Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.067968

ARTICLE

Research on Efficient Storage Consistency Verification Technology for On-Chain and Off-Chain Data

Wei Lin and Yi Sun*

School of Cryptographic Engineering, Information Engineering University, Zhengzhou, 45000, China

*Corresponding Author: Yi Sun. Email: 11112072@bjtu.edu.cn

Received: 17 May 2025; Accepted: 04 August 2025; Published: 23 October 2025

ABSTRACT: To enable efficient sharing of unbounded streaming data, this paper introduces blockchain technology into traditional cloud data, proposing a hybrid on-chain/off-chain storage model. We design a real-time verifiable data structure that is more suitable for streaming data to achieve efficient real-time verifiability for streaming data. Based on the notch gate hash function and vector commitment, an adaptive notch gate hash tree structure is constructed, and an efficient real-time verifiable data structure for on-chain and off-chain stream data is proposed. The structure binds dynamic root nodes sequentially to ordered leaf nodes in its child nodes. Only the vector commitment of the dynamic root node is stored on the chain, and the complete data structure is stored off-chain. This structure ensures tamper-proofing against malicious off-chain cloud servers of off-chain cloud servers. Preserves storage scalability space, realizes the immediate verification of stream data upon arrival, and the computational overhead of on-chain and off-chain hybrid storage verification is only related to the current data volume, which is more practical when dealing with stream data with unpredictable data volume. We formalize this as an efficient real-time verification scheme for stream data in on-chain and off-chain hybrid storage. Finally, the technology's security and performance were empirically validated through rigorous analysis.

KEYWORDS: On-chain and off-chain hybrid storage; verifiable technology; slot gate hash function; vector commitment

1 Introduction

The era of digital economy has driven exponential data growth of data volume. The rapid development of large-scale complex networks such as P2P networks, cloud computing, and sensor networks has also posed new challenges to the resource management of data. To ensure secure data outsourcing, terminal devices will entrust massive data resources to third parties, such as cloud servers. This way of moving data to the cloud enables more efficient allocation of data resources and is thus ubiquitous in enterprise data management. It is estimated that the global data sharing market will exceed one trillion US dollars in scale by 2026. However, due to the openness of the network, in an untrusted environment, data on the cloud faces security and privacy threats such as accidental damage and malicious tampering. Verifiable computation technologies cryptographically ensure for verifying data integrity and credibility during data sharing, which can provide an important guarantee for the secure sharing of data on the cloud. This technology is constantly improving in functionality and continuously enhancing in performance, providing a more reliable solution for cloud data security verification. However, due to the openness and untrustworthiness of cloud servers, there is still a threat of data being tampered with. With the development of smart contract functions, blockchain [1] has

been applied as a database to ensure that data is not tampered with. Therefore, blockchain applications have gradually transformed from a distributed ledger for simple transaction records to an effective and reliable data trusted storage carrier. The service process is calculated by the smart contract, which can ensure the fairness of the transaction results, bring a new solution for data security sharing on the cloud, and a new paradigm for hybrid storage on and off the chain emerges.

As shown in Fig. 1, the data to be processed is not only static and of known size, but also unbounded and fast streaming data (for example, a large amount of streaming data is generated in scenarios such as surveillance video, medical data, and industrial control). Since the data source, as the data owner, can only see part of the data in the current buffer and cannot predict the future data, the generation speed and scale of stream data are unpredictable, and the scale of its verifiable data structure cannot be determined in advance either. Moreover, the stream data is continuous and has no upper limit. When sharing the stream data, the data source no longer holds control over the stream data. Therefore, it is necessary to consider how to conduct real-time integrity verification on the continuously growing stream data.

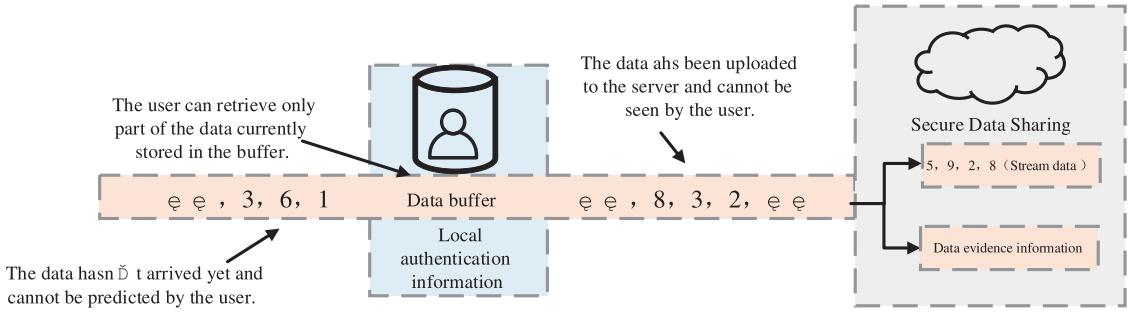

Figure 1: Schematic diagram of stream data sharing

On the blockchain with limited scalability, it is obviously impossible to maintain an entire frequently updated verifiable data structure. Unbounded stream data requires more frequent updates of the verifiable data structure. Therefore, dedicated efficient real-time verifiable data structures need to be designed for stream data. To find a more suitable efficient real-time verifiable data structure for streaming data, this chapter designs an efficient real-time verifiable technology for streaming data for on-chain and off-chain hybrid storage, achieving low maintenance costs on the blockchain, real-time construction along with the dynamic growth and changes of streaming data, and realizing the immediate verification of streaming data upon arrival. The main research work is as follows:

(1) Aiming at the problems of cloud server forging and tampering data, construct an adaptive trapdoor hash tree structure based on the trapdoor hash function and vector commitment, sequentially bind the dynamic root node and the ordered leaf nodes under the root node, do not need to pre-set the scale of data security sharing, and carry out real-time construction with the dynamic growth and change of the big data streams, so as to realise the immediate validation of the secure sharing of streaming data.

(2) Aiming at the problem that limited scalable blockchain cannot be applied to large-scale new stream data, we propose an efficient real-time verifiable data structure for stream data with mixed on-chain and off-chain storage, where only the vector commitment of the dynamic root node is stored on the chain, and the complete data structure is stored under the chain, which effectively prevents tampering by the off-chain cloud server and ensures sufficient storage space, and the efficient verifiability of the stream data is now achieved.

(3) Based on the above research, an efficient real-time verifiable scheme for streaming data with on-chain-off-chain hybrid storage is designed, which proves the security and efficiency of the technique through

security analysis and performance analysis, and proves that the computational overhead of on-chain-off-chain hybrid storage verification is only related to the current data volume, which is more practical when dealing with streaming data with unpredictable data volume.

As a result, this paper designs an efficient real-time verifiable data structure for streaming data under the hybrid on-chain-off-chain storage mode to ensure the integrity of the safe sharing of streaming data in hybrid on-chain-off-chain storage in view of the characteristics of continuity and unpredictability of streaming data.

2 Research Status

Due to the problems such as unpredictable scale, high verification real-time performance and delay sensitivity in the sharing of streaming data, the integrity verification of streaming data is different from the traditional verifiability of data. The design of its verifiable data structure should meet the requirements of efficient and real-time verification of streaming data. To be applicable to infinitely continuous stream data, Papamanthou et al. [2] proposed a non-interactive verifiable data structure for stream data based on the ideas of Mermerkel hash tree and cipher accumulation, called generalized Hash tree GHT. This scheme supports the complexity of dynamic data operations and can complete data updates without interaction. However, this scheme is overly complex. It is difficult to design and implement in practical applications. Yu et al. [3] proposed the fully homomorphic encrypted Merkle Hash Tree structure FHMT based on the Merkle Hash tree structure (MHT), and based on the homomorphic encryption algorithm to achieve verifiable secure sharing of stream data. Because the leaf nodes to the root node in the MHT structure are unidirectional, it is necessary to determine the information of all leaf nodes in advance. In view of the unpredictable data characteristics of streaming data, the MHT structure obviously needs further improvement. Meanwhile, since the stream data of the data source is cached to construct the MHT structure, a large amount of storage overhead of the stream data will also be wasted. The FHMT structure has less overhead at the data source end and is more suitable for resource-constrained terminal devices. Xu et al. [4,5] proposed a verifiable data structure suitable for stream data integrity by improving the FHMT structure, which is more suitable for terminal devices with weak computing power to handle local dynamic stream data. However, due to the low efficiency of fully homomorphic hash encryption and its lack of support for range queries, all the above schemes are not feasible in current practical applications.

To find a Verifiable Data structure that is more suitable for the secure sharing of Streaming data, Schroeder et al. [6] defined the concept of Verifiable Data Streaming (VDS). They constructed a new data structure CAT. Essentially, CAT is a generalized MHT, that is, a notch gate hash function is introduced in the process of constructing MHT nodes. The notch gate hash function ensures the security of the notch gate through the anti-collision property of the notch gate key. It realizes the verification of big data streams while reading, writing and sharing in the stream data security sharing service, which can better solve the problems of high verification real-time performance and delay sensitivity, and is more suitable for the application scenarios of stream data security sharing. However, the structural hierarchy of the CAT structure needs to be preset, and the amount of verifiable data is limited. If the preset level is too large, it will prolong the verification path and increase the verification overhead; if it is too small, it will greatly limit the amount of data that can be processed. To optimize the CAT structure, Schöder and Simkin [7] proposed a fully dynamic f-CAT scheme that eliminates the upper bound limit and constructed a practical integrity verification framework VeriStream for secure sharing of stream data. Krupp et al. [8] proposed a new verifiable data structure by combining vector commitment and CAT, called trap vector commitment. By constructing trap vector commitment at each node, the node data in the tree can be updated without changing the root node. Later, reference [9] proposed the PB-tree structure. In each node of the complete binary tree, Bloom filters were added, which greatly improved the search efficiency of this scheme. Tsai et al. [10] added weights and

hash functions based on PB-tree, further improving the efficiency of range query and verification in the secure sharing of stream data. When verifying the real-time update of stream data for secure sharing, it is necessary to frequently use node trap gates to update the underlying nodes, and there is a risk of key leakage during the update. Sun et al. [11] constructed a double-notch gate structure for nodes based on attribute-based encryption and proposed a dynamic verifiable data structure AC-MTAT with access control to achieve adaptive verification of data streams. Miao et al. [12] also proposed a scheme supporting stream data modification based on the double notch gate hash structure. For each pair of leaf nodes, the left child node uses a one-time notch gate to update the stream data, and the right child node constructs a binary tree to store the historical stream data. To solve the privacy and security problems faced by real-time updates in the secure sharing of streaming data, Sun et al. [13] proposed an adaptive verifiable data structure P-ATHAT, which realizes privacy-protected read-write verification for big data streams. Zhang et al. [14] stored the dynamic stream data in the local cache in time series by vector commitment to reach the amount of data supported by a single vector commitment, and appended the signed commitment value at the end of the vector commitment element. Wei et al. [15] also proposed a verifiable stream data verification scheme based on accumulators and BLS aggregate signatures. The accumulator is used to record the number of update operations. After each update operation of the streaming data, its old signature will become invalid under the action of the accumulator. This scheme achieves a constant cost overhead for the verification operation of secure sharing of streaming data. However, none of the above schemes can resist the forgery and replacement attack of the cloud server.

With the development of blockchain technology, a new paradigm of on-chain and off-chain hybrid storage has emerged. References [16,17] prove in the query results returned by the storage server with a typical decentralized structure of blockchain that smart contracts can perform audits fairly and correctly. Due to the high storage cost of blockchain, its overhead is also a problem that cannot be ignored. Subsequently, the architecture of lightweight on-chain and off-chain hybrid storage was proposed [18,19]. Lightweight hybrid storage means that only a small amount of summaries from the complete data are stored on the chain as metadata, while the vast amount of raw data is fully shared with off-chain cloud storage servers. Reference [20] implements data integrity verification for keyword search in a lightweight hybrid storage architecture based on Bloom filters. Cui et al. [21] proposed a hybrid storage blockchain vs Chain for keyword search, which not only supports dynamic updates but also ensures forward privacy. For large-scale graph data and graph queries, Li et al. [22] designed the verifiable data structure MELTree for subgraph matching queries, retaining a summary of the root on the chain for verification, which further optimized the on-chain storage cost. Cai et al. [23] adopted nonlinear rules to predict the rights and interests of decentralized structure nodes to ensure the authenticity and validity of the data. Liu et al. [24] used the Bloom filter aggregated signature algorithm and the oracle commitment mechanism to solve the problem of single point of failure of data. Erway et al. [25] proposed a dynamic data possession proof scheme based on the dynamic verification data structure. Yang et al. [26] used group encryption based on certificateless public keys as a possessiveness proof scheme for data, which can effectively counter collusion attacks. Sun et al. [27] conducted an independent subtree design based on the structure of MHT and proposed a data structure suitable for efficient verification of streaming data on the blockchain. Liu et al. [28] proposed a multi-dimensional verification scheme for on-chain and off-chain balance based on accumulators and parity checks, achieving fine-grained verification of range queries.

Through the above summary of the comparative analysis of technical solutions for the verifiability of streaming data as shown in Table 1, in order to ensure the safe sharing of continuous and uncapped streaming data, it is necessary to design an efficient real-time verifiable data structure for it, oriented to the mixed

storage mode of on-chain and off-chain, how to achieve the efficient real-time verifiability of streaming data is still of great research significance.

Table 1: Comparative analysis of streaming data verifiable technology options

Literature	Key technologies	Features
[2]	Password Accumulator, GHT	Resistant to forgery, replay and substitution attacks, supports batch auditing
[3]	FHMT	Less overhead, but less efficient full homomorphic encryption
[8]	CVC	Higher overhead and limited application scenarios
[10]	Hash Function	Frequent updates increase risk of key leakage
[13]	P-ATHAT	Suitable for streaming data outsourced storage, but not for hybrid storage
[20]	Suppressing the merkle index tree	Implementing on-chain lightweight hybrid keyword store queries
[23]	Prophecy machine	Resistant to forgery and replay attacks, but less efficient
[26]	Group encryption algorithm	Resistant to collusion, forgery, replay attacks, but not applicable to streaming data
[27]	Trapped hash function	Reduces streaming data update overhead, but requires pre-set data caps
[28]	Password accumulator, parity check	Balanced on-chain and off-chain efficiencies to support finer-grained authentication

For previous studies, none of the verifiable data structures that are efficient and secure for processing large-scale streaming data have the ability to detect data tampering. In view of the efficient real-time verifiable data structure of streaming data for on-chain and off-chain hybrid storage, the problem of efficient real-time verifiable data and data tampering of continuous unbounded stream data is proposed, this paper proposes an efficient real-time verifiable technology of streaming data for on-chain and off-chain hybrid storage, and designs an efficient real-time verifiable scheme for on-chain and off-chain hybrid storage, which realizes low maintenance cost on the blockchain on the premise of preventing data tampering, and builds it in real time with the dynamic growth and change of streaming data. Enables just-in-time validation of streaming data.

3 Preparatory Knowledge

3.1 Trapdoor Hash Function

The notch gate hash function is a special kind of Merkle hash function, which has the anti-collision characteristic of hash functions. For the person holding the trap gate, there exists an algorithm that outputs a random number r' that matches m' such that $Th(m,r) = Th(m',r')$ (m,r) and (m',r') are called a pair of collisions. The notch gate hash function is collision-resistant. If the notch gate is unknown, finding two pairs of different (m,r) and (m',r') such that $Th(m,r) = Th(m',r')$ is computationally infeasible.

Definition 1: *Trap gate hash function. A trap gate hash function is composed of the following polynomial-time algorithms* $(TrapGen, Th, Col)$:

$TrapGen(1^{\lambda}) \rightarrow (tpk, tsk)$: *The initial key generation algorithm takes the security parameter* 1^{λ} *as the trap key* tsk *and the hash key* tpk.

$Th(tpk, m, r) \rightarrow Th(m)$: *The algorithm for calculating the notch gate hash value: Input the public key of the notch gate hash function, message* $m \in \{0,1\}^{in}$, *and random value* r, *and calculate the corresponding notch gate hash value through the hash key.*

$Col(tsk, m, r, m') \rightarrow r'$: *Trapdoor hash collision calculation algorithm, input the private key and message* m *of the one-way trap gate hash function, the random number* r *and the new information* m' *that needs to match the collision. The algorithm outputs a random number* r' *such that* $Th(tpk, m, r) = Th(tpk, m', r')$.

The trap gate hash function is a special type of hash function that possesses the collision resistance of hash functions, meaning that no attacker can find two pairs of different (m, r) and (m', r') mapped to the same hash value. Its formal definition is as follows.

Definition 2: *Under the security parameter* λ, *if the advantage* $Adv^{Col}_{CH,\mathcal{A}}$ *of any polynomial-time attacker* $\mathcal{A}$ *can be ignored, then the notch gate hash function is collision-resistant and can be expressed by the following formula*:

$$Adv^{Col}_{CH,\mathcal{A}} = Pr\begin{bmatrix} Ch(cpk, m, r) = Ch(cpk, m', r'); \\ (m, r) \neq (m', r'); \\ CHGen(1^{\lambda}) \rightarrow (cpk, csk); \\ (m, r, m', r') \leftarrow \mathcal{A}(Ch) \end{bmatrix}$$

The notch gate hash function adopted in the efficient and real-time stream data verifiable technology in this paper is derived from the notch gate hash function constructed based on the discrete logarithm problem presented in reference [29].

3.2 Vector Commitment

Vector Commitment (VC) maps the message vector $\left(m_1, m_2, \cdots, m_q\right)$ to a fixed-length commitment that can be opened at a specific position, proving that m_i is the i committed message. Among the confidential transactions applicable to blockchain technology, Pedersan commitment is the most widely used. It can efficiently create commitment schemes and has good hiding and binding characteristics.

In the Pedersan commitment, two orders are selected as prime numbers p in group $\mathbb{G}_1$. Select two generators g and y from the $\mathbb{G}_1$ groups, as well as a random element r; Output $c = g^m y^r$. If group $\mathbb{G}_1$ is safe under the discrete logarithm assumption, it is promised that c has good hiding and binding characteristics.

Definition 3: *The vector commitment algorithm consists of the following several polynomial-time algorithms* $(Gen, Com, Open, Ver)$:

$Gen\left(1^{\lambda}, q\right) \rightarrow pp$: *The key generation function takes the security parameter* λ *and the vector size* q *as input and outputs the common parameter* pp.

$Com_{pp}\left(m_1, m_2, \cdots, m_q, r\right) \rightarrow (c, aux)$: *The commitment function takes in a vector composed of* q *messages and a random number* r, *and outputs a commitment* c *and auxiliary information* aux.

$Open_{pp}\left(i, m, c, aux\right) \rightarrow (\pi)$: *Output* π *to prove that message* m *is the* i*-th message in vector commitment* c.

$Ver_{pp}\left(i, m, c, \pi\right) \rightarrow (true/false)$: *Verify whether message* m *is the* i*-th message in vector commitment* c. *If it is, output true; otherwise, output* $false$. *Data consistency is disrupted and it no longer has integrity.*

Definition 4: *Under the security parameter λ, if the advantage $Adv_{CH,\mathcal{A}}^{Col}$ of any polynomial-time attacker $\mathcal{A}$ can be ignored, then the vector commitment is ordered, where:*

$$Adv_{CH,\mathcal{A}}^{Col} = Pr\begin{bmatrix} Com_{pp}(aux, m, i) = Ch(cpk, m', i'); \\ (m, i, c) \neq (m', i', c'); \\ Gen(1^{\lambda}) \rightarrow (cpk, csk); \\ (m, i, m', i') \leftarrow \mathcal{A}(Ch) \end{bmatrix}$$

4 System Model and Threat Model

4.1 System Model

An efficient, real-time and verifiable system model for stream data design for on-chain and off-chain hybrid storage, as shown in Fig. 2, consists of the following four parts.

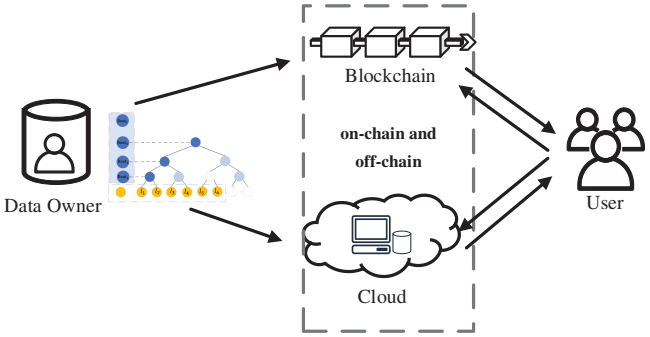

Figure 2: An efficient, real-time and verifiable system model for stream data in on-chain and off-chain hybrid storage

Data owner: The data owner refers to the data source, the owner of the streaming data. As a data source that continuously generates stream data, before securely sharing the data, it needs to perform a series of data processing operations on the original data, store the complete data in the cloud server, and store the vector commitment on the blockchain.

Cloud servers: They store and maintain a complete and verifiable data structure for streaming data, providing vast data storage space and powerful information processing capabilities for secure data sharing. They manage data with stored data files and respond to users' data query requests.

Blockchain: It is a third party that conducts fair audits, stores dynamic root nodes of data, and its verification is accomplished by decentralized smart contracts. Subsequently, it utilizes a small amount of key verification evidence stored on-chain to verify the storability proof returned by the cloud server, and further outputs fair audit results to ensure the fairness of the verification results.

User: Who is the legitimate visitor to the streaming data, initiates a data access request to the cloud server. After the cloud server verifies the identity, it responds and then returns the query result and related evidence for integrity verification. The user verifies the data integrity by calculating the information to be verified locally and sending it to the blockchain to call the verification smart contract.

4.2 Threat Model

The security threats of the designed model usually come from two aspects: blockchain nodes and semi-honest cloud servers.

(1) Semi-honest blockchain nodes: In this paper, it is assumed that most of the miner nodes in the blockchain are honest and curious, and as a limited extended chain, in order to avoid the cost consumed by verification, thus modifying the verification results and forging proofs to deceive other honest users.
(2) Semi-honest cloud server: the data of semi-honest cloud servers is damaged, or in order to obtain additional benefits, some infrequently accessed data or key data blocks of some files are directly deleted. When these damaged data are accessed by users, the cloud server attempts to forge the accessed data through leaf data under the same node. The most direct security risk lies in attempting to secure the sharing of data through the auditing process of smart contracts with the same verification information on the chain.

5 Efficient Real-Time Verifiable Technology for Stream Data in On-Chain and Off-Chain Hybrid Storage

5.1 An Efficient, Real-Time and Verifiable Data Structure for Stream Data of On-Chain and Off-Chain Hybrid Storage

Based on the notch gate hash function and vector commitment technology, an adaptive notch gate hash tree structure was constructed. An efficient real-time verifiable data structure for stream data in on-chain and off-chain hybrid storage was designed. The dynamic root node was sequentially bound with the ordered leaf nodes under this root node. Only the vector commitment of the dynamic root node was stored on-chain, and the complete data structure was stored off-chain. It not only effectively prevents off-chain cloud server tampering, but also ensures sufficient storage space to achieve immediate verification of stream data upon arrival. This verifiable data structure does not require the initialization of the depth of the verification tree and can adaptively expand according to the change in the scale of the data stream. It realizes the verification of big data streams while reading, writing and sharing in the stream data security sharing service. It can better solve problems such as the unpredictable scale of stream data, high verification real-time performance and delay sensitivity, and is suitable for the massive stream data sharing scenarios under the cloud platform. When stream data is inserted and updated as the underlying leaf node, there is always a unique notch gate hash node corresponding to it in the verification tree. Among them, leaf nodes all take the root node of the smallest subtree they are in as verification information, which is recorded on the blockchain. At the same time, a shorter verification path is generated, and the problem of single point of failure is also avoided.

During the adaptive expansion process of the off-chain complete verifiable data structure, each time the current structure fails to meet the data scale, a new root node is generated as verification information and stored on the blockchain. The original structure becomes the left subtree of the newly generated root node, with the depth increasing by 1, and the amount of data that can be supported expands to twice the original. As shown in Fig. 3, the structures of verification trees at three different scales are presented. The structures of verification trees at different scales are consistent, and the verification tree with a smaller scale is the left subtree of the larger verification tree. Since the amount of supported data doubles each time the validation tree is expanded, the expansion operations in the subsequent stages will not be particularly frequent.

After inserting every 2^i data blocks, the hash tree structure will form a full binary tree structure. To continue inserting new data blocks, the adaptive notch gate hash tree will adaptively expand one layer upwards. The original structure becomes a subtree of the newly generated root node, which is used as the verification root node for inserting data block m into its right subtree. Each time a new subtree is generated, the root node is $Root_i(0 < i \leq n)$. The set of root nodes is uploaded to the blockchain as verification evidence.

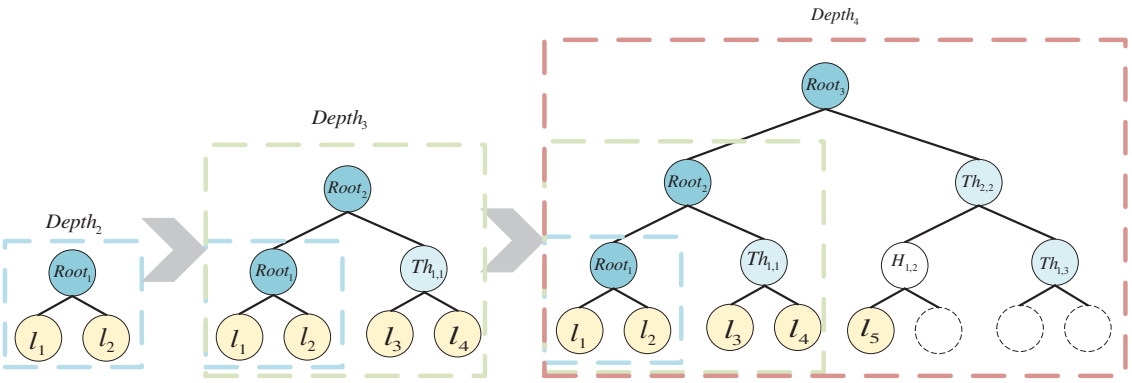

Figure 3: Adaptive hinged gate hash tree structures at different scales

In addition, the stream data is ordered. Although in the storage structure of the binary tree, there is unidirectionality from leaf nodes to the root node, in the continuously expanding notch gate hash tree, each time a root node is dynamically generated, the unidirectional hash operation of the leaf nodes under that dynamic root node can be verified. However, on the chain, due to limited scalability, only the information of the dynamic root node is stored. The situation as shown in Fig. 4 will occur. For instance, if the data stored in the cloud server has been damaged, in order to maintain its reputation, the cloud server forges other leaf data under the same layer root node at the damaged data location, such as tampering with the data by exchanging the sequence of 5 and 7. For the data that the user hopes to query, the query result returned by the cloud server should be $l_5' = l_5$. After forging data through the cloud server, $l_5' = l_7$ data was actually returned. And since the cloud server simultaneously holds the verification path information from all leaf nodes to the root node, as long as the corresponding data and its verification path information are returned, the on-chain verification results of the leaf nodes under the same root node are the same. Therefore, the integrity of the query stream data cannot be effectively verified. Therefore, based on vector commitment, the sequentiality of stream data is guaranteed to achieve efficient, real-time and verifiable stream data for on-chain and off-chain hybrid storage.

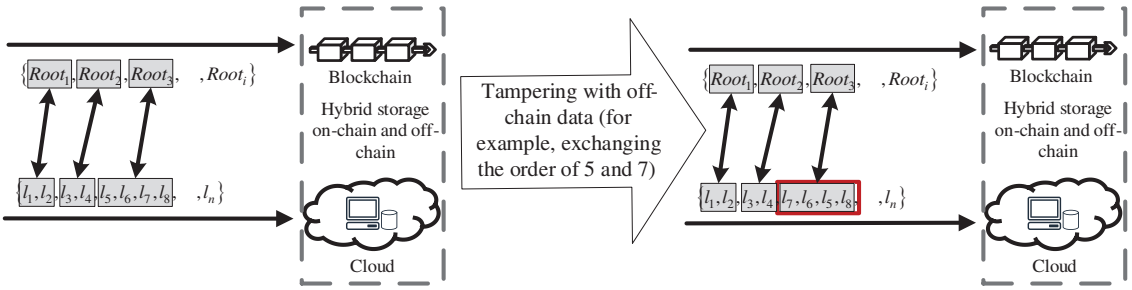

Figure 4: Schematic diagram of the problem of data forgery in cloud servers

Therefore, in the hinged gate hash tree structure, the dynamic root node is sequentially bound to all the leaf nodes under the root node based on vector commitment to ensure the secure sharing of stream data. That is, each layer's root node makes a commitment to its affiliated leaf node, that is, the sequence range of verifiable stream data. That is, data set $D_d = \{Root_d, l_{2^{d-1}+1}, l_{2^{d-1}}, \cdots, l_{2^d}, \}$ generates a commitment C_d based on vector commitment, and $Com_{pp}(D_d, r) \rightarrow (C_d, aux)$ uplores the commitment to the blockchain as evidence for data integrity verification.

As shown in Fig. 5, an efficient real-time verifiable data structure for stream data is designed based on an adaptive slot gate hash tree. The dynamic root node is sequentially bound to the ordered leaf nodes under this root node. Only the vector commitment of the dynamic root node is stored on the chain, and the complete data structure is stored off-chain. This not only effectively prevents off-chain cloud server tampering but also ensures sufficient storage space. Realize the immediate verification of stream data upon arrival.

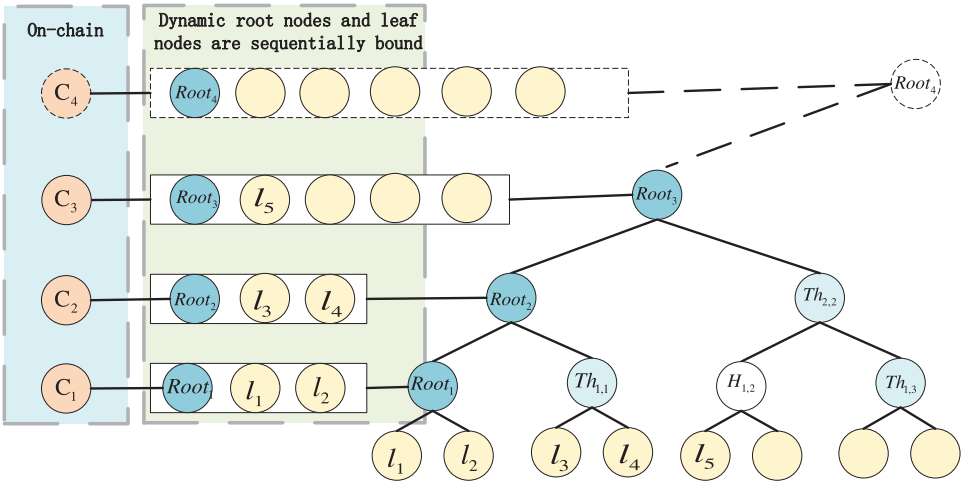

Figure 5: Adaptive notch gate hash tree verification data structure

5.2 Formal Definition

The efficient real-time verifiable scheme for stream data in on-chain and off-chain hybrid storage consists of tuples of the following polynomial-time algorithms: $\left(Source_{Append}; Server_{Append}; BC_{Append}\right)$.

5.2.1 Data Source Data Addition Algorithm

$Source_{Append}\left(m, st, r\right) \rightarrow (C)$: This algorithm is run by the data source, namely the data owner. For the real-time arriving stream data, an adaptive slot gate hash tree structure is constructed based on the slot gate hash function and vector commitment technology. And when the stream data forms a full binary tree each time, this root node is taken as the first value, and together with all the leaf nodes of its right subtree, namely $D_d = \{Root_d, l_{2^{d-1}+1}, l_{2^{d-1}}, \cdots, l_{2^d}\}$, a vector commitment C_d, $Com_{pp}\left(D_d, r\right) \rightarrow \left(C_d, aux\right)$ is generated. And upload it to the blockchain. When the depth value increases by $d \leftarrow d+1$, the capacity of the verification tree doubles by $capacity = capacity * 2$, generating a new root node $new\ root \leftarrow R_d$. The specific process refers to the Algorithm 1: data source data addition algorithm:

Algorithm 1: Data source data addition algorithm $Source_{Append}\left(m, st, r\right)$

//Input: Data blocks m ***, state vectors*** st ***, and random numbers promised*** r ***by generation vectors***
//Output: Vector commitment c ***and auxiliary information*** aux
$for\ h = 0\ to\ st.dummySet.length\ \{$
$if\ (c = capacity)$
$\quad C.add = Com_{pp}\left(root, m_{c/2+1}, m_{c/2+2}, \cdots, m_c, r\right)$
$\quad if\ (this.capacity == 0)\ \{$
$\quad\quad this.capacity = 1;\}$
$\quad else\{$

(Continued)

Algorithm 1 (continued)

```
        this.capacity* = 2;
        Root.add = new root;
        Root.leftChild = this.root;
        this.root = Root; }
   Depth + +; }
return C;
```

5.2.2 Blockchain Data Addition Algorithm

$BC_{Upload}(Root, C)$: After the data owner constructs the notch gate hash tree structure and vector commitment for the stream data arriving locally, the complete verifiable data structure is uploaded to the cloud server, and the vector commitment of the root node of each layer is uploaded to the blockchain as verification information. The specific process can be referred to the Algorithm 2: blockchain data addition algorithm.

Algorithm 2: Blockchain data addition algorithm $BC_{Upload}(Root, C)$

```
//Input: The notch gate hash tree generates new root nodes Root for each layer and the corresponding vector commitments C
//Output: On-chain feedback True/False
blockchain.upload({
    'Root_hash': Root.hash_value;
        'VC': C.commitment_value;
          })
        return True;
```

5.2.3 Blockchain Data Verification Algorithm

$BC_{Verify}(\pi, Root', data\_block) \rightarrow True/False$: A legitimate user queries the stream data at location i from the cloud server. After verifying the identity, the cloud server responds to the legitimate user's inquiry and returns the stream data m at location i and the corresponding verification path $auth'$ stored in the cloud server. It also calculates the evidence $Open_{pp}(i, m, c, aux) \rightarrow (\pi)$ of the vector commitment where the stream data is located, which proves that message m is the i-th message in vector commitment c. The user calculates the root node of the information to be verified locally through the verification path and initiates a verification request to the blockchain. The blockchain verifies the integrity of the data based on the query location i, root node $Root'$ and the corresponding commitment evidence π sent by the user. The specific algorithm refers to the Algorithm 3: blockchain data verification algorithm.

Algorithm 3: Blockchain data verification algorithm $BC_{Verify}(\pi, Root', data\_block)$

```
//Input: The root hash of this layer calculated Root' by the user, the commitment evidence π , and the on-chain query data block data_block
//Output: Feedback on verification results True/False
current_hash = hash_function(data_block)
   for node in Root'
       if node.side == 'left'
```

(Continued)

Algorithm 3 (continued)

$current\_hash =$
$hash\_function\,(node.hash\_value + current\_hash)$;
$else$
$current\_hash =$
$hash\_function\,(current\_hash + node.hash\_value)$;
$if\ current\_hash.root! = Root'$
$returnFalse$;
$if\ !Ver_{pp}\,(i, m, current\_hash.VC, \pi)$
$return\ False$;
$return\ True$;

6 Safety Analysis

To ensure the security of the efficient real-time verifiable technology for stream data in on-chain and off-chain hybrid storage, this section defines a security model Ck, mainly involving correctness, security and integrity.

6.1 Proof of Correctness

In the scheme designed in this chapter, if λ is the public parameter correctly generated by the initialization algorithm $Setup$, the data owner and the algorithm calling $Append$ insert $q := q\,(\lambda)$ data, and correctly update the corresponding node hash values, random numbers and attribute labels. Based on the sink gate hash function and vector commitment, the verification evidence is constructed and the $Upload$ function is called and uploaded to the blockchain. Due to the immutable feature of the blockchain, the user initiates a verification request to the smart contract for the data questioned to the cloud server, calls the $Verify$ algorithm, and the output is $True$. Then, the cloud server has given the correct query feedback to the data, and the user's query of the data is correct. Otherwise, if the data is tampered with and the on-chain verification fails, then the scheme is said to be correct.

6.2 Proof of Security

The security of the scheme in this chapter is proved through the interactive game between Challenger $\mathcal{C}$ and Attacker $\mathcal{A}$. Suppose the depth of the notch gate hash tree constructed based on the test data is $D = ploy\,(\lambda)$. If Opponent $\mathcal{A}$ wants to win, then it needs to output a tuple $(\pi', Root')$ that satisfies $(\pi', Root') \notin Q$ and $BC_{Verify}\,(\pi', Root', data\_block) = True$. Then it is claimed that it has won the security game, with the advantage being: $\mathrm{Adv}_{\Pi,i}^{CH\_VC}\,(\lambda) = \Pr\left[\mathrm{Exp}_{\Pi,i}^{CH\_VC}\,(\lambda) - 1\right]$.

The attacker modifies or replaces one or more elements in the query results. The forged query results contain elements that do not belong to the real data, that is, the structural retention of the scheme is compromised.

Theorem 1: *If CH is a collision-resistant notch gate hash function and VC is a sequential vector commitment, then the scheme in this chapter has structural preservation.*

Proof: Assuming that the verification tree does not have structural retention, there exists an effective opponent A that can win the defined secure interactive game with a non-negligible probability. Algorithm B_{Ch} can find the collision of CH in the trap gate hash tree. A calculates the collision through this algorithm to generate $(m', auth', \pi')$, where $auth'$ is the set of verification path nodes from leaf node m' to the root node. The following is discussed in two cases.

If it is $auth' \neq auth_i$, then $m' \neq m_i$. Meanwhile, $\pi' \neq \pi_i$ and A have forged data and paths. If the forged data and the two real paths can calculate the same root node $Root'$, attempting to obtain the correct verification result on the chain through $Root'$, and if the verification is passed, then $Ver_{pp}(i, m', current\_hash.VC, \pi') = Ver_{pp}(i, m_i, current\_hash.VC, \pi_i)$, this is contradictory to $\pi' \neq \pi_i$, proving that the scheme in this chapter has structural retention.

If $auth' = auth_i$, A finds a collision, but this contradicts the collision-resistant property of the hash function. Therefore, the scheme of this chapter has structural retention.

Therefore, the efficient real-time verifiable technology of streaming data can ensure the secure sharing of data for on-chain and off-chain hybrid storage. □

6.3 Proof of Integrity

In this chapter, for the efficient real-time verifiable technology of stream data for on-chain and off-chain hybrid storage, it is ensured that the efficient real-time stream data is uniformly mixed stored off-chain and off-chain, that is, to resist the forgery and tampering attacks of other entities. The sequentiality of the complete stream data of the cloud server is successfully verified. Given the security parameter δ, it is stipulated to the decentralized blockchain as an on-chain third-party impartial audit. If the data is lightweight stored on the chain. When the complete off-chain data encounters damage such as single point of failure, the consistency of on-chain and off-chain data is disrupted, that is, the security parameters are missing, and the on-chain stored information cannot pass the verification of data integrity. Based on the immutable feature of blockchain, data integrity can be verified by storing information on the chain, and users are allowed to verify the integrity of streaming data. Then, the scheme in this chapter can provide integrity protection for the secure sharing of data for on-chain and off-chain hybrid storage.

7 Performance Evaluation

7.1 Proof of Correctness

Theoretically, the update efficiency of the verification tree structure during the sharing of stream data insertion and the verification efficiency during data access are deduced and analyzed in detail:

In the initial stage, the depth value of the Schroder verification tree is taken as a constant D. In this scheme, the depth of the verification tree based on the notch gate hash function is $d = 0$, and it adaptively changes dynamically with data insertion subsequently. When performing real-time insertion and update in the verification tree, for half of the nodes in the tree, only the collision value of the upper slot gate hash node needs to be calculated during the insertion and update, while for the quarter of the nodes, two nodes need to be calculated upward. By doing so, the update path when inserting data can be obtained. In the verification tree based on the slot gate hash function, when all leaves insert data, the average update length of the verification path from leaf nodes to dynamic root nodes is:

$$Append\_len = 1 \times \frac{1}{2} + 2 \times \frac{1}{4} + 3 \times \frac{1}{8} + \ldots + (d-1) \times \frac{1}{2^{d-1}} + d \times \frac{1}{2^d} = 2 - \frac{1}{2^{d-1}}$$

The notch gate hash function has the characteristic of adaptive expansion. At any moment, the data volume of the verification tree is always matched with the current depth. Therefore, no matter how large the scale of the tree is, the average update length when data is inserted is stable at about 2. Fig. 6 shows the comparison of the average insertion update path length with depth in the scheme of this chapter based on the trap gate hash function verification tree structure and the Merkle hash verification tree (MHT) structure.

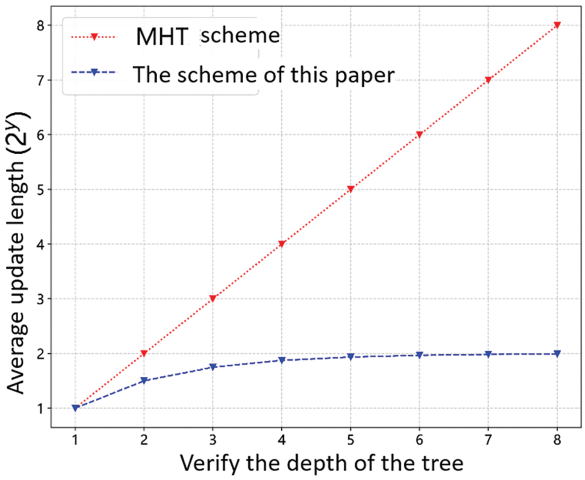

Figure 6: A comparison chart of the average insertion update path length varying with depth

7.2 Theoretical Analysis

Based on the verifiable data structure, first ignoring the Gas cost of blockchain storage access in Reference [6] and the scheme in this chapter, the computational complexity of various operations in the following four schemes was compared, as shown in Table 2. The scheme in this chapter has the minimum initialization overhead in the initial stage, and at the same time, insert. The computational complexity of both the query and verification processes has been reduced from the original $O\left(\log_2 n\right)$ to $O\left(\log_2 i\right) 1 \le i \le n$. That is to say, as the amount of processed data increases, the time consumed by the scheme in this chapter has a smaller increase rate compared to other schemes.

Table 2: Comparison of computational complexity of different operations in different schemes

	Preprocessing	**Insertion**	**Query**	**Verification**
[5]	$O\left(2n-1\right)$	$O\left(\log_2 n\right)$	$O\left(\log_2 n\right)$	$O\left(\log_2 n\right)$
[6]	$O\left(\log_2 n\right)$	$O\left(\log_2 n\right)$	$O\left(\log_2 n\right)$	$O\left(\log_2 n\right)$
[14]	$O\left(1\right)$	$O\left(\log_2 i\right)$	$O\left(\log_2 n\right)$	$O\left(\log_2 n\right)$
The scheme	$O\left(1\right)$	$O\left(\log_2 i\right)$	$O\left(\log_2 i\right)$	$O\left(\log_2 i\right)$

7.3 Functionalities Analysis

This section conducts theoretical analysis on the real-time performance and third-party audit fairness of the efficient real-time verifiable scheme for stream data in on-chain and off-chain hybrid storage, and proves the efficiency of the scheme in this chapter.

(1) Real-time performance: In this chapter, an efficient real-time verifiable data structure for ensuring the integrity of stream data is designed based on the slot gate hash function. The initial construction does not require specifying the depth. The storage scale is adaptively expanded according to the amount of arriving data, achieving immediate verification in the secure sharing of stream data. It can be better applied to the big data stream scenarios under the cloud platform. The verification efficiency of secure sharing of stream data in practical applications has been improved.

(2) Fairness of third-party auditing: The existing stream data security sharing schemes, due to the lack of trusted third-party auditing, have the security risk of data tampering. This chapter is based on decentralized blockchain as a trusted third-party audit, which is regulated by the security of smart contracts. Due to the immutable nature of blockchain, once a smart contract is triggered for verification, the audit is not controlled by any single private node. Therefore, blockchain serves as a third-party audit in the scenario of secure sharing of streaming data, ensuring that the verification process and results are not tampered with. Ensure the fairness of the verification results.

7.4 Experimental Analysis

The experimental environment for fabric blockchain simulation towards on-chain-off-chain hybrid storage is built through VMware virtual machine with Ubuntu 18.04.1 system. To simulate the on-demand verification of secure sharing of streaming data, the process of collecting streaming data by the data owner (IoT device) is simulated by the software, and a total of 2^8 data block is simulated, and the size of each data block is 1 MB.

The effectiveness of the scheme in this chapter in practical use is evaluated through experiments for comparative analysis such as overheads, due to the fact that the scheme in this chapter is targeted at hybrid on-chain and off-chain storage of efficient verifiable data structures. Therefore, this section also implements the data security sharing scheme with GEM2-Tree, as shown in Fig. 7, by comparing the maintenance cost of the two structures on the blockchain when the data is updated with different efficiencies, with the constant change of the data update rate of the on-chain off-chain hybrid storage, the consumption of Gas on the chain of this scheme is significantly better than that of the GEM2-Tree scheme, which ultimately verifies the high efficiency of the scheme in this chapter.

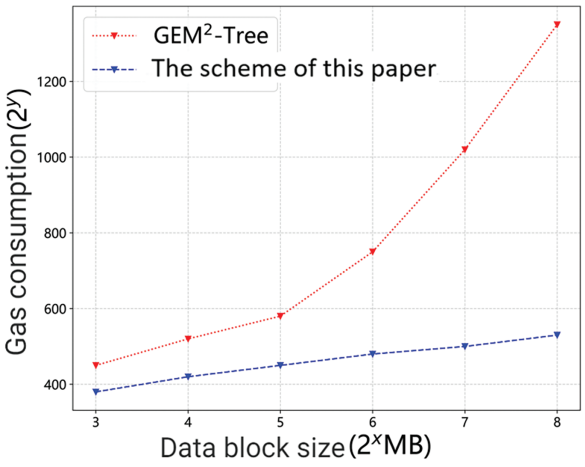

Figure 7: Gas consumption vs. renewal rate

8 Summary

This paper proposes an efficient, real-time and verifiable technology for stream data in on-chain and off-chain hybrid storage. Based on the notch gate hash function and vector commitment, an adaptive notch gate hash tree structure is constructed in real time with the dynamic growth and changes of big data streams. An efficient real-time verifiable data structure for on-chain and off-chain stream data is proposed. The dynamic root node is sequentially bound with the ordered leaf nodes under this root node, and only the vector commitment of the dynamic root node is stored on the chain. Off-chain storage of complete data structures

not only effectively prevents off-chain cloud server tampering but also ensures sufficient storage space. There is no need to specify the depth during the initial construction, achieving immediate verification for secure sharing of stream data. Moreover, the computational overhead of on-chain and off-chain hybrid storage verification is only related to the current data volume, making it more practical when dealing with stream data with unpredictable data volumes. Based on the above research, an efficient real-time verifiable scheme for streaming data was designed. Finally, through security analysis and performance analysis, the security and efficiency of this technology are proven, and it is demonstrated that this technology can prevent attacks such as forgery and tampering.

9 Future Work

In future work, our research direction will focus on two key areas: Regarding the development of on-chain-off-chain hybrid storage, not only for large-scale streaming data, but also for the design of verifiable data structures for the storage of various types of data such as addition, deletion, modification, etc., the scalability and security of the hybrid storage need to be further improved. I believe that this new paradigm of secure sharing has a broad development prospect.

Acknowledgement: Not applicable.

Funding Statement: 1. This work has been supported by the National Cryptologic Science Fund of China (Grant No. 2025NCSF02020) awarded to Yi Sun. 2. This work has been supported by the Natural Science Foundation of Henan Province (Grant No. 242300420297) awarded to Yi Sun.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Wei Lin; data collection: Wei Lin; analysis and interpretation of results: Wei Lin; draft manuscript preparation: Wei Lin, Yi Sun. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available from the corresponding author, Yi Sun, upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Nakamoto S. Bitcoin: a peer-to-peer electronic cash system [Internet]; 2008 [cited 2025 Aug 1]. Available from: https://bitcoin.org/.pdf.
2. Papamanthou C, Shi E, Tamassia R, Yi K. Streaming authenticated data structures. In: Advances in cryptology—EUROCRYPT 2013. Berlin/Heidelberg, Germany: Springer; 2013. p. 353–70. doi:10.1007/978-3-642-38348-9_22.
3. Yu CM. POSTER: lightweight streaming authenticated data structures. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security; 2015 Oct 12–16; Denver, CO, USA. New York, NY, USA: ACM; 2015. p. 1693–5. doi:10.1145/2810103.2810117.
4. Xu J, Wei L, Wu W, Wang A, Zhang Y, Zhou F. Privacy-preserving data integrity verification by using lightweight streaming authenticated data structures for healthcare cyber-physical system. Future Gener Comput Syst. 2020;108(1):1287–96. doi:10.1016/j.future.2018.04.018.
5. Xu J, Wei L, Zhang Y, Wang A, Zhou F, Gao CZ. Dynamic Fully Homomorphic encryption-based Merkle Tree for lightweight streaming authenticated data structures. J Netw Comput Appl. 2018;107(3):113–24. doi:10.1016/j.jnca.2018.01.014.

6. Schroeder D, Schroeder H. Verifiable data streaming. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security; 2012 Oct 16–18; Raleigh, NC, USA. New York, NY, USA: ACM; 2012. p. 953–64. doi:10.1145/2382196.2382297.
7. Schöder D, Simkin M. VeriStream—a framework for verifiable data streaming. In: Financial cryptography and data security. Berlin/Heidelberg, Germany: Springer; 2015. p. 548–66. doi:10.1007/978-3-662-47854-7_34.
8. Krupp J, Schröder D, Simkin M, Fiore D, Ateniese G, Nuernberger S. Nearly optimal verifiable data streaming. In: Public-key cryptography—PKC 2016. Berlin/Heidelberg, Germany: Springer; 2016. p. 417–45. doi:10.1007/978-3-662-49384-7_16.
9. Li R, Liu AX, Wang AL, Bruhadeshwar B. Fast range query processing with strong privacy protection for cloud computing. Proc VLDB Endow. 2014;7(14):1953–64. doi:10.14778/2733085.2733100.
10. Tsai IC, Yu CM, Yokota H, Kuo SY. VENUS: verifiable range query in data streaming. In: Proceedings of the IEEE INFOCOM 2018—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS); 2018 Apr 15–19; Honolulu, HI, USA. Piscataway, NJ, USA: IEEE; 2018. p. 160–5. doi:10.1109/INFCOMW.2018.8406898.
11. Sun Y, Chen XY, Du XH, Xu J. Dynamic authenticated method for outsourcing data stream with access control in cloud. Chin J Comput. 2017;40(2):337–50. (In Chinese).
12. Miao M, Li J, Wang Y, Wei J, Li X. Verifiable data streaming protocol supporting update history queries. Int J Intell Syst. 2022;37(12):11342–61. doi:10.1002/int.23045.
13. Sun Y, Liu Q, Chen X, Du X. An adaptive authenticated data structure with privacy-preserving for big data stream in cloud. IEEE Trans Inf Forensics Secur. 2020;15:3295–310. doi:10.1109/TIFS.2020.2986879.
14. Zhang Z, Chen X, Ma J, Tao X. New efficient constructions of verifiable data streaming with accountability. Ann Telecommun. 2019;74(7):483–99. doi:10.1007/s12243-018-0687-7.
15. Wei J, Tian G, Shen J. Optimal verifiable data streaming protocol with data auditing. In: Computer security—ESORICS 2021. Berlin/Heidelberg, Germany: Springer; 2021. p. 296–312 doi:10.1007/978-3-030-88428-4_15.
16. Wang M, Guo Y, Zhang C, Wang C, Huang H, Jia X. MedShare: a privacy-preserving medical data sharing system by using blockchain. IEEE Trans Serv Comput. 2021;16(1):438–51. doi:10.1109/TSC.2021.3114719.
17. Zhang X, Xu Z, Cheng H, Che T, Xu K, Wang W, et al. Secure collaborative learning in mining pool via robust and efficient verification. In: Proceedings of the 2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS); 2023 Jul 18–21; China. Piscataway, NJ, USA: Hong Kong; 2023. p. 794–805. doi:10.1109/ICDCS57875.2023.00012.
18. Peng Z, Xu J, Hu H. BlockShare: a blockchain empowered system for privacy-preserving verifiable data sharing. IEEE Data Eng Bull. 2022;45(2):14–24.
19. Sun Z, Zhao P, Wang C, Zhang X, Cheng H. An efficient and secure trading framework for shared charging service based on multiple consortium blockchains. IEEE Trans Serv Comput. 2022;16(4):2437–50. doi:10.1109/TSC.2022.3216659.
20. Zhang C, Xu C, Wang H, Xu J, Choi B. Authenticated keyword search in scalable hybrid-storage blockchains. In: Proceedings of the 2021 IEEE 37th International Conference on Data Engineering (ICDE); 2021 Apr 19–22; Chania, Greece. Piscataway, NJ, USA: IEEE; 2021. p. 996–1007. doi:10.1109/ICDE51399.2021.00091.
21. Cui N, Wang D, Li J, Zhu H, Yang X, Xu J, et al. Enabling efficient, verifiable, and secure conjunctive keyword search in hybrid-storage blockchains. IEEE Trans Knowl Data Eng. 2023;36(6):2445–60. doi:10.1109/TKDE.2023.3324128.
22. Li S, Zhang Z, Zhang M, Yuan Y, Wang G. Authenticated subgraph matching in hybrid-storage blockchains. In: Proceedings of the 2024 IEEE 40th International Conference on Data Engineering (ICDE); 2024 May 13–16; Utrecht, Netherlands. Piscataway, NJ, USA: IEEE; 2024. p. 1986–98. doi:10.1109/ICDE60146.2024.00159.
23. Cai Y, Irtija N, Tsiropoulou EE, Veneris A. Truthful decentralized blockchain oracles. Int J Netw Mgmt. 2022;32(2):e2179. doi:10.1002/nem.2179.
24. Liu X, Feng J. Trusted blockchain oracle scheme based on aggregate signature. J Comput Commun. 2021;9(3):95–109. doi:10.4236/jcc.2021.93007.
25. Erway CC, Küpçü A, Papamanthou C, Tamassia R. Dynamic provable data possession. ACM Trans Inf Syst Secur. 2015;17(4):1–29. doi:10.1145/2699909.

26. Yang G, Han L, Bi J, Wang F. A collusion-resistant certificateless provable data possession scheme for shared data with user revocation. Clust Comput. 2024;27(2):2165–79. doi:10.1007/s10586-023-04078-8.
27. Sun YS, Yang JC, Xia Q. SMT: efficient authenticated data structure for streaming data on block-chain. J Softw. 2023;34(11):5312–29. (In Chinese).
28. Liu Q, Peng Y, Xu M, Jiang H, Wu J, Wang T, et al. MPV: enabling fine-grained query authentication in hybrid-storage blockchain. IEEE Trans Knowl Data Eng. 2024;36(7):3297–311. doi:10.1109/tkde.2024.3359173.
29. Garman C, Green M, Miers I. Decentralized anonymous credentials. In: Proceedings of the 2014 Network and Distributed System Security Symposium; 2014 Feb 23–26; San Diego, CA, USA. doi:10.14722/ndss.2014.23253.