

Article

Blockchain-Enhanced Cross-Chain Storage and Access for Industrial Internet of Things Data

Yuexia Zhang ^{1,2,*}, Yutong Wang ^{1,2}, Jie Lu ^{1,2} and Haonan Zhang ^{1,2}

¹ Center for Target Cognition Information Processing Science and Technology, Beijing Information Science and Technology University, Beijing 102206, China; pripiyat77@163.com (Y.W.); lujie_bistu@163.com (J.L.); zhang.haonan@bistu.edu.cn (H.Z.)

² Key Laboratory of Modern Measurement & Control Technology, Ministry of Education, Beijing Information Science and Technology University, Beijing 102206, China

* Correspondence: zhangyuexia@bistu.edu.cn

Abstract

To support data sharing among industrial devices and information systems, blockchain provides a shared, tamper-evident ledger for recording data storage and access in the Industrial Internet of Things (IIoT). However, managing heterogeneous IIoT data across multiple blockchains requires coordination of storage, access authorization, and cross-chain interactions. To address these challenges, this paper proposes a blockchain-enabled cross-chain data storage and sharing scheme for IIoT. Firstly, a three-layer IIoT model is introduced, encompassing the device layer, storage layer, and blockchain layer, employing a multi-chain distributed storage mode to handle data diversity. Subsequently, based on this model, the overall architecture and key interaction sequences of the cross-chain storage system are designed, clarifying the execution processes for data uploading, cross-chain requests, and verification. Finally, a concurrent contract-based algorithm is introduced to coordinate cross-chain data access. Simulations demonstrate that the proposed algorithm enhances data security and access efficiency. The proposed cross-chain data storage system significantly improves throughput compared to single-chain systems, while the node fault tolerance rate remains largely consistent with that of single-chain consensus. Simulations demonstrate that the proposed CPHTLC algorithm enhances data security and access efficiency. The proposed cross-chain data storage system significantly improves throughput compared to single-chain systems, while the node fault tolerance rate remains largely consistent with that of single-chain consensus.

Keywords: Industrial Internet of Things; blockchain; cross-chain storage; CPHTLC; data security



Academic Editors: Weiwei Jiang,
Chunjiong Zhang and Tao Xie

Received: 20 August 2026

Revised: 7 October 2026

Accepted: 16 October 2026

Published: 18 October 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

The integration of next-generation information technology with manufacturing has made the Industrial Internet of Things a major driver of global industrial transformation [1–3]. Intelligent sensing devices, industrial networks, and cloud platforms enable real-time perception, precise control, and informed decision-making throughout production, thereby improving operational efficiency [4–7]. However, the rapid growth of connected devices has substantially increased both the volume and heterogeneity of industrial data. Data types have also become more complex and diverse as IIoT deployments have expanded. Mainstream storage solutions still rely heavily on centralized server architectures, whose security limitations are increasingly evident in industrial environments. Centralized nodes represent

attractive single points of failure and are vulnerable to data tampering, leakage, and service interruption [8–10]. A successful cyberattack on such a node can affect both the availability and credibility of the associated industrial data. Consequently, centralized architectures may not satisfy industrial requirements for data integrity, confidentiality, and continuous service. They also face limited storage capacity, high processing latency, and isolated data silos when handling massive heterogeneous datasets [11,12]. Replication across distributed storage nodes reduces dependence on any single node for data availability. In heterogeneous IIoT deployments, however, maintaining storage-location records and access permissions across participating systems introduces an additional coordination requirement. Unresolved security, performance, and coordination issues can impede real-time industrial applications and the reliable long-term development of IIoT. These security and performance limitations impede real-time industrial applications and the reliable long-term development of IIoT. They therefore create an urgent need for a storage architecture that can support secure, scalable, and trustworthy industrial data sharing.

To address the trust and integrity limitations of centralized storage, academia and industry have introduced blockchain technology into IIoT data management [13–15]. Its decentralization, immutability, and traceability provide useful properties for secure data management. Distributed ledgers support data integrity and credibility, whereas smart contracts automate the management of data-access permissions [16]. In an IIoT environment, blockchain can record the full progression from device collection and transmission to final storage. This immutable chain of records supports data traceability, source verification, and accountability [17]. Nevertheless, solving the trust problem through blockchain does not by itself resolve the performance and heterogeneity problems identified above. Conventional single-chain architectures encounter network-performance bottlenecks in complex IIoT scenarios. Their limited processing capacity makes it difficult to satisfy the real-time requirements of massive industrial datasets [18]. Moreover, industrial data types differ in their security, latency, and storage-cost requirements. A single chain lacks the flexibility to accommodate these heterogeneous and sometimes conflicting demands. This mismatch between trusted storage and differentiated data requirements motivates a move beyond the single-chain architecture.

Accordingly, cross-chain storage and access technologies have emerged to overcome the constraints of single-chain architectures [19,20]. A collaborative multi-chain ecosystem can assign different types of industrial data to suitable blockchains and thereby optimize storage-resource allocation. By enabling data sharing and smart-contract invocation across heterogeneous blockchains, cross-chain interoperability allows platforms with different capabilities to work together [21,22]. This assignment allows each chain to serve data with compatible security, real-time, and cost requirements. However, distributing data across multiple chains introduces a new problem: heterogeneous blockchain systems must interact securely and efficiently. Addressing this problem requires unified cross-chain protocols and reliable mechanisms for transferring trust [23,24]. Existing solutions remain inadequate for complex IIoT scenarios. Many rely on a single communication path or a centralized intermediary and cannot meet industrial requirements for service reliability and real-time performance [25]. Their dependence on a single path also exposes cross-chain services to interruption after a communication failure. Traditional exchange protocols provide limited support for concurrent access across multiple chains [26]. The resulting performance bottleneck restricts large-scale cross-chain deployment in IIoT. Therefore, an effective IIoT cross-chain architecture must preserve blockchain-based trust while also supporting reliable and concurrent data access.

Against this background, this paper proposes a complete cross-chain data storage and sharing scheme for IIoT. The scheme responds to the preceding challenges at both the

storage-architecture and cross-chain-transaction levels. First, it distributes heterogeneous IIoT data across multiple blockchains according to their storage requirements. Distinct blockchains can therefore manage different data types instead of placing all storage and processing demands on one chain. Second, the system architecture specifies the collaborative relationships and data flows among the device, storage, and blockchain layers. It also defines the interactions required for storage, on-chain registration, authorization, and cross-chain retrieval. Finally, the CPHTLC contract algorithm secures cross-chain access through concurrent multi-path transaction execution. Parallel execution allows a transaction to proceed through multiple communication paths rather than relying on a single channel. The principal contributions are summarized as follows:

- We formulate a three-layer, multi-chain IIoT storage model that links each region's blockchain records to data retained in servers and databases. The model specifies how a resource's hash, storage location, and access conditions are registered and used to locate off-chain content across chains.
- We define the data structures and interaction sequence for local and cross-chain access, covering terminal submission, on-chain metadata registration, access-control checks, server-side retrieval, and the recording of access outcomes.
- We propose a concurrent-path hash time-locked contract (CPHTLC) procedure for paid cross-chain data access. The procedure coordinates commitment and redemption across participating chains, and its transaction behavior, system throughput, and transaction success rate are examined through simulation.

2. Related Work

As mentioned in Section 1, IIoT faces multiple challenges in data sharing, including trust deficits of centralized storage, performance bottlenecks of single-chain architectures, and reliability issues in cross-chain interaction. To tackle these challenges, existing research primarily focuses on three dimensions: enhancing data security, designing trusted storage architectures, and cross-chain data operations. This section will review representative works addressing the above issues sequentially, analyzing their contributions and limitations.

In enhancing data security, researchers primarily employ technologies such as encryption, authentication, and privacy computing to fortify the protection of various stages in the IIoT data lifecycle. For example, Arachchige et al. proposed a unified identification and authentication framework that strengthens identity management for heterogeneous IIoT devices [27]. The framework reduces attack risks by providing a common identity-management mechanism across device types. Liu et al. developed a secure multiparty computation protocol based on N-truncated polynomial ring encryption and coordinate mapping [28]. The protocol supports privacy-preserving set intersection and union for real-time IIoT data without universal restrictions. Ahmed et al. proposed a joint optimization algorithm for secure mobile edge computing resources in 6G IIoT [29]. Dynamic coordination of computing resources protects data while providing a secure offloading method for IIoT terminals. Huang et al. used generative adversarial imitation learning to train intelligent agents [30]. These agents proactively detect and defend against the leakage of private data. Although these methods strengthen security at individual stages, they do not establish trusted traceability across the complete data-use process. This remaining limitation shifts the research focus from local protection to trustworthy storage and source verification.

To address trustworthiness challenges, blockchain technology, with its distributed and tamper-proof characteristics, has been introduced into IIoT data management. For instance, Li et al. designed a designated-verifier aggregate signature scheme based on permissioned

blockchains [31]. The scheme provides controllable data verification while protecting privacy. Tan et al. combined blockchain with virtual-real data fusion for secure data sharing in manufacturing IoT [32]. Their resource assistance mechanisms and polygonal semantic rules protect resource-constrained nodes without conventional encryption. Chen et al. integrated blockchain and edge computing in a trusted task scheduling architecture [33]. Negotiated game smart contracts and configurable nodes provided incentive compatibility and trusted data exchange in distributed IIoT environments. The architecture also maintained data immutability and supported reliable verification. Yao et al. combined smart contracts, fine-grained operational control, and reputation penalty mechanisms to establish a trusted IIoT environment [34]. These components jointly provided security and privacy guarantees for system operation. Xia et al. used multiple edge servers and group signatures in a lightweight blockchain IoT service architecture [35]. Anonymous authentication and auditing reduced computation and communication overhead while preserving the trustworthiness of IIoT data. Despite these advances in trustworthiness, most solutions remain based on a single chain. Their throughput and storage-efficiency bottlenecks create the need for trusted interaction across multiple blockchain systems.

To achieve trusted data circulation among heterogeneous blockchain systems in IIoT, cross-chain technology has become a current research hotspot. For example, Yang et al. combined fully homomorphic encryption with a cross-chain architecture in a publicly auditable sealed-bid auction protocol [36]. The protocol supports privacy-preserving data exchange and trusted transaction verification in IIoT environments. Duan et al. used fuzzy commitment and physically unclonable functions in a cross-domain privacy-preserving authentication scheme [37]. These mechanisms support both authentication and data consistency across IIoT domains. Yang et al. designed an asynchronous sidechain scheme based on delayed aggregate signatures for trusted collaboration in multi-domain networks [38]. The scheme targets secure data exchange under asynchronous communication conditions. Zhang et al. introduced time-aware cross-chain data migration with collective control and execution storage mechanisms [39]. Their method supports reliable migration and trusted traceability of historical data in multi-chain environments. Ranathunga et al. used verifiable penalty-reward mechanisms in a blockchain-based hierarchical decentralized federated learning framework [40]. This design preserves data trustworthiness during cross-organizational collaboration. Wang et al. proposed a lightweight blockchain cross-domain data sharing scheme based on re-encryption [41]. Barbàra and Schifanella [42] combined HTLCs with multiparty computation and threshold signatures to coordinate token exchange among multiple participants. Liu et al. [43] proposed LightPay, which uses adapter signatures to support atomic off-chain multipath payments while preserving the unlinkability of partial payments. Zhang et al. [44] developed DeCross for IIoT cross-chain collaboration, combining auditable consensus with parallel processing on CPUs and GPUs. These studies address payment-channel coordination, multiparty settlement, and cross-chain consensus, respectively. However, cross-chain capability alone does not guarantee reliable concurrent access. Many solutions still depend on centralized intermediaries or single communication paths and therefore retain single-point-of-failure risks. Their limited concurrency also constrains real-time performance and scalability in IIoT scenarios.

In summary, existing solutions mostly focus on a specific aspect of IIoT data sharing, failing to form a collaborative mechanism ensuring high throughput, strong trustworthiness, and efficient cross-chain interoperability. Table 1 summarizes the main features of the related studies and the proposed scheme. Based on this, this paper proposes a blockchain-enabled cross-chain data storage and sharing scheme. It constructs a multi-chain distributed storage model to handle the heterogeneity of IIoT data and proposes the

Concurrent Path Hash Time-Locked Contract algorithm to ensure security while supporting high-concurrency cross-chain data sharing.

Table 1. Comparison of related studies.

Studies	Scope	Method	Chain Model	Object	Main Function
[27–30]	Data privacy	Encryption	–	IIoT data	Secure processing
[31–35]	Trusted management	Blockchain	Single-chain	Data records	Verification and sharing
[36,40]	Domain collaboration	Encryption and blockchain	Single-chain	Bids or models	Exchange and aggregation
[37–39,41]	Data sharing	Blockchain and access control	Single-chain	Data and metadata	Controlled exchange
[42–44]	Atomic exchange	Hash time locks	Cross-chain	Digital assets	Conditional settlement
This study	IIoT storage and access	Hash time locks	Cross-chain	IIoT data	Storage and authorized retrieval

3. System Model

Figure 1 presents the blockchain-based cross-chain storage and sharing model for IIoT data. The model comprises three layers, beginning with the IIoT device layer of heterogeneous IIoT devices. Their set is denoted by $TTE = \{TE_1, TE_2, \dots, TE_j, \dots, TE_{m \times n}\}$, where m is the number of blockchains and n is the number of devices covered by each blockchain. Here, TE_j denotes an IIoT data-acquisition terminal, such as an equipment sensor, an environmental sensor, or an inspection camera. These terminals collect measurements, equipment-status records, or images and send the data to a server for database storage. The associated blockchain registers the data hash, storage location, and access conditions. An authorized terminal on another chain can then request the stored data through the cross-chain access procedure. Every terminal requires a globally unique identifier so that the blockchain can associate submitted data with its source. The public identity of terminal TE_i is defined as $ID_i = \{PK_{d_i}, Cert_{d_i}\}$, where PK_{d_i} is the public key and $Cert_{d_i}$ is the certificate binding that key to the terminal identity. The corresponding private key SK_{d_i} is retained locally by the terminal and used to generate signatures. Recipients authenticate signed messages by validating the certificate and verifying the signature with PK_{d_i} . Together, these values bind a terminal identity to its blockchain address, signature, and certificate. The terminal record (TR) is the data package created by terminal TE_i and sent to the selected storage server for off-chain storage, denoted as

$$TR = \{ID_i, IN_j, sig_i, hash, tp, data\}, \quad (1)$$

where IN_j indicates that terminal TE_i belongs to blockchain BC_j . The term sig_i is the signature generated with private key SK_{d_i} , and $hash$ is the data hash. In addition, tp records the generation timestamp, and $data$ contains the data generated by terminal TE_i . Its on-chain registration request is represented as

$$req = \{ID_i, IN_j, sig_i, hash, tp, size\}, \quad (2)$$

where $size$ denotes the size of the data.

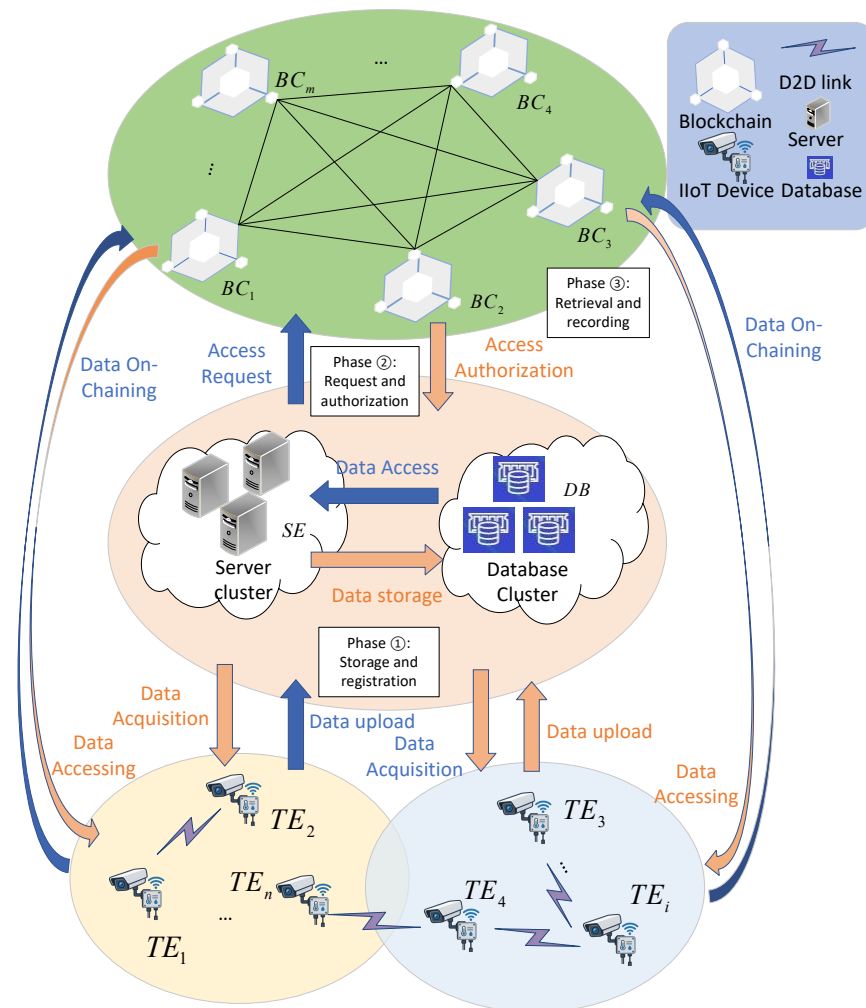


Figure 1. Blockchain-based cross-chain storage and sharing system model for IIoT data.

In the first phase, IIoT terminals upload the collected data to servers for storage in databases. The data hashes and storage information are registered on the corresponding blockchains. And in the second phase, a terminal submits an access request to its local blockchain. For cross-chain access, the request is forwarded to the target blockchain, which verifies access permissions and authorizes the corresponding server. At last, in the third phase, the authorized server retrieves the requested data from the database and returns them to the terminal. The access result is recorded on the participating blockchains.

We consider equipment condition monitoring and visual inspection across two production areas as a representative industrial process. In the first area, equipment sensors collect operating measurements and status records, while inspection cameras capture images of inspected items. A local server stores these data in a database, and the blockchain responsible for the area registers their hashes, storage locations, and access conditions. When a terminal in the second area requests a record, its local blockchain forwards the request to the blockchain responsible for the data, which verifies the registered access conditions and returns the storage information. Upon authorization, the terminal retrieves the requested measurements or images from the server, and the access outcome is recorded on-chain.

The second layer provides data storage. The server and database sets are denoted by $SSE = \{SE_1, SE_2, \dots, SE_p\}$ and $SDB = \{DB_1, DB_2, \dots, DB_q\}$, respectively. The resource capacity of the k -th server is $RRC_k = \{RSC_k, RCR_k, RCOR_k\}$. In this expression, RSC_k , RCR_k , and $RCOR_k$ denote its storage, computing, and communication resources, respectively.

The third layer is the blockchain layer, represented by $SBC = \{BC_1, BC_2, \dots, BC_s\}$. It occupies the central resource-management role in the system. In particular, this layer supports on-chain registration, data storage, and data queries initiated by IIoT terminals. The blockchain also manages the server and terminal resources associated with secure data storage and access. Terminals with sufficient storage capacity may additionally act as storage resources rather than relying exclusively on servers. Devices covered by the same blockchain communicate through D2D links. Terminals and servers in the same coverage area use Orthogonal Frequency Division Multiplexing to avoid mutual interference.

Interactions among blockchains support cross-chain storage and sharing requests, and both the request process and outcome are recorded on-chain. Each blockchain covers a specific IIoT region and maintains information about its devices and their data-storage locations. The data content itself remains in servers and databases rather than being stored directly on the blockchain. Servers collect data from IIoT devices, format the content, and store it in the appropriate database. They then register the data hash and related storage information on the blockchain platform. For cross-chain access, an IIoT terminal first submits a request to its local blockchain. The local blockchain queries its records to identify the chain containing the target data and forwards the request to that chain. The target blockchain searches its on-chain records for the storage location and related metadata. It returns this information through the request path, after which the terminal accesses the data from the corresponding storage resource.

4. System Design

This section specifies the design derived from the system model. Standardized data formats first provide interoperability among heterogeneous components. The subsequent subsections describe on-chain registration, single-chain storage and access, and the request and verification sequences for cross-chain scenarios. Finally, the Concurrent Path Hash Time-Locked Contract (CPHTLC) algorithm addresses the security and atomicity of cross-chain transactions.

4.1. Data Format Design

Cross-chain storage and access require consistent metadata, smart-contract interfaces, and cross-chain message formats across IIoT terminals and blockchain platforms. Servers and databases store the data content, while blockchains maintain the metadata used to identify resources, locate stored content, and verify access conditions. Standardized metadata supports these operations, and consistent contract interfaces specify the parameters for registration, queries, and access requests. Defined cross-chain messages enable participating blockchains to process requests and associate them with the corresponding responses. The storage server handles the acquired content according to its data format. The on-chain registration request from an IIoT terminal uses *req* (Equation (2)). Cross-chain interactions use a standard structure containing relevant information about both transaction parties, as summarized in Table 2:

Table 2. Cross-chain request data format.

Symbol	Parameter Description
Ac_{id}	Unique identifier of the requesting blockchain
Ce_{id}	Unique identifier of the responding blockchain
tp	Cross-Chain Data Transaction Timestamp
$T_{confirm}$	Data Confirmation Receipt Identifier

The format in Table 2 is represented as

$$BBT = \{Ac_{id}, Ce_{id}, tp, T_{confirm}\}. \quad (3)$$

Here, Ac_{id} and Ce_{id} identify the requesting blockchain and the responding blockchain responsible for the requested resource, respectively.

After a database stores the data, the associated storage information must be registered on the blockchain. This registration links the off-chain content to an immutable on-chain record without placing the complete data on the chain. The uploaded metadata identifies the resource, its storage location, its hash, and the conditions governing access. Table 3 defines the complete upload structure:

Table 3. Metadata entries in the data-upload structure.

Metadata Item	Description	Metadata Item	Description
<i>Identifier</i>	Unique identifier of the resource	<i>Type</i>	Type of the resource
<i>Title</i>	Name of the resource	<i>Language</i>	Language of the resource's knowledge content
<i>Creator</i>	Primary creator of the resource content	<i>Source</i>	Source of the resource
<i>Subject</i>	Description of the resource's subject	<i>Storage</i>	Storage location of the resource
<i>Description</i>	Description of the resource content	<i>Format</i>	Physical or digital manifestation of the resource
<i>Contributor</i>	Other contributors to the resource	<i>Coverage</i>	Scope of application of the resource
<i>Rights</i>	Copyright owner of the resource	<i>Price</i>	Transfer price of the resource
<i>Date</i>	Resource generation time	<i>Relation</i>	Index relationship with other resources
<i>Publisher</i>	Name of the entity that may publish the resource	<i>Hash</i>	Hash of the original data
<i>AccessControl</i>	Cross-chain access control	<i>AccessLevel</i>	Access permission level
<i>AllowedChains</i>	Chains allowed to access	<i>AccessPolicy</i>	Access policy

Identifier is the unique identifier of a digital resource, such as a URI, URL, DOI, ISBN, or ISSN. *Storage* identifies the server holding the resource, whereas *Format* specifies its media type or storage capacity. *Price* records the transfer price and is set to 0 for a free resource. *Creator*, *Subject*, *Description*, *Contributor*, and *Publisher* describe the resource and the entities associated with it. *Date* records its generation time, while *Hash* binds the metadata entry to the original data. The remaining fields follow the descriptions in Table 3. Compared with single-chain storage, cross-chain storage additionally requires *AccessControl*, *AllowedChains*, *AccessPolicy*, and *AccessLevel*. *AllowedChains* indicates whether cross-chain access is permitted and lists the authorized blockchain names or identifiers. A value of “Yes” permits such access, whereas “No” denies it. *AccessLevel* defines permissions such as “Read-Only”, “Read-Write”, and “Full Access”. *AccessPolicy* specifies restrictions based on time, designated nodes, or access frequency. Together, these fields express both whether cross-chain access is allowed and how an authorized requester may use the resource.

Data submitted by IIoT terminals or servers requires a unified exchange format so that the blockchain platform can parse metadata and validate signatures. Figure 2 presents the structure designed for interactions between IIoT terminals and the blockchain. The complete structure contains fields for several advanced functions used during contract deployment and upgrade. Only a subset is required for the registration procedure considered here. On-chain metadata registration primarily uses the Payload and EndorsementEntry sender modules.

The Payload module contains the basic fields required for on-chain metadata registration. These include the chain identifier *chain_id*, transaction type *tx_type*, globally unique transaction identifier *tx_id*, and transaction timestamp. It also records *contract_name*, transaction parameters, the invoked contract method, and the gas limit. The sender module describes the transaction initiator through *org_id*, *member_type*, *member_info*, and the member's transaction signature.

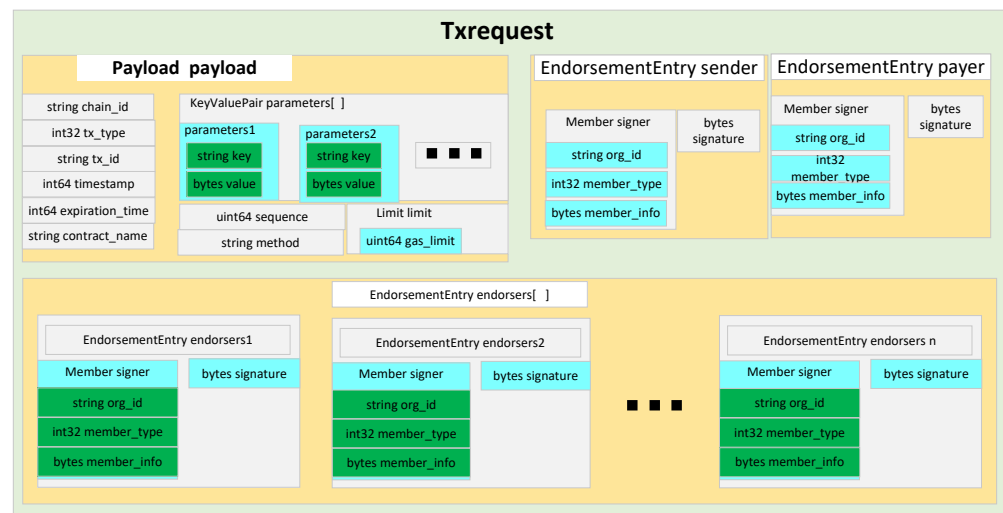


Figure 2. Data exchange format between an IIoT terminal and the blockchain platform.

4.2. Single-Chain Data Upload and Application Process

Data generated by an IIoT terminal are stored in a database, while the corresponding record is registered on the blockchain platform. Separating content storage from metadata registration reduces the volume of data maintained directly on-chain. The blockchain nevertheless retains the information required to locate, authenticate, and authorize access to the stored content. Figure 3 illustrates the single-chain sequence for data storage, upload, and usage applications.

Data storage begins with smart-contract deployment. A smart contract is a computer protocol that propagates, verifies, or executes contractual operations without a third party. Its execution allows trusted transactions to proceed according to predefined logic. The resulting transactions are traceable and irreversible. The dedicated contract in this system supports on-chain data registration for IIoT terminals and provides the associated registration and query functions. Its data structures reflect the metadata characteristics of IIoT resources. Contract design requires both the functional methods and the parameter structures expected by those methods. Compared with a single-chain contract, this design additionally provides a method for cross-chain access. The principal methods are as follows:

1. **Metadata Registration Method Evidence:** Its parameters include the globally unique ID assigned to an on-chain metadata registration transaction and the hash of the metadata structure. The ID uniquely identifies the transaction within the blockchain system. The hash is also unique and represents the complete metadata structure. The remaining parameters contain the metadata content to be stored on the blockchain platform.
2. **Hash and ID Checking Methods ExistsOfHash, ExistsOfId:** These methods determine whether the ID and hash supplied during on-chain metadata registration already exist. This check prevents a new registration from reusing identifiers already recorded by the contract. Existing values require reconstruction before registration proceeds.
3. **Metadata Query Method Find:** Both the hash and ID uniquely identify a blockchain transaction. A metadata or related-transaction query therefore requires either value. Find comprises FindByHash for hash-based retrieval and FindById for identifier-based retrieval.
4. **Cross-chain Access Method cross-chain:** An IIoT terminal invokes this method if the requested resource is registered on another blockchain. The method establishes interaction with the target blockchain and retrieves information about the requested data. It then determines whether the target resource is accessible under the applicable policy.

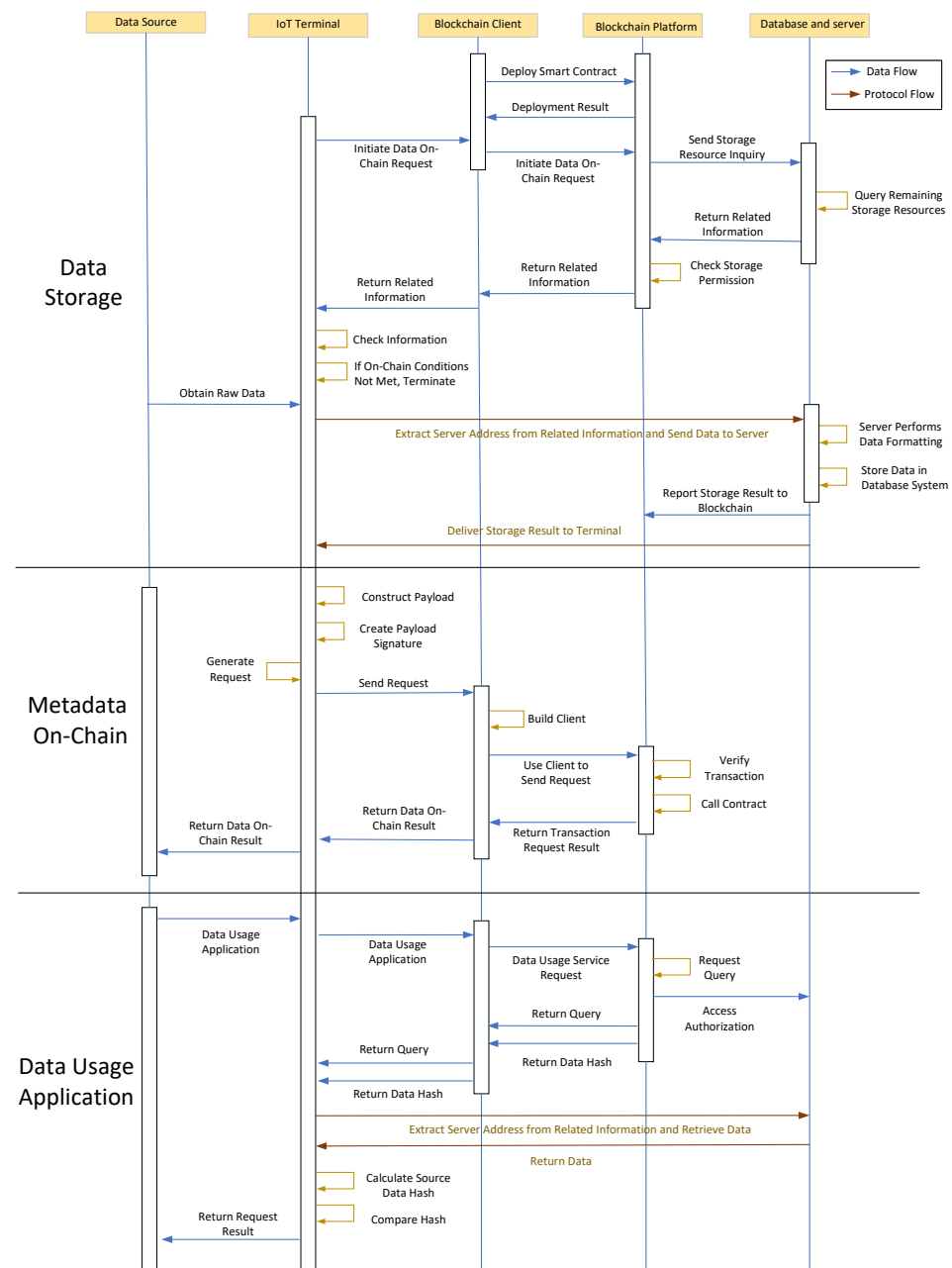


Figure 3. Sequence diagram for data storage, upload, and usage applications.

The contract also provides auxiliary methods and batch operations for managing multiple metadata records. Table 4 lists the principal methods, parameters, return types, and functions.

The smart contract is deployed on the blockchain system and remains active until explicitly terminated. Resource providers use its methods to upload metadata to the chain. Consumers invoke the query methods to obtain that metadata and locate specific data content. The same contract therefore supports both resource publication and subsequent discovery. The IIoT terminal submits an on-chain data registration request req (Equation (2)) to the blockchain. The request specifies the data size required for storage allocation. The blockchain then queries the servers for their remaining storage resources. Each server compares the requested size with its available capacity and returns the relevant resource information. Based on these responses, the blockchain platform selects the applicable result and reports it to the IIoT terminal. The terminal evaluates the returned result before continuing the storage process. A successful response allows it to extract the selected

server address and transmit data TR for storage. The server formats the data according to its storage requirements and writes the content to the database. It then returns the storage result to both the blockchain platform and the terminal. The returned storage result initiates on-chain registration at the IIoT terminal. The terminal constructs the Request body, signs the payload with its private key, and serializes the message. These operations bind the registration message to the terminal identity and prepare it for transmission. The terminal then submits the Request body to the blockchain platform for signature verification. A failed verification prevents registration, whereas successful verification causes the platform to record the data storage information on-chain.

Table 4. Smart contract methods and parameters.

Method Name	Parameters	Return Type	Description
Evidence	$Id, hash, metadata$	error	Metadata upload
ExistsOfHash	$hash$	Bool, error	Check hash existence
ExistsOfId	id	Bool, error	Check ID existence
FindByHash	$Hash$	Evidence, error	Query by hash
FindById	id	Evidence, error	Query by ID
EvidenceBatch	$evidences$	error	Batch metadata upload
UpdateEvidence	$Id, hash, metadata$	error	Update metadata

For single-chain data use, the IIoT terminal first sends a request to the blockchain. The blockchain queries the storage information associated with the requested resource. It returns this information to the terminal and simultaneously sends an access-authorization instruction to the server. From the returned record, the terminal extracts the address of the server containing the data. It then sends a data request directly to that server. The server compares the request with the authorization information received from the blockchain. A mismatch causes the server to deny access. If the two are consistent, the server retrieves the data from the database and returns them to the terminal.

4.3. Cross-Chain Access Process

Cross-chain access is required if the requested data are not registered on the IIoT terminal's local blockchain. In this case, the local chain must identify and communicate with the blockchain responsible for the target resource. The procedure also coordinates authorization between the two chains and the storage server. Figure 4 presents the corresponding access sequence.

As shown in Figure 4, a data requester first sends a data-use application to terminal TE_i . The terminal prepares a data-use request, denoted by $dreq$, for its local blockchain client. The request has the following structure:

$$dreq = \{ID_i, IN_j, sig_i, type_{data}, tp\}, \quad (4)$$

where sig_i is the signature of IIoT terminal TE_i on the submitted request, and $type_{data}$ identifies the requested data type. The blockchain client parses the fields in $dreq$ and forwards the structured application to the blockchain platform. The platform validates the request and rejects any invalid submission. For a valid request, it queries the relevant metadata to identify the blockchain responsible for the requested data. The resulting record contains the target-chain information required for subsequent communication. The platform returns this information and the query result to the IIoT terminal. After identifying the blockchain

holding the requested data, the local blockchain sends an inter-blockchain data-use request, denoted by $ptreq$, to that blockchain. The request has the following structure:

$$ptreq = \{ID_{pbc}, ID_{tbc}, type_{data}, sig_{pbc}, N_{seq}, tp\}, \quad (5)$$

where ID_{pbc} and ID_{tbc} identify the original and target blockchains, respectively. The original blockchain signs the requested data content with sig_{pbc} , and N_{seq} is the request sequence number.

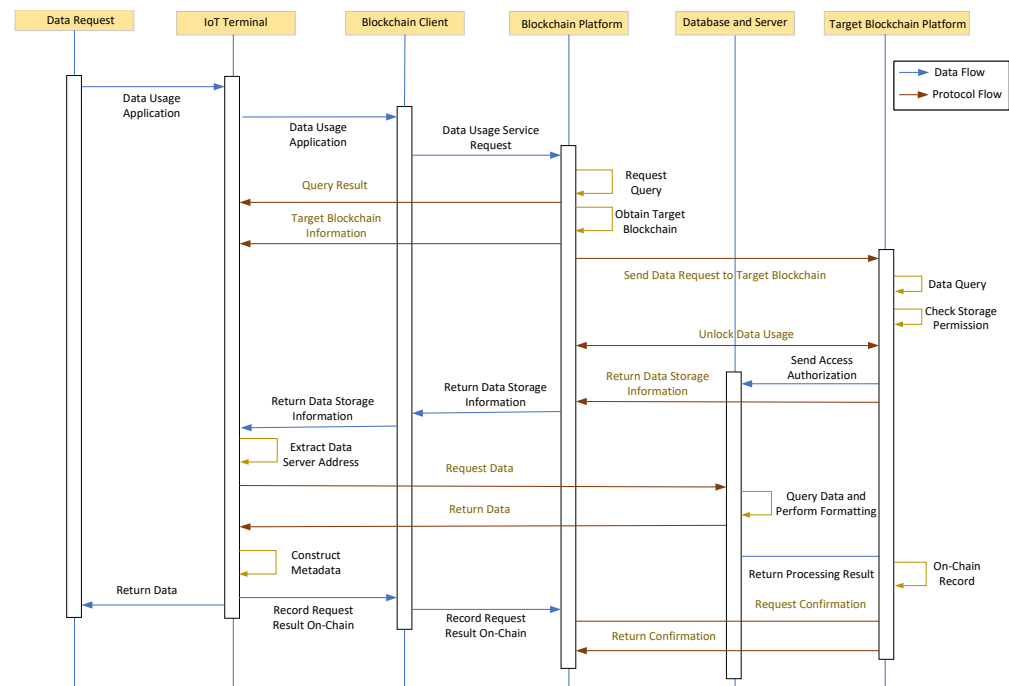


Figure 4. Cross-chain data request sequence diagram.

The target blockchain queries its records after receiving the data usage request. Once the data are located, it checks the access permissions recorded for that resource. A policy that prohibits access causes immediate rejection. Otherwise, the target blockchain compares the required access level with the level assigned to the original blockchain. An insufficient level also results in rejection and terminates further processing. If the level is sufficient, the target blockchain initiates the data-unlocking procedure. It then extracts the storage address and related information from the queried metadata. Finally, it sends the following authorization information to the server holding the data:

$$tsreq = \{ID_{pbc}, ID_{tbc}, type_{data}, sig_{tbc}, N_{seq}, V_{code}, tp\}, \quad (6)$$

where sig_{tbc} is the target blockchain's signature for the requested data type $type_{data}$, and V_{code} is the access-verification code. The server later uses this code when validating the terminal's data request. At the same time, the target blockchain returns the storage information to the original blockchain in the following structure:

$$tpreq = \{ID_{pbc}, ID_{tbc}, type_{data}, sig_{tbc}, N_{seq}, V_{code}, addr_{data}, tp\}, \quad (7)$$

where $addr_{data}$ is the data storage address. Using sequence number N_{seq} , the original blockchain matches this response to the corresponding request. It extracts V_{code} and $addr_{data}$ from $tpreq$ and passes them to the blockchain client. The client forwards these values to the

IIoT terminal. The terminal combines its identity and signature with the target-chain and storage information. It then constructs the following request for the server:

$$dsreq = \{ID_i, IN_j, ID_{tbc}, type_{data}, sig_i, V_{code}, tp\}. \quad (8)$$

The IIoT terminal sends this request to the data storage server at $addr_{data}$. The server verifies the terminal signature, verification code, and related request information. A successful check permits the server to query the database and return the requested data in the following structure:

$$sreq = \{ID_i, IN_j, ID_{tbc}, data, sig_s, V_{code}, tp\}, \quad (9)$$

where $data$ contains the data requested by the IIoT terminal, and sig_s is the storage server's signature on that content. The signature allows the terminal to verify the origin of the returned data. The server simultaneously reports the request-processing result to the target blockchain.

After verifying the returned information and server signature, the IIoT terminal delivers the requested data to the data requester. It then constructs an on-chain record of the request result and submits it to the blockchain client. The record preserves the outcome of the cross-chain access procedure for subsequent tracing. The client validates this record and forwards it to the blockchain platform. The platform subsequently sends confirmation request BBT to the target blockchain. The target blockchain validates the confirmation request against its local transaction information. Following successful verification, it returns a confirmation message and completes the recorded interaction.

4.4. CPHTLC Data-Unlocking Algorithm

In a multi-blockchain system, some cross-chain data become available only after payment and therefore require a transaction process [45]. The proposed CPHTLC algorithm extends the conventional hash time-locking mechanism for this purpose. Whereas an HTLC primarily supports transactions between two blockchains, CPHTLC coordinates transactions involving multiple chains. It terminates only after all associated cross-chain operations have completed. This design preserves consistency and atomicity, secures paid-data transactions, and prevents unilateral loss. The following analysis first introduces HTLC and then describes its CPHTLC extension.

A Hash Time-Locked Contract (HTLC) is widely used for cross-chain transactions and atomic swaps. It combines a cryptographic hash lock with a time lock to preserve transaction atomicity and security. Both parties share a hash value that defines the unlocking condition. To obtain the transaction content, the recipient must reveal a matching secret before a specified deadline. Failure to do so automatically returns the locked assets to the sender. The mechanism thus protects both parties without a trusted intermediary. Figure 5 illustrates the HTLC transaction process.

The HTLC transaction includes the following steps:

1. The source blockchain BC_i sends a transaction request, and the target blockchain BC_j returns a request acknowledgment.
2. The source blockchain BC_i generates a random secret s and sets the transaction deadline t_1 . It then computes the following hash value:

$$H = \text{hash}(s). \quad (10)$$

The source blockchain sends H and t_1 to BC_j and uses them to lock the assets of user TE_p . The assets can be unlocked only if s is supplied before t_1 ; otherwise, they are returned to the account of TE_p .

3. User TE_q receives H and t_1 and locks its assets using H and a second deadline t_2 . The deadlines satisfy the following condition:

$$t_1 > t_2. \quad (11)$$

The assets of TE_q can be unlocked only if s is provided before t_2 ; otherwise, the assets are returned to the account of TE_q .

4. User TE_p reveals s on target blockchain BC_j to unlock the assets of TE_q . With the revealed secret, TE_q unlocks the assets of TE_p on source blockchain BC_i . Each user consequently obtains the counterparty's assets, completing the atomic exchange.

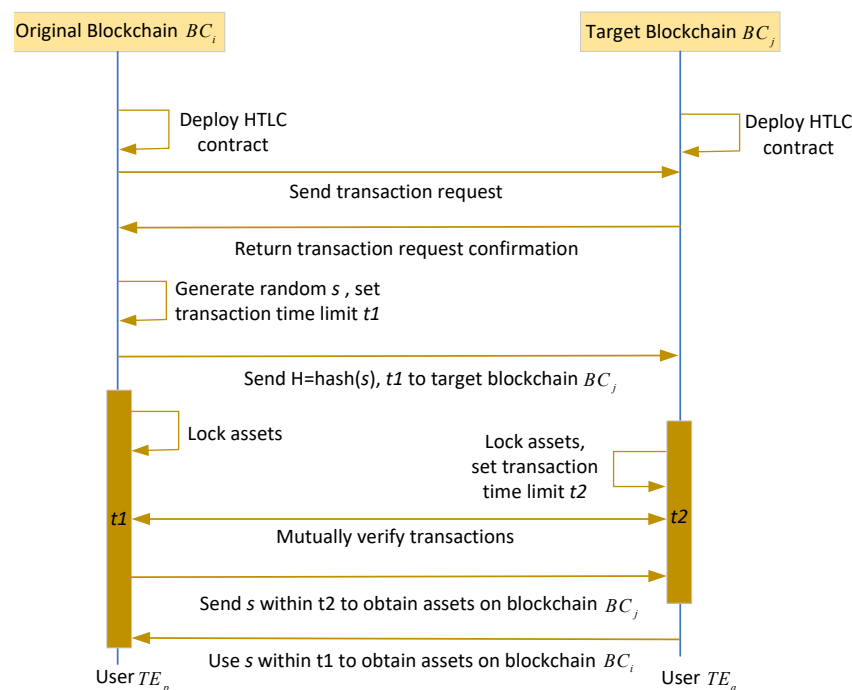


Figure 5. HTLC transaction process.

The hash-locking mechanism permits an asset or subsequent operation to be unlocked only with a secret that satisfies the committed hash. The one-way and collision-resistant hash function makes recovery of this secret computationally infeasible. Time locks additionally preserve atomicity by requiring both parties to complete the transaction within predefined periods. This constraint prevents a dishonest participant from postponing settlement indefinitely. Nevertheless, a conventional HTLC connects only two parties. Multi-user transactions require a separate contract for each pair, so the number of contract operations grows with participation. This overhead reduces efficiency in a multi-chain environment. CPHTLC instead executes payments and data transactions concurrently over multiple paths. Other paths can complete a transaction if one channel fails, improving both success rate and robustness. Multi-path execution also obscures the complete flow of funds from individual participants and thereby provides additional transaction privacy.

Let the participant sets of blockchains BC_i and BC_j be defined as follows:

$$P_{BC_i} = \{TE_1, TE_2, \dots, TE_n\}, \quad (12)$$

$$P_{BC_j} = \{TE_1, TE_2, \dots, TE_m\}. \quad (13)$$

During data access, members of either set may act as requesters or providers. The global participant set is represented as follows:

$$P = \{P_{BC_1}, P_{BC_2}, \dots, P_{BC_i}, \dots, P_{BC_j}, \dots\}. \quad (14)$$

CPHTLC consists of three stages:

1. Preprocessing stage: Through multiparty computation, the participants jointly generate a shared secret s and use $H = \text{hash}(s)$ as the locking condition. A threshold-signature protocol produces an aggregate public key. A smart contract is then deployed on each participating blockchain to manage funds and protocol logic.
2. Commitment stage: The participants transfer funds to a pay-to-script-hash address derived from the aggregate public key. The protocol proceeds only after every participant has locked the required funds, thereby limiting malicious behavior.
3. Redemption stage: The participants redeem the assets on the participating blockchains by revealing the preimage of the shared secret. The execution order is

$$P_{BC_1} \rightarrow P_{BC_2} \rightarrow \dots \rightarrow P_{BC_i} \rightarrow \dots \rightarrow P_{BC_j} \rightarrow \dots. \quad (15)$$

CPHTLC secures paid data access while overcoming the two-blockchain restriction of a conventional HTLC. It allows an IIoT terminal to access related data concurrently across multiple blockchains. The flow of the CPHTLC algorithm is shown in Algorithm 1.

Algorithm 1 Concurrent Path Hashed Time-Lock Contract (CPHTLC)

Input: Participants $P = \{P_{BC_1}, \dots, P_{BC_n}\}$; Assets $A = \{a_{BC_1}, \dots, a_{BC_n}\}$; Time-lock sequence $T = \{t_1, \dots, t_n\}$.

Output: SUCCESS if all assets are redeemed; otherwise the confirmed settlement status of each contract.

```

1: Participants jointly generate  $s$  and compute  $h = H(s)$ ;
2: Generate  $PK_{agg}$  using the specified threshold-signature policy;
3: for  $i = 1$  to  $n$  do
4:   Deploy the contract on  $BC_i$  with  $h, t_i, PK_{agg}$ , the beneficiary, and the refund address;
5: end for
6: parallel for  $i = 1$  to  $n$  do
7:    $P_{BC_i}$  submits a transaction to lock  $a_{BC_i}$  under the contract conditions;
8: end parallel for
9: if all commitments are confirmed with sufficient time for redemption then
10:  Disclose the shared preimage  $s$ ;
11:  Concurrently, participating clients monitor and relay the disclosed preimage;
12:  Obtain  $s$  from a confirmed contract disclosure or another participating client;
13:  Verify the transaction association and  $H(s) = h$ , then forward  $s$  to the remaining clients;
14:  for  $i = 1$  to  $n$  do
15:    if  $s$  is available,  $H(s) = h$ , and  $a_{BC_i}$  remains redeemable before  $t_i$  then
16:      Submit  $s$  to redeem  $a_{BC_i}$  to the beneficiary recorded in the contract;
17:      Check confirmation before proceeding, or record the unresolved state at expiry;
18:    end if
19:  end for
20: end if
21: for  $i = 1$  to  $n$  do
22:  if  $a_{BC_i}$  remains locked and  $t_i$  has expired then
23:    Submit a refund of  $a_{BC_i}$  to the refund address recorded in the contract;
24:  end if
25:  Record the confirmed settlement state, retaining unresolved contracts for further monitoring;
26: end for
27: if all required redemptions are confirmed then
28:  return SUCCESS;
29: end if
30: return the recorded settlement states;

```

All participating contracts use the same preimage s and committed hash h . Each participating client obtains s from a confirmed disclosure in a participating contract or from another participating client and forwards it within the same transaction context. The receiving client checks the transaction association and verifies $H(s) = h$ against its contract. Participating clients continue observation and relay during redemption. The expiry schedule must accommodate these operations and the ordered redemption sequence. Completion assumes timely communication and confirmation, together with adherence to the preimage-release rule. A prolonged partition or premature disclosure may result in partial settlement. Refunds apply to funds remaining locked after expiry, while completed redemptions retain their confirmed state.

5. System Validation

5.1. Security Analysis

The proposed system distributes different data types across multiple blockchains, reducing the storage burden on any single chain. This arrangement allows storage resources to be assigned according to the requirements of each data type. The blockchain records also preserve traceability throughout local and inter-chain exchange. The principal security properties are as follows.

1. Authentication and access control: The receiver validates the sender's certificate and verifies the signature over the request identifiers, content, and freshness fields. For the cross-chain request in Equation (5), replay protection requires

$$0 \leq t_{recv} - t_p \leq \Delta t \wedge (ID_{pbc}, ID_{tbc}, N_{seq}) \notin \mathcal{R}. \quad (16)$$

Here, t_p and t_{recv} denote the request and reception times, Δt is the acceptance window, and $\mathcal{R}$ stores accepted-request identifiers. This condition assumes synchronized clocks, unique sequence numbers for each source–target chain pair, and retention of request records throughout the acceptance window. Following authentication and authorization, duplicate checking and request registration are performed atomically to prevent repeated execution. The storage server releases data according to the owner's access policy and blockchain authorization. Compromise of the server or an authorized recipient remains a potential source of disclosure.

2. Data integrity: The terminal verifies retrieved content against its registered on-chain hash. Under collision resistance and the integrity of the confirmed blockchain record, a matching hash establishes consistency with the registered data, while a mismatch results in rejection. Measurement accuracy depends on the acquisition device.
3. Cross-chain transaction security: CPHTLC confirms all required commitments before releasing the shared preimage. Contracts enforce redemption deadlines and permit refunds of unredeemed funds after expiry. Atomic completion requires secure preimage generation and release, together with sufficient time for ordered redemption. Premature disclosure through participant collusion may enable partial settlement. Liveness depends on continued transaction processing and submission of redemption or refund requests, so prolonged blockchain outages delay settlement and recovery. Successful data retrieval additionally requires availability of the off-chain storage service.

5.2. CPHTLC Algorithm Performance Analysis

This section evaluates the proposed CPHTLC algorithm for concurrent cross-chain data access and examines the performance of the complete multi-chain storage system. The implementation environment comprises the ChainMaker 2.0 series and a CentOS 7 guest system running on a Windows 11 host. The performance simulations use PBFT and

examine transaction time, transaction success rate, system throughput, and fault tolerance under different node counts, transaction volumes, and network delays. Table 5 summarizes the implementation environment and simulation parameters.

Table 5. Implementation environment and simulation parameters.

Parameter	Value
Blockchain platform	ChainMaker 2.0 series
Host operating system	Windows 11
Guest operating system	CentOS 7
Linux kernel	3.10, x86_64
Consensus mechanism	PBFT
Maximum number of blockchains	5
Maximum number of nodes per blockchain	170
Maximum network delay	200 ms
Maximum transactions per batch	100
Maximum node count in the transaction analysis	60
Maximum node count in the failure analysis	151

Figure 6 relates blockchain transaction time to transaction volume for several node counts. The horizontal and vertical axes indicate transaction volume and time, respectively, while color distinguishes each node-count configuration. The transaction volume ranges from 20 to 100. Transaction time rises with volume because larger batches require more total processing, with the increase becoming more pronounced at higher transaction volumes. The simulation randomly assigns transactions across multiple submitting nodes rather than using a single submission point. This allocation reduces congestion associated with concentrating all requests at one node. At a fixed volume, transaction time also increases with the node count because consensus generates more inter-node messages. Moreover, PBFT requires approval from more than two-thirds of participating nodes. A larger network therefore requires responses from more nodes before reaching a successful consensus decision. Too few nodes provide weak failure resistance because a small number of failures can disable the system. Conversely, an excessively large set increases consensus latency. Node selection must therefore balance fault tolerance against transaction time.

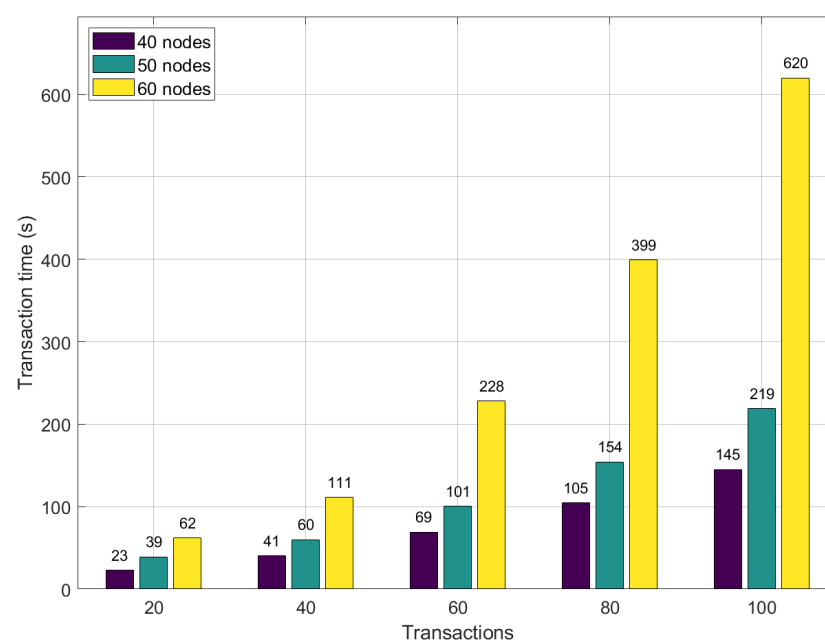


Figure 6. Relationship between transaction time and transaction volume under different node counts.

Figure 7 characterizes transaction success rate as a function of network delay for different node counts. Network delay and transaction success rate are plotted on the horizontal and vertical axes, respectively. Across all configurations, success rate decreases as either network delay or the number of nodes increases. Longer network delays extend the average time required for a node to process and propagate a transaction. An HTLC transaction fails if this process does not finish before the time lock expires. Accordingly, the success rate approaches zero as network delay approaches the contract deadline. Larger node sets further increase the time required for consensus. Once the combined network and consensus delay exceeds the time-lock threshold, the number of expired contracts rises sharply. A practical deployment must therefore select the time lock by jointly considering network delay and the number of consensus nodes.

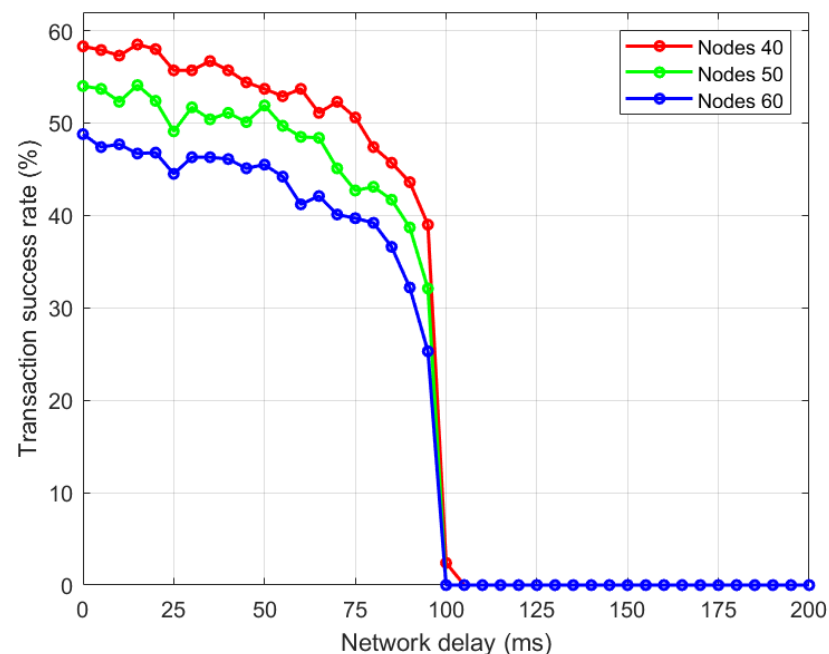


Figure 7. Relationship between transaction success rate and network delay under different node counts.

5.3. System Performance Analysis

HTLC-based control secures data storage and access across blockchains, although contract execution introduces additional latency. This overhead affects the number of transactions that the system can complete under different node and chain configurations. The following simulations therefore assess both throughput and fault tolerance for the complete multi-chain system.

Figure 8 plots system throughput against the number of nodes per blockchain for several chain counts. The horizontal axis gives the nodes per chain, while the vertical axis reports aggregate system throughput. Each curve represents a different number of blockchains. With the chain count fixed, throughput declines as the number of nodes per chain increases. A larger node set requires more communication during consensus and therefore finalizes fewer transactions per unit time. By contrast, at a fixed node count per chain, adding blockchains increases throughput through concurrent transaction processing. Multiple blockchains can process independent transactions in parallel instead of serializing all work on one chain. The proposed architecture can therefore benefit from additional chains, provided that cross-chain coordination overhead remains controlled.

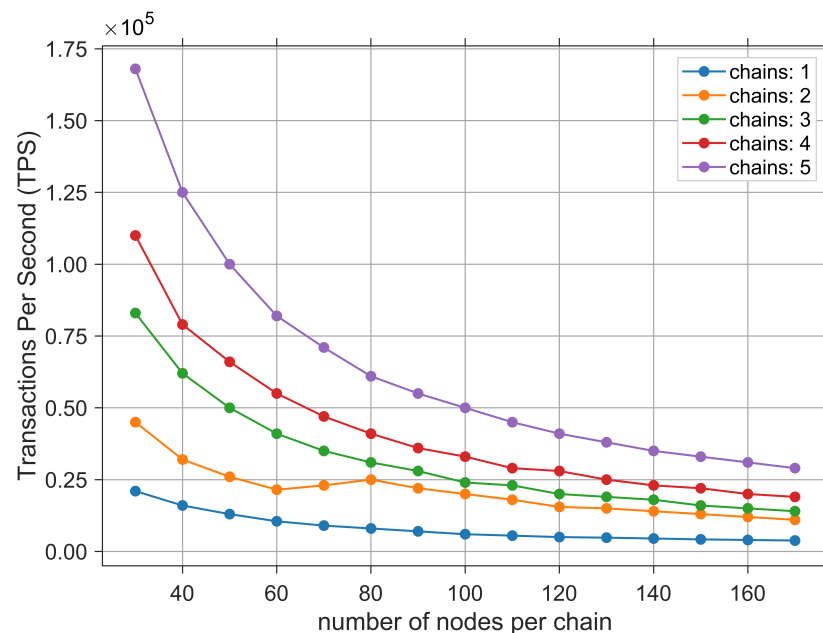


Figure 8. Relationship between system throughput and nodes per chain under different numbers of blockchains.

Figure 9 evaluates throughput as the number of failed nodes increases under different total node counts. The horizontal axis specifies failed nodes, and the vertical axis reports system throughput. Throughput declines with additional failures and reaches zero after failures exceed one-third of all nodes. This result follows the PBFT fault-tolerance bound because effective consensus cannot continue beyond that threshold. Before reaching the bound, systems with more total nodes exhibit lower throughput, consistent with Figure 8. The reduction reflects the additional time required for consensus among a larger set of participants. System design must therefore consider both the HTLC deadline and the PBFT configuration. The deadline must accommodate consensus and network delays without becoming so long that transaction completion is unnecessarily postponed.

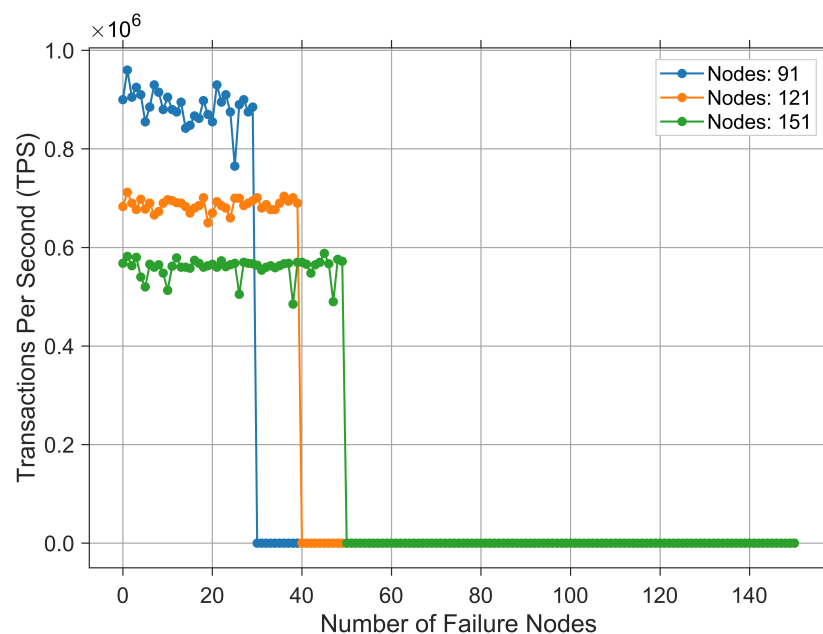


Figure 9. Relationship between system throughput and failed nodes under different total node counts.

To compare the performance of different contract algorithms, HTLC, P-HTLC, and CPHTLC are evaluated with 50 nodes per blockchain, 100 transactions, and two to five blockchains. The HTLC baseline executes bilateral transactions sequentially, whereas P-HTLC allows multiple independent bilateral HTLC instances to execute concurrently.

Figure 10 shows that the average transaction time decreases as the number of participating blockchains increases under the current experimental setting. This trend is mainly due to the increased processing resources and cross-chain parallelism rather than the number of blockchains itself. HTLC suffers from sequential execution and accumulated waiting delays, while P-HTLC reduces part of this dependency. CPHTLC further enables concurrent contract execution, resulting in the lowest transaction time. With five blockchains, CPHTLC reduces the average transaction time by about 26.4% and 12.8% compared with HTLC and P-HTLC, respectively.

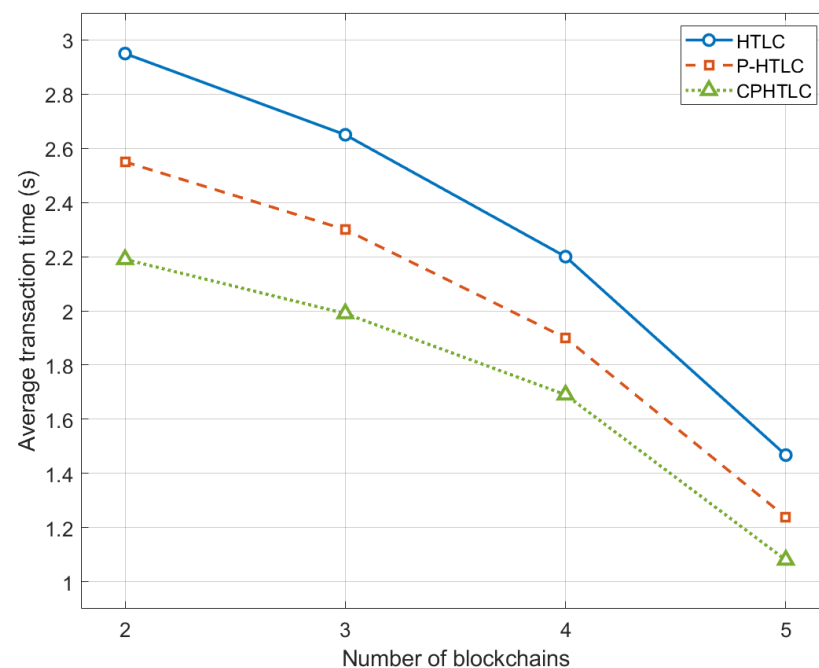


Figure 10. Average transaction time of different contract algorithms with varying numbers of blockchains.

Figure 11 shows that system throughput increases with the number of participating blockchains because more processing resources and parallel execution opportunities become available. However, the throughput increase gradually slows as the number of blockchains grows. The three schemes utilize these resources differently. HTLC is limited by sequential contract execution, P-HTLC improves parallelism to some extent, and CPHTLC can exploit concurrent cross-chain execution more effectively. As a result, the throughput advantage of CPHTLC becomes more pronounced as the number of blockchains increases. With five blockchains, CPHTLC improves throughput by about 38.2% and 16.0% over HTLC and P-HTLC, respectively.

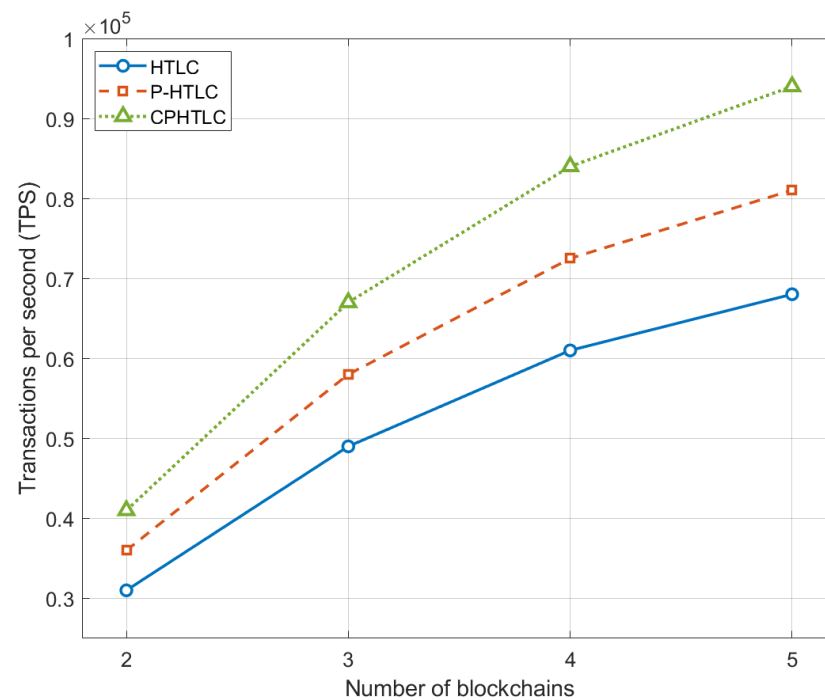


Figure 11. System throughput of different contract algorithms with varying numbers of blockchains.

6. Conclusions

This study developed a multi-chain cross-chain storage system for IIoT data. The system distributes different data types across blockchains and protects the authenticity, immutability, and traceability of their storage records. A CPHTLC-based access method uses concurrent paths to improve reliability and efficiency while preserving the atomicity of paid-data transactions. The architecture comprises device, data-storage, and blockchain layers. The device layer contains both data sources and data consumers. The data-storage layer stores and retrieves the complete content through its servers and databases. Multiple blockchains manage distinct data subsets and participate in cross-chain access. As the coordinating layer, the blockchain system controls storage allocation, metadata registration, authorization, and access verification. The interaction sequences cover data storage, on-chain registration, local access, and cross-chain access. Standardized transaction, metadata, authorization, and response structures support the complete data flow across heterogeneous terminals and blockchain platforms. These structures also provide a consistent basis for communication between the device, storage, and blockchain layers. CPHTLC extends the HTLC mechanism from asset and energy transactions to IIoT data storage and exchange. Simulations indicate that distributed multi-chain processing yields higher aggregate throughput than a single-chain system. Meanwhile, the failure threshold remains consistent with the one-third PBFT bound. The system can therefore increase the processing capacity of blockchain-based IIoT data management without reducing the fault tolerance of the underlying consensus mechanism. These findings support its use as a reference architecture for secure industrial data storage and sharing. Future work will evaluate cross-chain access latency in a prototype IIoT deployment, including inter-chain communication, authorization, CPHTLC execution, consensus, and interactions with local safety controllers for time-critical operations. It will also extend the current study to larger-scale IIoT scenarios with more nodes, participating chains, and transaction workloads.

Author Contributions: Conceptualization, Y.Z. and Y.W.; methodology, Y.Z. and Y.W.; software, Y.W.; validation, Y.W., J.L. and H.Z.; formal analysis, Y.W.; investigation, Y.W. and J.L.; resources, Y.Z. and H.Z.; data curation, Y.W.; writing—original draft preparation, Y.W.; writing—review and editing, Y.Z., J.L. and H.Z.; visualization, Y.W.; supervision, Y.Z.; project administration, Y.Z.; funding acquisition, Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: The work was supported in part by the National Natural Science Foundation of China (NSFC) Youth Fund (62301058) and in part by Sub Project of National Key Research and Development plan in 2020 No. 2020YFC1511704, and in part by Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing 2025.

Data Availability Statement: The data presented in this study are available from the corresponding author upon reasonable request.

Acknowledgments: The authors used ChatGPT (GPT-5.6 Sol, OpenAI) for language polishing and grammatical correction to improve the readability of the manuscript. The tool was not used to generate the original research content, methodology, data, or results. All scientific content was developed by the authors.

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

References

- Guo, F.; Yu, F.R.; Zhang, H.; Li, X.; Ji, H.; Leung, V.C.M. Enabling Massive IoT Toward 6G: A Comprehensive Survey. *IEEE Internet Things J.* **2021**, *8*, 11891–11915. [\[CrossRef\]](#)
- Nguyen, D.C.; Ding, M.; Pathirana, P.N.; Seneviratne, A.; Li, J.; Niyato, D.; Dobre, O.; Poor, H.V. 6G Internet of Things: A Comprehensive Survey. *IEEE Internet Things J.* **2022**, *9*, 359–383. [\[CrossRef\]](#)
- Chi, H.R.; Wu, C.K.; Huang, N.-F.; Tsang, K.-F.; Radwan, A. A Survey of Network Automation for Industrial Internet-of-Things Toward Industry 5.0. *IEEE Trans. Ind. Inform.* **2023**, *19*, 2065–2077. [\[CrossRef\]](#)
- Rawat, B.S.; Srivastava, A.; Shrivastava, V.; Singh, G.; Garg, N. A Comprehensive Analysis of Applications in Internet of Things Networks in 5G and 6G. In Proceedings of the 2024 2nd International Conference on Computer, Communication and Control (IC4), Indore, India, 8–10 February 2024. [\[CrossRef\]](#)
- Guha Roy, B.; Guha Roy, D.; Datta, P.; Bhatia Khan, S.; Asiri, F.; Ayadi, M. Quality of Service-Aware 6G-Enabled NB-IoT for Health Monitoring in Long Distance High-Speed Trains. *IEEE Trans. Consum. Electron.* **2025**, *71*, 1136–1147. [\[CrossRef\]](#)
- Rahmani, H.; Shetty, D.; Wagih, M.; Ghasempour, Y.; Palazzi, V.; Carvalho, N.B.; Correia, R.; Costanzo, A.; Vital, D.; Alimenti, F.; et al. Next-Generation IoT Devices: Sustainable Eco-Friendly Manufacturing, Energy Harvesting, and Wireless Connectivity. *IEEE J. Microwaves* **2023**, *3*, 237–255. [\[CrossRef\]](#)
- Din, I.U.; Taj, I.; Awan, K.A.; Almogren, A.; Altameem, A. Quantum and GAN-Driven Digital Twin Approach for IoT-Based Consumer Electronics Manufacturing. *IEEE Internet Things J.* **2025**, *12*, 3734–3741. [\[CrossRef\]](#)
- Neshenko, N.; Bou-Harb, E.; Crichigno, J.; Kaddoum, G.; Ghani, N. Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 2702–2733. [\[CrossRef\]](#)
- Martalò, M.; Pettorrù, G.; Atzori, L. A Cross-Layer Survey on Secure and Low-Latency Communications in Next-Generation IoT. *IEEE Trans. Netw. Serv. Manag.* **2024**, *21*, 4669–4685. [\[CrossRef\]](#)
- Hammad, M.; Badshah, A.; Almeer, M.A.; Waqas, M.; Song, H.H.; Chen, S.; Han, Z. Lightweight and Robust Key Agreement for Securing IIoT-Driven Flexible Manufacturing Systems. *IEEE Internet Things J.* **2025**, *12*, 17197–17209. [\[CrossRef\]](#)
- Weerasinghe, S.; Zaslavsky, A.; Loke, S.W.; Medvedev, A.; Abken, A.; Hassani, A. Context Caching for IoT-Based Applications: Opportunities and Challenges. *IEEE Internet Things Mag.* **2023**, *6*, 96–102. [\[CrossRef\]](#)
- Bena, Y.A.; Ibrahim, R.; Mahmood, J.; Al-Dhaqm, A.; Alshammari, A.; Nasser, M.; Yusuf, M.N.; Ayemowa, M.O. Big Data Governance Challenges Arising From Data Generated by Intelligent Systems Technologies: A Systematic Literature Review. *IEEE Access* **2025**, *13*, 12859–12888. [\[CrossRef\]](#)
- Abdalzaher, M.S.; Krichen, M.; Shaaban, M.F.; Fouda, M.M. Quality-Focused Internet of Things Data Management: A Survey, Perspectives, Open Issues, and Challenges. *IEEE Internet Things J.* **2025**, *12*, 46431–46458. [\[CrossRef\]](#)
- Fizza, K.; Jayaraman, P.P.; Banerjee, A.; Auluck, N.; Ranjan, R. IoT-QWatch: A Novel Framework to Support the Development of Quality-Aware Autonomic IoT Applications. *IEEE Internet Things J.* **2023**, *10*, 17666–17679. [\[CrossRef\]](#)
- Xue, S.; Huang, H.; Liu, J.; Yang, Q.; Zhao, L.; Wu, H. An Effective Scheme to Solve Critical Data Missing Problems for IoT-Based Smart Energy Management. *IEEE Internet Things J.* **2025**, *12*, 4466–4474. [\[CrossRef\]](#)

16. Zhang, H.; Zhang, X.; Guo, Z.; Wang, H.; Cui, D.; Wen, Q. Secure and Efficiently Searchable IoT Communication Data Management Model: Using Blockchain as a New Tool. *IEEE Internet Things J.* **2023**, *10*, 11985–11999. [\[CrossRef\]](#)
17. Liu, Y.; Zhang, C.; Yan, Y.; Zhou, X.; Tian, Z.; Zhang, J. A Semi-Centralized Trust Management Model Based on Blockchain for Data Exchange in IoT System. *IEEE Trans. Serv. Comput.* **2023**, *16*, 858–871. [\[CrossRef\]](#)
18. Wu, W.; Peng, H.; Tong, F.; Li, L. Novel Secure Data Transmission Methods for IoT Based on STP-CS with Multilevel Critical Information Concealment Function. *IEEE Internet Things J.* **2023**, *10*, 4557–4578. [\[CrossRef\]](#)
19. Tong, F.; Chen, X.; Huang, C.; Zhang, Y.; Shen, X. Blockchain-Assisted Secure Intra/Inter-Domain Authorization and Authentication for Internet of Things. *IEEE Internet Things J.* **2023**, *10*, 7761–7773. [\[CrossRef\]](#)
20. Xu, R.; Nagothu, D.; Chen, Y.; Aved, A.; Ardiles-Cruz, E.; Blasch, E. A Secure Interconnected Autonomous System Architecture for Multi-Domain IoT Ecosystems. *IEEE Commun. Mag.* **2024**, *62*, 52–57. [\[CrossRef\]](#)
21. Belchior, R.; Vasconcelos, A.; Guerreiro, S.; Correia, M. A Survey on Blockchain Interoperability: Past, Present, and Future Trends. *ACM Comput. Surv.* **2021**, *54*, 168. [\[CrossRef\]](#)
22. Ren, K.; Ho, N.-M.; Loghin, D.; Nguyen, T.-T.; Ooi, B.C.; Ta, Q.-T.; Zhu, F. Interoperability in Blockchain: A Survey. *IEEE Trans. Knowl. Data Eng.* **2023**, *35*, 12750–12769. [\[CrossRef\]](#)
23. Li, X.; Li, R. A Comprehensive Review for 4-D Trust Management in Distributed IoT. *IEEE Internet Things J.* **2023**, *10*, 21738–21762. [\[CrossRef\]](#)
24. Gangwani, P.; Perez-Pons, A.; Upadhyay, H. A Semantic Trust Management Model for Wireless Sensor Networks. *IEEE Sens. J.* **2025**, *25*, 3672–3681. [\[CrossRef\]](#)
25. Alhandi, S.A.; Kamaludin, H.; Alduais, N.A.M. Trust Evaluation Model in IoT Environment: A Comprehensive Survey. *IEEE Access* **2023**, *11*, 11165–11182. [\[CrossRef\]](#)
26. Macedo, E.L.C.; Delicato, F.C.; de Moraes, L.F.M.; Fortino, G. Assigning Trust to Devices in the Context of Consumer IoT Applications. *IEEE Consum. Electron. Mag.* **2024**, *13*, 12–21. [\[CrossRef\]](#)
27. Arachchige, P.C.M.; Bertok, P.; Khalil, I.; Liu, D.; Camtepe, S.; Atiquzzaman, M. A Trustworthy Privacy Preserving Framework for Machine Learning in Industrial IoT Systems. *IEEE Trans. Ind. Inform.* **2020**, *16*, 6092–6102. [\[CrossRef\]](#)
28. Liu, X.; Wang, R.; Peng, L.; Luo, D.; Xu, G.; Chen, X.-B.; Xiong, N.; Liu, X. Secure Computing Protocols for Internet of Things Data Fusion Based on Set Intersection and Union Against Malicious Enemies. *IEEE Internet Things J.* **2024**, *11*, 24918–24932. [\[CrossRef\]](#)
29. Ahmed, M.; Khan, W.U.; Alrayes, F.S.; Said, Y.; Al-Sharafi, A.M.; Kim, M.-H.; Dashdondov, K.; Ullah, I. Joint Encryption and Optimization for 6G MEC-Enabled IoT Networks. *IEEE Access* **2025**, *13*, 79757–79770. [\[CrossRef\]](#)
30. Huang, C.; Chen, S.; Zhang, Y.; Zhou, W.; Rodrigues, J.J.P.C.; de Albuquerque, V.H.C. A Robust Approach for Privacy Data Protection: IoT Security Assurance Using Generative Adversarial Imitation Learning. *IEEE Internet Things J.* **2022**, *9*, 17089–17097. [\[CrossRef\]](#)
31. Li, T.; Wang, H.; He, D.; Yu, J. Designated-Verifier Aggregate Signature Scheme with Sensitive Data Privacy Protection for Permissioned Blockchain-Assisted IIoT. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 4640–4651. [\[CrossRef\]](#)
32. Tan, J.; Shi, J.; Wan, J.; Dai, H.-N.; Jin, J.; Zhang, R. Blockchain-Based Data Security and Sharing for Resource-Constrained Devices in Manufacturing IoT. *IEEE Internet Things J.* **2024**, *11*, 25558–25567. [\[CrossRef\]](#)
33. Chen, Y.; Li, B.; Li, W.; Zeng, B.; Yin, J.; Deng, S. Blockchain-Based Nash Bargaining for Task Scheduling in IoT Edge Computing Environments. *IEEE Internet Things J.* **2024**, *11*, 13851–13864. [\[CrossRef\]](#)
34. Yao, P.; Yan, B.; Yang, T.; Wang, Y.; Yang, Q.; Wang, W. Security-Enhanced Operational Architecture for Decentralized Industrial Internet of Things: A Blockchain-Based Approach. *IEEE Internet Things J.* **2024**, *11*, 11073–11086. [\[CrossRef\]](#)
35. Xia, H.; Cobblah, C.N.A.; Xia, Q.; Gao, J. A Scalable and Memory-Efficient Architecture for Blockchain-Based IoT Privacy and Security. *IEEE Internet Things J.* **2025**, *12*, 29811–29829. [\[CrossRef\]](#)
36. Yang, B.; Chen, L.; Shen, J.; Wang, H.; Ma, Y. FHE-Based Publicly Verifiable Sealed-Bid Auction Protocol Atop Cross-Blockchain. *IEEE Trans. Netw. Serv. Manag.* **2025**, *22*, 832–845. [\[CrossRef\]](#)
37. Duan, L.; Wang, Y.; Liu, J.; Ni, W.; Li, C.; Wang, W.; Liu, Z.; Jamalipour, A. Secure and Fine-Grained Data Sharing in Internet of Things: Integration of Interplanetary File System and Cross-Blockchain for Access Control. *IEEE Internet Things J.* **2025**, *12*, 37301–37308. [\[CrossRef\]](#)
38. Yang, L.; Dong, X.; Wan, Z.; Gao, S.; Tong, W.; Lu, D.; Shen, Y.; Du, X. AsyncSC: An Asynchronous Sidechain for Multi-Domain Data Exchange in Internet of Things. In Proceedings of the IEEE INFOCOM 2025–IEEE Conference on Computer Communications, London, UK, 19–22 May 2025; pp. 1–10. [\[CrossRef\]](#)
39. Zhang, M.; Qu, Q.; Ning, L.; Fan, J. On Time-Aware Cross-Blockchain Data Migration. *Tsinghua Sci. Technol.* **2024**, *29*, 1810–1820. [\[CrossRef\]](#)
40. Ranathunga, T.; McGibney, A.; Rea, S.; Bharti, S. Blockchain-Based Decentralized Model Aggregation for Cross-Silo Federated Learning in Industry 4.0. *IEEE Internet Things J.* **2023**, *10*, 4449–4461. [\[CrossRef\]](#)
41. Wang, F.; Cui, J.; Zhang, Q.; He, D.; Zhong, H. Blockchain-Based Secure Cross-Domain Data Sharing for Edge-Assisted Industrial Internet of Things. *IEEE Trans. Inf. Forensics Secur.* **2024**, *19*, 3892–3905. [\[CrossRef\]](#)

42. Barbàra, F.; Schifanella, C. MP-HTLC: Enabling Blockchain Interoperability Through a Multiparty Implementation of the Hash Time-Lock Contract. *Concurr. Comput. Pract. Exp.* **2023**, *35*, e7656. [[CrossRef](#)]
43. Liu, Y.; Liang, W.; Xie, K.; Xie, S.; Li, K.; Meng, W. LightPay: A Lightweight and Secure Off-Chain Multi-Path Payment Scheme Based on Adapter Signatures. *IEEE Trans. Serv. Comput.* **2024**, *17*, 1622–1635. [[CrossRef](#)]
44. Zhang, Y.; He, X.; Wang, X.; Wang, H.; Xu, K. DeCross: Toward Accountable and Efficient Cross-Chain Collaboration in IIoT. *IEEE Trans. Ind. Inform.* **2025**, *21*, 4947–4957. [[CrossRef](#)]
45. Yu, B.; Guan, Y.; Geng, S.; Miao, L.; Zhang, Y.; Gong, Y. A Blockchain-Enhanced Secure and Reliable Data Transaction Scheme in MAS via HTLC. In Proceedings of the 2024 3rd Conference on Fully Actuated System Theory and Applications (FASTA), Shenzhen, China, 10–12 May 2024; pp. 494–499. [[CrossRef](#)]

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