored in the blockchain, and the off-chain storage server updates the data in the database.

5.3. Scheme Description

In this section, we propose the design of an on-chain and off-chain collaborative storage scheme based on the construction of incrementally aggregatable subvector commitments.

Define the signature $\delta := ((U, C), n), U = g^{\prod_{i\in[n]} e_i}$, progressive construction of U as the value of the RSA accumulator when expanding or reducing the records, $S_I = g^{\prod_{i\in[n]\setminus I} e_i}$ as the witness of accumulator, commitment C, state $st := \pi_I$, where $\pi_I := (S_I, \Lambda_I)$, for each e_i can be generated by the prime generation algorithm, e.g., $e_i \leftarrow PrimeGen(i)$. For all $J \subseteq I$, anyone who has possession of S_I can be able to compute $S_J \leftarrow S_I \prod_{j\in I\cup J} e_j$.

Setup phase: given the security parameter K, create a group with the hidden order $G \leftarrow Ggen(1^\lambda)$ and sample a generator $g \xleftarrow{\$} G$. It defines a deterministic collision-resistant function, $PrimeGen$, which maps integers to prime numbers of $L + 1$ bits. Set original signature $\delta_0 \leftarrow ((1, g), n_0), n_0 \leftarrow 0$, state$st_0 \leftarrow g$ (All algorithms implicitly accept the public parameter as input). Creating Parameters pp and Initial Commitments for Empty records in an Off-Chain Storage Server $C \leftarrow VC.Com(), pp := (G, g, PrimeGen)$, Use this function to compute aggregate auxiliary proof information S_I.

Query phase: the data verifier sends the index set Q of the query location to the off-chain storage server and queries the blockchain to get the signature δ and computes

$$S_Q \leftarrow S_I^{\prod_{i\in I\setminus Q} e_i} \tag{6}$$

If the data set index Q to be queried is a subset of the previously computed proof I in the off-chain storage server, i.e.,, $Q \subseteq I$, the proof Λ_I is disaggregated, $\Lambda_Q \leftarrow VC.Disagg(S_Q, I, V_I, A_I, Q)$. Otherwise, the aggregated proof protocol needs to be executed to generate the location proof Λ_Q and return it to the data verifier.

Verification phase: the data verifier runs the commitment verification algorithm based on obtaining the current latest signature δ_n and proof π_Q from the blockchain, and data V_Q returned from the off-chain storage server:

$$b \leftarrow VC.Ver(pp, C, Q, V_Q, \Lambda_I) \wedge (S_Q^{\prod_{i\in Q} e_i} = U) \tag{7}$$

If $b = 1$, it means that data V_Q are valid.

Update phase: mainly divided into modification, insertion, and deletion operations.

The data owner provides operation $op = (mod, add, del)$ to update data $\Delta = (K, V_K)$. Firstly, it determines whether K is a subset of I. If not, it is necessary to calculate S_K, $e_i \leftarrow PrimeGen(i)$, and $S_K = g\prod_{i\in[n]\setminus K} e_i$, $i \in K$ that need to be uploaded by data initialization, and the blockchain runs a smart contract to verify whether s_K is legal or not, i.e., $b \leftarrow (S_K^{\prod_{i\in K} e_i} = U)$. Then, the data owner transmits the updated data to the off-chain storage server, and updates the database data. There are the following operations depending on the update form:

- Modify:$op = mod, \Delta := (K, V'_K)$

$$C' \leftarrow C \cdot \prod_{i\in K} S_i^{V'_i - V_i} \tag{8}$$

Updating other supporting information $U' \leftarrow U, \Lambda'_I \leftarrow \Lambda_I, n' \leftarrow n, S'_K \leftarrow S_K, \Upsilon_\Delta \leftarrow (V_K, S_K)$.

- Insert:$op = add, \Delta := (K, V'_K)$

If the subscript position of the inserted data does not extend beyond the maximum length of the commitment, n, and the index collection K does not store data, $K \subseteq I$, the proof is aggregated directly using the aggregation algorithm; otherwise, the following calculation is performed:

$$C' \leftarrow C \cdot \prod_{i \in K} S_i^{V_i} \tag{9}$$

$$U' \leftarrow U \Pi_{i \in K} S_i^{V_i} \tag{10}$$

Updating other supporting information $\Lambda'_I \leftarrow \Lambda_I,\ S'_I \leftarrow S_I,\ n' \leftarrow n$.

If the length of the inserted data exceeds the maximum length n of the commitment, i.e., $K \cap I = \varnothing$, then

$$C' \leftarrow C \cdot \prod_{j \in K} S_j^{V_i} \tag{11}$$

$$U' \leftarrow U \Pi_{i \in K} e_i \tag{12}$$

Updating other supporting information $\Lambda'_I \leftarrow \Lambda_I, S'_I \leftarrow S_I, Y_A \leftarrow S_K, n' \leftarrow n + |K|$.

- Delete:$op = del, \Delta := K$

$$C' \leftarrow \frac{C}{\Pi_{j \in K} S_i^{V_i}} \tag{13}$$

$$C' \leftarrow S_I^{\Pi_{i \in I \setminus K} e_i} = S_K \tag{14}$$

$$\Lambda'_I \leftarrow \Lambda_I^{\Pi_{j \in K} e_j} \tag{15}$$

Updating other supporting information $S'_I \leftarrow S_I, \Upsilon_\Delta \leftarrow (V_K, S_K), n' \leftarrow n - |K|$.

Finally updated signature $\delta' = \delta_{n+1} := ((U', C'), n')$.

Execute the smart contract to verify signature δ' to ensure the legitimacy of the signature, and store δ' and auxiliary information on the blockchain chain after passing the verification. Upload the updated proof Υ_Δ and data (Q, V_Q) to the off-chain storage server, and verify the proof $\Lambda_{I \cup Q}$. After passing the verification, save the data into the database.

The off-chain storage server validates $\Upsilon_\Delta := (V_K, S_K)$, e.g.,$b \leftarrow (S_K^{\Pi_{i \in K} e_i} = U)$. Approval of the validation indicates that the off-chain storage server has successfully stored the data.

5.4. Aggregation Proof Protocol

When there are multiple clients initiating queries, the smart contract aggregates the proofs and aggregates them into a short proof. Further, when the data verifier queries again, if the queried set belongs to a subset of the already aggregated proofs, the off-chain database can send the aggregated proofs disaggregated directly to the clients, thus avoiding the need to compute the proofs again.

The aggregation proof protocol is divided into two parts, the on-chain and off-chain executions, the on-chain execution aggregates S_I, S_J into S_K, and the off-chain aggregates Λ_I and Λ_J into Λ_K. The data verifier receives $\pi_K = (S_K, A_K)$ and can verify the data.

Aggregation proof: input signature δ_n, two query outputs (I, V_I, π_I) , (J, V_J, π_J), subscript indexes I and J to be aggregated, aggregated values V_I and V_J, proofs to be aggregated π_I and π_J. The protocol aggregates two proofs into one proof π_K, where $K = I \cup J$. The specific formula is as follows:

$$S_K \leftarrow ShamirTrick(S_I, S_J, \prod_{i \in I} e_i, \prod_{i \in J} e_i) \tag{16}$$

$$A_K \leftarrow VC.Agg((S_I, S_J), (I, V_I, A_I), (I, V_J, A_J)) \tag{17}$$

Finally, return $\pi_K = (S_K, \Lambda_K)$. The algorithm that the aggregation proof protocol executes in the smart contract is as Algorithm 2:

Algorithm 2 Aggregation proof algorithms

```
    Input: auxiliary verification information $S_I, S_J$, index set $I, J$, $\{e_i\}_{i \in I \cup J}$, proof $V_I, V_J$
    Output: auxiliary verification information $S_{I \cup J}$, aggregation proof $A_{I \cup J}$
 1: $product_1 \leftarrow 1$; $product_2 \leftarrow 1$
 2: for each element $i$ in $I$ do
 3:     $product_1 \leftarrow product_1 \cdot e_i$
 4: end for
 5: for each element $j$ in $J$ do
 6:     $product_2 \leftarrow product_2 \cdot j$
 7: end for
 8: $S_k \leftarrow \text{ShamirTrick}(S_I, S_J, product_1, product_2)$
 9: for each element $j$ in $J$ do
10:     $\phi_j \leftarrow S_j^{-1/e_j}$
11: end for
12: for each element $i$ in $I$ do
13:     $\psi_i \leftarrow S_i^{1/e_i}$
14: end for
15: $S_k \leftarrow \text{ShamirTrick}(\frac{\Lambda_I}{\Pi_{j \in J} \phi_j^{v_j}}, \frac{\Lambda_J}{\Pi_{i \in I} \psi_i^{v_i}}, product_1, product_2)$;
16: return $S_K$
```

5.5. Security Analysis

We assess the correctness and security of the proposed scheme under the conditions that the protocol presupposes that the processes are consistent, the sender is correct, and the network is in a stable "good" state. Specifically, we explore the security performance of the whole scheme without any failures. We outline the security parameters of the proposed scheme. Intuitively, we necessitate that in a single query by a data verifier, a malicious off-chain storage server is unable to convince the data verifier that the forged data is true. To establish this, we permit the attacker A freely choose the system's history. From a starting empty state, progressively construct the history records, each consisting of updates (additions, deletions, modifications). The data verifier queries the summary δ generated based on the history and the suggestions of the adversary A while getting the corresponding data results.

Theorem 1 (correctness). *If the data owner and the off-chain storage server honestly execute our proposed scheme algorithm and aggregate proof protocol, the verification result obtained by the data verifier will always be a correct value.*

Proof.

$$\begin{aligned}
\Lambda_K &= \Lambda_I^{e_{INK}} \cdot \prod_{j \in I \setminus K} \chi_j^{v_j} \\
&= \left(\prod_{j=1, j \notin I}^{n} S_j^{v_j} \right)^{\frac{1}{e_I} e_{I \cap K}} \cdot \prod_{j \in I \wedge K} \left(S_j^{\frac{1}{e_K}} \right)^{v_j} \\
&= \left(\prod_{j=1, j \notin I}^{n} S_j^{v_j} \right)^{\frac{1}{e_K}} \cdot \left(\prod_{j \in I \setminus K} S_j^{v_j} \right)^{1/e_K} \\
&= \left(\prod_{j=1, j \notin K}^{n} S_j^{v_j} \right)^{1/e_K}
\end{aligned} \tag{18}$$

ρ_I and σ_J can be calculated as follows:

$$\rho_I := \frac{\Lambda_I}{\Pi_{j \in J} \phi_j^{v_j}} = \left(\prod_{j=1, j \notin I \cup J}^{n} S_j^{v_j} \right)^{1/e_I} \tag{19}$$

$$\sigma_J := \frac{\Lambda_I}{\prod_{j\in I}\psi_j^{v_j}} = \left(\prod_{j=1,j\notin I\cup J}^{n} S_j^{v_j}\right)^{1/e_J} \tag{20}$$

$$\begin{aligned}\Lambda_K &:= ShamirTrick(\rho_I, \sigma_J, e_I, e_J)\\ &= ShamirTrick\left(\left(\prod_{j=1,j\notin I\cup J}^{n} S_j^{v_j}\right)^{1/e_I}, \left(\prod_{j=1,j\notin I\cup J}^{n} S_j^{v_j}\right)^{1/e_J}, e_I, e_J\right)\\ &= \left(\prod_{j=1,j\notin I\cup J}^{n} S_j^{v_j}\right)^{\frac{1}{e_I e_J}}\\ &= \left(\prod_{j=1,j\notin I\cup J}^{n} S_j^{v_j}\right)^{\frac{1}{e_{I\cup J}}}\end{aligned} \tag{21}$$

□

Theorem 2. *A malicious off-chain storage server that fails to convince the verifier to accept invalid outputs proves that the scheme is secure, i.e., the attacker succeeds if it persuades the verification algorithm to validate incorrect output values based on the input values. For any probability polynomial time adversary A has*

$$Pr[Exp_A(k)[DB] = 1] \leq negl(k) \tag{22}$$

where negl refers to the negligible function with respect to k. Specifically, A has advantages in the following Algorithm 3:

Algorithm 3 Experiment

```
    Input: security parameter $k$
    Output: accept or reject
 1: $(pp, \delta_0, st_0) \leftarrow \mathrm{Setup}(1^k)$
 2: $((op^i, \Delta^i, \Upsilon^i_\Delta), Q, V_Q^*, \pi^*) \leftarrow A(pp, \delta_0, st_0)$
 3: for $i \in [l]$ do
 4:     if $op^i \in \{\mathrm{mod}, \mathrm{add}\}$ then parse $\Delta^i = (K, V_k')$
 5:         $\forall i \in K{:}V_i^* \leftarrow V_i'; \forall i \in [|V|]\backslash K{:}V_i^* \leftarrow V_i;$
 6:     else if $op^i = \mathrm{del}$ then parse $\Delta^i = K$
 7:         $\forall i \in K{:}V_i^* \leftarrow V_i;$
 8:         run the update algorithm: input $(\delta_{i-1}, op^i, \Delta^i, \Upsilon^i_\Delta)$, output$(b_i, \delta_i)$;
 9:     end if
10:     $b \leftarrow \bigwedge b_i$
11: end for
12: $b \leftarrow b \wedge V_i^* \neq V_i \wedge VC.Ver(pp, C, Q, V_Q, \Lambda_I);$
13: return $b$
```

Proof. This approach can be regarded as sustaining and revising a vector commitment C and an RSA accumulator U. U represents the RSA accumulator for all e_i and serves to authenticate S_I. For every accumulator to maintain its integrity, the calculated cumulative value U must be derived truthfully. This is ensured in this scheme as we assume the existence of a valid history. It is possible to know through historical data whether U is of the correct form ($U = g^{\prod_{i\in[n]} e_i}$), and consequently it can be safely checked S_I is in the proper form (by checking $S_I^{\prod_{i\in I} e_i} = U$), this can be guaranteed by the security of the RSA accumulator. Upon verifying the validity of S_I, it reduces to a positional binding issue for vector promises. Therefore, we give the theorem and proof of position binding of vector commitment below. □

Theorem 3. *Consider Ggen as the generator of the hidden order group, where the low-order assumption is valid. Consequently, the previously defined SVC scheme possesses the location binding property.*

Proof. We begin by defining the game S_0 as the genuine position-binding game stated in the theorem, with the goal of proving such that for any probabilistic polynomial-time algorithm A, $Pr[\text{Game}_0 = 1] = \text{negl}(\lambda)$.

```
Game Game_0
1: $crs \leftarrow \text{VC.Setup}(1^n, m)$
2: $(C, I, \pi, \tau, \pi^*) \leftarrow A(crs)$
3: $b \leftarrow VC.Ver(crs, C, I, \vec{y}, \pi) = 1 \wedge \vec{y} \neq \vec{y}^* \wedge VC.Ver(crs, C, I, \vec{y}^*, \pi^*) = 1;$
4: return $b$
```

Now, let the game $Game_1$ be the same as above, except the opponent outputs S_I and S_I' making $S_I = g^{e_{[n]}} I = S_I^*$.

```
Game $Game_1$
1: $crs \leftarrow VC.Setup(1^\lambda, 1, n);$
2: $(C, I, \vec{y}, (S_I, A_I), \vec{y}^*, (S_I^*, A_I^*)) \leftarrow \mathcal{A}(crs);$
3: if $S_I \neq g^{e_{[n] \vee I}}$ or $S_I^* \neq g^{e_{[n] \vee I}}$ then abort
4: $b \leftarrow C = \Lambda_I^{e_I} \prod_{i \in I} \left( S_I^{e_{I \setminus \{i\}}} \right)^{y_i} \wedge \vec{y} = \vec{y}^* \wedge C = \Lambda_I^{*e_I} \prod_{i \in I} \left( S_I^{*e_{I \setminus \{i\}}} \right)^{\vec{y}^*};$
5: return $b$
```

Then, $Pr[\text{G}_0 = 1] < Pr[\text{G}_1 = 1] + \text{negl}(\lambda)$. In $Game_0$, $\text{S}_\text{I}^{\text{e}_\text{I}} = \text{U}_\text{n} = \text{g}^{\text{e}_{[\text{n}]}}$. Assume $\text{S}_\text{I} \neq \text{g}^{\text{e}_{[\text{n}] \vee \text{I}}}$ then $\text{g}^{\text{e}_{[\text{n}] \setminus \text{I}}} = \text{S}_\text{I}^*$; therefore, $\text{S}_\text{I}^{\text{e}_\text{I}} = \text{S}_\text{I}^{*\text{e}_\text{I}}$, $\left(\text{S}_\text{I}^{-1}\text{S}_\text{I}^*\right)^{\text{e}_\text{I}} = 1$. Since S_I^* can be computed efficiently, and $\text{e}_\text{I} < 2^{\text{poly}(\lambda)}$ holds. This forms a low-order solution to the hidden order group Solution to the problem. Under low-level assumptions, it only occurs with negligible probability, and the same is true for S_I^*.

Let game $Game_2$ be the same as above, except each opponent needs to receive $e_i \leftarrow PrimeGen(i)$ and $S_i = g^{e_{[n] \setminus \{i\}}}$, where $i \in [n]$.

```
Game $Game_2$
1: $(G, g, \text{PrimeGen}) \leftarrow VC.Setup(1^\lambda, 1, n);$
2: $e_i \leftarrow PrimeGen(i);$
3: for each $i$ in $[n]$ do
4:     $S_I = g^{\prod_{i \in [n] \setminus \{i\}} e_i};$
5: end for
6: $(C, I, \vec{y}, A_I, \vec{y}^*, A_I^*) \leftarrow \mathcal{A}(G, g, PrimeGen, \{S_i\}_{i \in [n]});$
7: $b \leftarrow C = \Lambda_I^{e_I} \prod_{i \in I} S_i^{\nu_i} \wedge \vec{y} = \vec{y}^* \wedge C = \Lambda_I^{*e_I} \prod_{i \in I} S_i^{\vec{y}^*};$
8: return $b$
```

It can be directly concluded that $Pr[\text{Game}_1 = 1] = Pr[\text{Game}_2 = 1]$, and $Game_2$ Same as the position-bound game of the SVC scheme, based on the assumption $Pr[\text{Game}_2 = 1] = \text{negl}(\lambda)$.

Therefore, we conclude that the previously defined SVC scheme possesses the location binding property. □

6. Performance of Our Scheme

In this scheme, the experimental analysis is divided into off-chain and on-chain parts based on whether the operation is executed on the blockchain. Firstly, we tested the communication overhead of the system, focusing mainly on the setup phase. Then, we evaluated the time cost of off-chain operations to assess the computational overhead of our protocol off-chain.

We tested the time costs for querying, verification, and validation of the off-chain storage server with different numbers of data blocks. On-chain, we primarily conducted simulation analysis on the efficiency of smart contract execution. The proposed scheme was implemented using the pairing-based cryptography library (JPBC) and the off-chain storage database was deployed using MongoDB. The smart contract was deployed in the JavaScript VM environment of the Remix compiler. The configuration of the devices used in the experiments is shown in Table 1.

Table 1. Parameter configuration table.

Parameter	Storage Server	Data Owner/Verifier
CPU	3.1 GHz	2.8 GHz
Memory	16 GB	8 GB
Hard Disk	1 TB	128 GB
Network Speed	300 Mb	100 Mb

6.1. Communication Overhead

In the collaborative on-chain and off-chain storage scheme, during the setup phase, data owners need to send initial setup information to the off-chain storage server and the blockchain. This information is used later for querying, verification, and updating of the data. The experiment in this chapter compares the scheme proposed in this paper with the schemes referred to as Ref [34] and the Bsmd [35] scheme.

In the Ref scheme, the user divides the file into blocks to generate copies and data tags and sends them to the cloud organizer (CO) for verification. After passing, the CO distributes the data and tags it to the cloud server (CS) and records the storage location. The Bsmd scheme is similar to the system storage model of this paper. However, in the setup phase, the data owner generates signatures based on the original data and stores these signatures on the chain after verification by other nodes, while storing the complete data off the chain. Both the Ref and Bsmd schemes need to calculate and submit the corresponding location information for each submitted data, and its size is linear and related to the amount of data submitted. In the scheme proposed in this chapter, it is necessary to compute and submit information about the position of each piece of data submitted, with the size being linear and related to the volume of data submitted. In the scheme proposed in this chapter, we amortize the verification parameters over each upload and update, only calculating and sending $(G, g, PrimeGen)$ during the setup phase, which has a fixed size and is independent of the volume of data submitted. We experimentally compared the parameter overhead during the setup phase of these three schemes, with the results shown in Figure 3. It can be concluded that the scheme proposed in this chapter has a higher communication efficiency and can enhance system performance.

6.2. Off-Chain Overhead

In this section, to better test the aggregation performance of the proposed scheme, we divided the queries into two types: Query1 and Query2. Both are batch queries that require the off-chain storage server to run either a de-aggregation or aggregation algorithm. The distinction lies in that for Query1, when the query location $Q \cap I = \varnothing$, it necessitates the aggregation of S_i and Λ_i. In another scenario, Query2, when the query location $Q \cap I = \varnothing$, a de-aggregation algorithm is needed, where I represents the proofs that have already been aggregated in the database. Since in practical applications, data owners need to upload S_i at the time of data upload, we will pre-calculate the corresponding S_i at the respective positions. We created a file of 20 MB and partitioned it into 10,000 data blocks, each block being 2 Kb in dimensions.

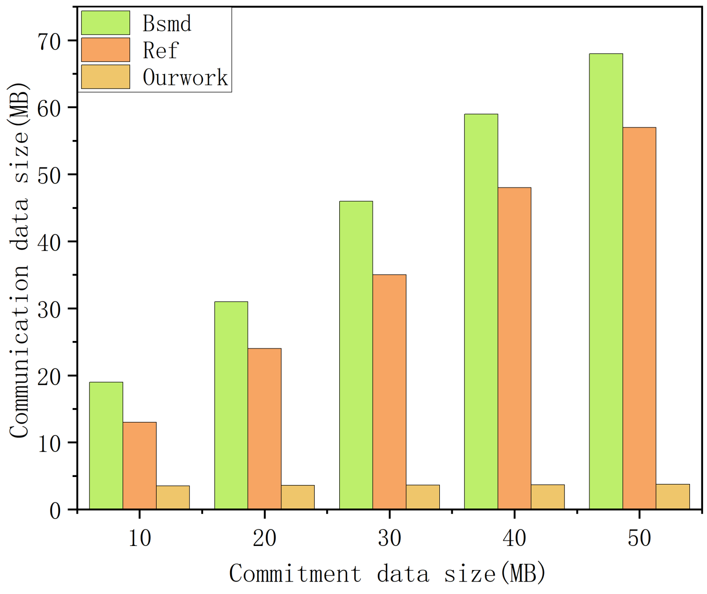

Figure 3. Comparison result of communication overhead of different schemes.

As shown in Figure 4, specifically because we have implemented preprocessing techniques, the time required for our verification operation, Verify, is the lowest among the three operations and does not vary significantly with various quantities of data blocks. This indicates that the verification process is both fast and stable. The time expense of the Query1 operation increases with the number of data blocks, but the overall increase is moderate, showing a certain linear growth trend. The time expense of the Query2 operation also increases with the quantity of data blocks, and it has the highest time expense among all operations. Particularly when the quantity of data blocks is large, its time cost is significantly higher than that of Query1 and the verification operation.

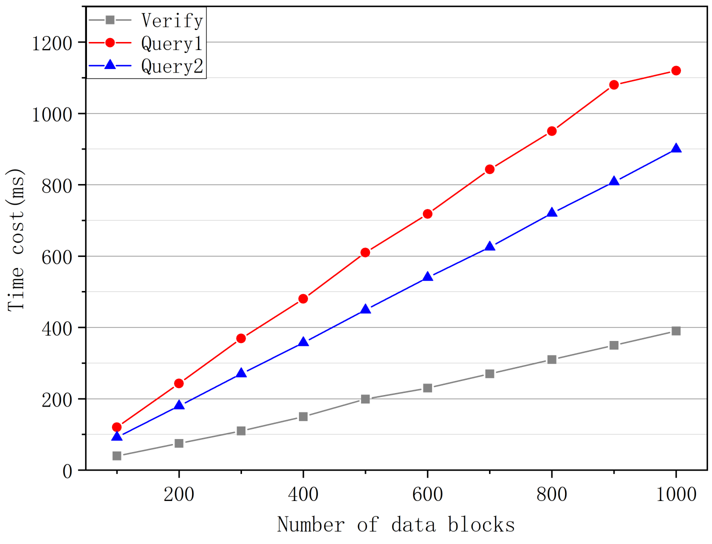

Figure 4. Comparison chart of verification and query experimental results.

At all data block sizes, the time required for Query1 is noticeably higher than that for Query2 and the verification operation. Especially when there are many data blocks, the time required is far higher than the other two operations. This indicates that when the query location $Q \cap I = \varnothing$, when the positions involved in the query need to run the aggregation algorithm, the processing time significantly increases, suggesting that the aggregation algorithm may be more computationally complex. Therefore, we divide the aggregation algorithm into two parts: one that runs on-chain and another that runs off-chain.

In the experimental analysis of data updates within the scheme, we categorized updates into modifications, insertions, and deletions. As shown in Figure 5, modifications do not require updates to the accumulator, thereby saving a significant amount of time, making the cost of modification operations the lowest. However, insertions and deletions involve updates to the accumulator, leading to relatively higher costs. Additionally, since deletion operations require updates to the proofs, they incur the highest costs, followed by insertions. This distinction in cost reflects the computational overhead associated with managing the completeness and uniformity of the data when the underlying structure of the dataset changes.

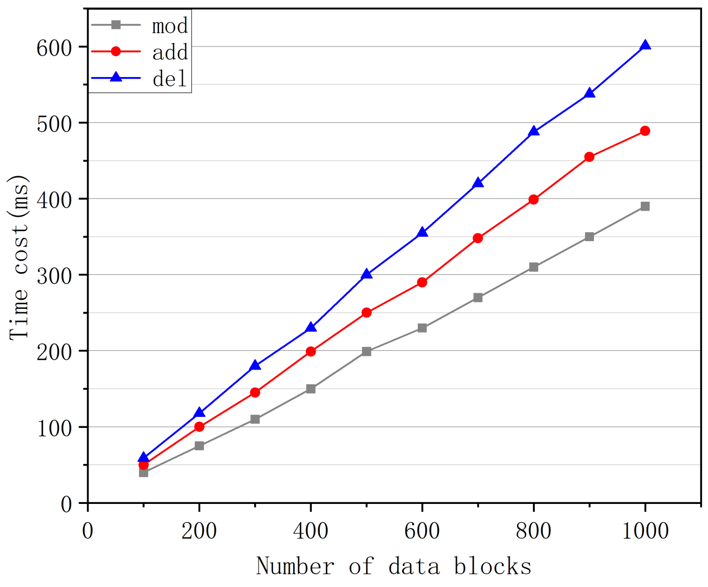

Figure 5. Comparison chart of updated experimental results.

6.3. On-Chain Overhead

In the scheme proposed in this chapter, the on-chain operations within the blockchain primarily involve signature verification and the execution of parts of the aggregation algorithm by smart contracts.

The smart contracts mainly execute a portion of the aggregation algorithm proposed in this scheme, specifically utilizing the ShamirTrick to generate auxiliary proof information. To simulate the conditions of a real blockchain network for testing and developing smart contracts, the experiments deploy the smart contracts using the JavaScript VM environment in the Remix compiler. The deployment time for smart contracts using Remix is approximately 2 seconds. This setup allows for an efficient and realistic testing environment, aiding in the optimization of the smart contracts before they are deployed on an actual blockchain network.

Given the limited count of nodes in the test network, the mining difficulty is relatively low, which allows for faster block generation speeds. However, in real-world application scenarios, as the count of nodes participating in the network rises, maintaining a low mining difficulty could lead to the production of too many blocks at the same time, potentially causing congestion in network transactions. Therefore, the average block time on the Ethereum mainnet is currently stabilized at around 12 seconds.

To comprehensively evaluate the performance of the smart contracts in this setup, we used the Caliper tool to conduct detailed tests on the verification and aggregation functions within the smart contracts. Each function was independently tested 400 times to accurately measure various performance metrics, including sending rate, maximum delay, minimum delay, average delay, and throughput, as shown in Table 2. These tests confirmed that both verification and aggregation operations can be executed successfully, with verification operations being more efficient than aggregation operations. This efficiency difference is crucial for optimizing blockchain performance, especially in environments where rapid transaction processing is required. Improving the efficiency of verification operations can significantly enhance system responsiveness, while optimizing aggregation operations can further boost the overall performance of the system in handling complex data tasks.

Table 2. Execution efficiency of verification and aggregation.

Operation	Sending Rate/TPS	Max Delay/s	Min Delay/s	Avg Delay/s	Throughput/TPS
Verification	376.58	24.99	10.9	13.11	17.06
Verification	379.43	24.94	11.09	13.04	17.07
Verification	383.42	24.98	11.01	12.87	17.01
Aggregation	302.18	45.38	19.12	42.28	10.98
Aggregation	299.84	44.70	18.94	42.10	11.07
Aggregation	300.36	44.56	18.98	42.33	10.97

7. Conclusions

The proposed on-chain and off-chain collaborative storage system model architecture for processing large-scale spatio-temporal big data has important research significance. First, by utilizing incrementally aggregatable subvector commitments built on RSA, an efficient on-chain and off-chain collaborative storage scheme is implemented. This not only improves the communication efficiency of the system but also significantly reduces the computational overhead in the setup phase, providing a practical solution for the secure storage of large-scale spatio-temporal data. Secondly, the aggregation proof protocol designed in this study allows query set-based aggregation or decomposition proofs in the verification phase and is integrated with smart contracts. This design not only enhances the system's processing efficiency for update and verification transactions but also improves the overall operational performance and response speed of the system. In practical applications, these innovations not only help manage and protect large amounts of spatio-temporal big data containing time and geographic information tags, but also ensure the consistency and integrity of data storage. Future work prospects include further optimization of algorithms and protocols to improve the scalability and adaptability of the system, especially in the face of growing data size and complexity. In addition, more in-depth security analysis can be explored, including further research and testing of attack models to ensure the robustness and security of the system in the face of various threats. In summary, the research work proposed in this paper provides a solid theoretical foundation and practical application value for solving the security and efficiency challenges of large-scale spatio-temporal big data storage and, at the same time, lays the foundation for in-depth exploration and innovation in related fields in the future.

Author Contributions: Conceptualization, Mingjia Han and Ding Huang; methodology, Xinyi Yang; validation, Ding Huang; formal analysis, Huachang Su and Yekang Zhao; writing—original draft preparation, Mingjia Han and Xinyi Yang; writing—review and editing, Yekang Zhao and Yongjun Ren; supervision, Yongjun Ren; project administration, Ding Huang and Xinyi Yang. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (Grant No. 62072249).

Data Availability Statement: No new data were created nor analyzed in this study. Data sharing is not applicable to this article.

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

References

1. Teng, Z.; Yu, Y.; Zhou, H. Analysis of Data Security Technology Based on Cloud Storage. *Netw. Secur. Technol. Appl.* **2023**, *2023*, 61–62.
2. Guo, Y. Analysis of Network Information Security Issues in Big Data Environment. *Netw. Secur. Technol. Appl.* **2022**, *2022*, 58–59.
3. Niu, S. A Review of Cryptographic Technologies Based on Cloud Computing. *Electron. Technol. Softw. Eng.* **2021**, *2021*, 227–230.
4. Yang, C.; Chen, X.; Xiang, Y. Blockchain-based publicly verifiable data deletion scheme for cloud storage. *J. Netw. Comput. Appl.* **2018**, *103*, 185–193. [CrossRef]
5. Zhang, P.; Cheng, X.; Su, S.; Wang, N. Task allocation under geo-indistinguishability via group-based noise addition. *IEEE Trans. Big Data* **2022**, *9*, 860–877. [CrossRef]
6. Nakamoto, S.; Bitcoin, A. A peer-to-peer electronic cash system. *Bitcoin* **2008**, *4*, 15. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 6 August 2024).
7. Feng, J.; Yang, L.T.; Ren, B.; Zou, D.; Dong, M.; Zhang, S. Tensor recurrent neural network with differential privacy. *IEEE Trans. Comput.* **2023**, *73*, 683–693. [CrossRef]
8. Zeng, S.; Huo, R.; Huang, T. A Comprehensive Review of Blockchain Technology: Principles, Progress, and Applications. *J. Commun.* **2020**, *41*, 134–151.
9. Zhang, X.; Niu, B.; Gong, T. Scalable Storage Model for Account-based Blockchains. *J. Beijing Univ. Aeronaut. Astronaut.* **2022**, *48*, 708–715.
10. Ren, Y.; Lv, Z.; Xiong, N.N.; Wang, J. HCNCT: A cross-chain interaction scheme for the blockchain-based metaverse. *ACM Trans. Multimed. Comput. Commun. Appl.* **2024**, *20*, 1–23. [CrossRef]
11. Chen, J.; Yang, H.; He, K.; Li, K.; Jia, M.; Du, R. Current Status and Prospects of Blockchain Scalability Technologies. *J. Softw.* **2024**, *35*, 828–851.
12. Sun, Z.; Zhang, X.; Xiang, F.; Chen, L. Research Progress on the Scalability of Blockchain Storage. *J. Softw.* **2021**, *32*, 1–20.
13. Luo, C.; Jin, X.; Zhang, Y.; Cai, H.; Liu, Y.; Jing, X.; Huang, G. Data and Location Decoupling Algorithm for Heterogeneous DHT Storage. *J. Softw.* **2023**, *34*, 4930–4940.
14. Wu, T.; Jourjon, G.; Thilakarathna, K.; Yeoh, P.L. MapChain-D: A distributed blockchain for IIoT data storage and communications. *IEEE Trans. Ind. Inform.* **2023**, *19*, 9766–9776. [CrossRef]
15. Hassanzadeh-Nazarabadi, Y.; Küpçü, A.; Özkasap, Ö. LightChain: Scalable DHT-based blockchain. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *32*, 2582–2593. [CrossRef]
16. Li, Y.; Yu, Y.; Wang, X. Three-tier storage framework based on TBchain and IPFS for protecting IoT security and privacy. *ACM Trans. Internet Technol.* **2023**, *23*, 1–28. [CrossRef]
17. Mahmud, M.; Sohan, M.S.H.; Reno, S.; Sikder, M.B.; Hossain, F.S. Advancements in scalability of blockchain infrastructure through IPFS and dual blockchain methodology. *J. Supercomput.* **2024**, *80*, 8383–8405. [CrossRef]
18. Feng, J.; Yang, L.T.; Zhu, Q.; Choo, K.K.R. Privacy-preserving tensor decomposition over encrypted data in a federated cloud environment. *IEEE Trans. Dependable Secur. Comput.* **2018**, *17*, 857–868. [CrossRef]
19. Ren, Y.; Leng, Y.; Qi, J.; Sharma, P.K.; Wang, J.; Almakhadmeh, Z.; Tolba, A. Multiple cloud storage mechanism based on blockchain in smart homes. *Future Gener. Comput. Syst.* **2021**, *115*, 304–313. [CrossRef]
20. Feng, T.; Kong, F.; Liu, C.; Ma, R.; Albettar, M. Blockchain-Based Dual Verifiable Cloud Storage Scheme. *J. Commun.* **2021**, *42*, 192–201.
21. Zhou, Y.; Chen, L. Blockchain-Based Fine-Grained Cloud Data Security Storage and Deletion Scheme. *J. Electron. Inf. Technol.* **2021**, *42*, 1856–1863.
22. Zhang, C.; Xu, Y.; Hu, Y.; Wu, J.; Ren, J.; Zhang, Y. A blockchain-based multi-cloud storage data auditing scheme to locate faults. *IEEE Trans. Cloud Comput.* **2021**, *10*, 2252–2263. [CrossRef]
23. Sun, L.; Wang, Y.; Ren, Y.; Xia, F. Path signature-based xai-enabled network time series classification. *Sci. China Inf. Sci.* **2024**, 1–15. [CrossRef]
24. Wu, K.; Ma, Y.; Cai, H.; Jing, X.; Huang, G. BETASCO: A Consortium Blockchain System for Smart Contract Sharding. *J. Softw.* **2023**, *34*, 5042–5057.
25. Zheng, P.; Xu, Q.; Zheng, Z.; Zhou, Z.; Yan, Y.; Zhang, H. Meepo: Sharded consortium blockchain. In Proceedings of the 2021 IEEE 37th International Conference on Data Engineering (ICDE), Chania, Greece, 19–22 April 2021; pp. 1847–1852.
26. Ren, Y.; Leng, Y.; Cheng, Y.; Wang, J. Secure data storage based on blockchain and coding in edge computing. *Math. Biosci. Eng* **2019**, *16*, 1874–1892. [CrossRef] [PubMed]
27. Fan, Y.; Sheng, D.; Wang, L. Blockchain Storage Optimization Based on Erasure Coding. *Chin. J. Comput.* **2022**, *45*, 858–876.
28. Lajam, O.; Mohammed, S. Optimizing efficiency of P2P content distribution with network coding: Principles, challenges, and future directions. *J. Netw. Comput. Appl.* **2024**, *223*, 103825. [CrossRef]
29. Li, M.; Qin, Y.; Liu, B.; Chu, X. Enhancing the efficiency and scalability of blockchain through probabilistic verification and clustering. *Inf. Process. Manag.* **2021**, *58*, 102650. [CrossRef]
30. Mamun, A.A.; Yan, F.; Zhao, D. BAASH: Lightweight, efficient, and reliable blockchain-as-a-service for hpc systems. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, St. Louis, MI. USA, 14–19 November 2021; pp. 1–18.

31. Zhang, P.; Cheng, X.; Su, S.; Wang, N. Effective truth discovery under local differential privacy by leveraging noise-aware probabilistic estimation and fusion. *Knowl.-Based Syst.* **2023**, *261*, 110213. [CrossRef]
32. Wu, H.; Ashikhmin, A.; Wang, X.; Li, C.; Yang, S.; Zhang, L. Distributed error correction coding scheme for low storage blockchain systems. *IEEE Internet Things J.* **2020**, *7*, 7054–7071. [CrossRef]
33. Thai, Q.T.; Ko, N.; Byun, S.H.; Kim, S.M. Design and implementation of NDN-based Ethereum blockchain. *J. Netw. Comput. Appl.* **2022**, *200*, 103329. [CrossRef]
34. Yang, X.; Pei, X.; Wang, M.; Li, T.; Wang, C. Multi-Replica and Multi-Cloud Data Public Audit Scheme Based on Blockchain. *IEEE Access* **2020**, *8*, 144809–144822. [CrossRef]
35. Ren, Y.; Huang, D.; Wang, W.; Yu, X. BSMD: A blockchain-based secure storage mechanism for big spatio-temporal data. *Future Gener. Comput. Syst.* **2023**, *138*, 328–338. [CrossRef]