Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon

Research article

Secure cross-chain interaction solution in multi-blockchain environment

Lili Cheng [a], Zhiying Lv [b], Osama Alfarraj [c], Amr Tolba [c], Xiaofeng Yu [d,*], Yongjun Ren [b]

[a] *Nanjing Second Space Network Technology Co., Ltd, Nanjing, JiangSu, China*
[b] *Engineering Research Center of Digital Forensics of Ministry of Education, School of Computer, Nanjing University of Information Science and Technology, Nanjing, JiangSu, China*
[c] *Computer Science Department, Community College, King Saud University, Riyadh, Saudi Arabia*
[d] *School of Business, Nanjing University, Nanjing, JiangSu, China*

ARTICLE INFO

Keywords:
Blockchain interoperability
Multichain
Cross-chain transaction
Data interaction
Security

ABSTRACT

In the context of the increasingly diversified blockchain technology, interoperability among heterogeneous blockchains has become key to further advancing this field. Existing cross-chain technologies, while facilitating data and asset exchange between different blockchains to some extent, have exposed issues such as insufficient security, low efficiency, and inconsistent standards. Consequently, these issues give rise to significant obstacles in terms of both scalability and seamless communication among blockchains within a multi-chain framework. To address this, this paper proposes an efficient method for cross-chain interaction in a multi-chain environment. Building upon the traditional sidechain model, this method employs smart contracts and hash time-locked contracts (HTLCs) to design a cross-chain interaction scheme. This approach decentralizes the execution of locking, verifying, and unlocking stages in cross-chain transactions, effectively avoiding centralization risks associated with third-party entities in the process. It also greatly enhances the efficiency of fund transfers between the main chain and sidechains, while ensuring the security of cross-chain transactions to some extent. Additionally, this paper innovatively proposes a cross-chain data interaction strategy. Through smart contracts on the main chain, data from sidechains can be uploaded, verified, and stored on the main chain, achieving convenient and efficient cross-chain data sharing. The contribution of this paper is the development of a decentralized protocol that coordinates the execution of cross-chain interactions without the need to trust external parties, thereby reducing the risk of centralization and enhancing security. Experimental results validate the effectiveness of our solution in increasing transaction security and efficiency, with significant improvements over existing models. Our experiments emphasize the system's ability to handle a variety of transaction scenarios with improved throughput and reduced latency, highlighting the practical applicability and scalability of our approach.

* Corresponding author.
E-mail address: xiaofengyu@nju.edu.cn (X. Yu).

https://doi.org/10.1016/j.heliyon.2024.e28861
Received 24 January 2024; Received in revised form 20 March 2024; Accepted 26 March 2024

Available online 29 March 2024

1. Introduction

Currently, with the continuous development of blockchain technology [1–3], the application areas of blockchain are constantly expanding, accompanied by an increasing number of structurally diverse blockchains [4]. These blockchains are constantly updated and iterated and will inevitably face more complex business requirements [5]. They will begin to generate the demand for interconnection and interoperability of data, assets and functions across heterogeneous chains and develop towards cross-chain applications [6]. However, the lack of interconnection between different blockchain systems greatly restricts and hinders the further development and application of blockchain technology, and the demand for interconnection of applications and data between chains has become an urgent problem to be solved in the path of blockchain development. Therefore, research on the expandability and interoperability of blockchain has become more and more popular in the current blockchain direction [7–9].

Blockchain interoperability, also known as cross-chain interoperability, focuses on realizing data interoperability [10], value interoperability [8] and function interoperability [11] between different blockchains. Data interoperability focuses on cross-chain data access and cross-chain data transfer between different blockchains. Value interoperability focuses on cross-chain asset exchange and cross-chain asset transfer between different blockchains. Functional interoperability aims at realizing the integration of functions between different blockchains, such as the cross-chain invocation of smart contracts.

Currently, mainstream cross-chain technology mainly includes four methods [12–14]: hash time locking, notary mechanism, side chain/relay chain technology and distributed key control. Among them, sidechain technology [15] as a cross-chain technology realizes asset interaction and data interaction between different blockchains through bidirectional hooks. Sidechain technology is great value for many problems faced by blockchain. For example, the use of sidechains can enable users to conveniently interact between different blockchains and securely realize the transfer of assets and data [16]. By linking a specific sidechain to the main chain, it can effectively alleviate the pressure of processing transactions on the main chain, thus realizing a kind of extension of the performance of the existing blockchain. Because the currency between the sidechain and the main chain can be converted at will, the sidechain does not need to maintain its own independent currency, so it can be used to realize the expansion of the performance of the existing blockchain [17]. Therefore, the sidechain can be used to realize the exploration of new functions, and if the functions of the sidechain prove to be popular, the sidechain can eventually be made the new main chain by transferring all the assets of the main chain to the sidechain and being abandoned for use.

In a multi-chain environment, cross-chain interaction is critical, which enables asset interaction and data interaction between different blockchains. This multi-chain environment is characterized by the main chain as the core chain, which connects with multiple side chains and realizes the interactions between the chains through cross-chain technology. A multi-chain environment with a main chain and multiple side chains features a core main chain, extended side chain functionality, and cross-chain interactions.

However, the current application of sidechain technology in a multi-chain environment may suffer from security risks, inconsistent cross-chain standards, and cross-chain performance limitations [18,19]. Although the existing two-way pegging mechanism can realize asset transfer between sidechain and mainchain, it suffers from centralization risk and low security. Multi-signature or joint pegging mechanisms achieve decentralization to a certain extent but still rely on a few entities to manage the transfer of funds and do not completely eliminate the problem of centralization. Simplified Payment Verification (SPV) proofs [20], while enabling decentralization, need to wait for confirmation cycles, reducing the efficiency of cross-chain transactions and increasing the size of transactions. In summary, all current two-way pegging mechanisms have a variety of problems, which makes all blockchain systems that adopt sidechain as a cross-chain interaction method face many risks, such as centralization and low security, as well as a significant impact on the time and space efficiency of cross-chain transactions.

Addressing these challenges, this paper builds upon traditional sidechain technology and enhances the bidirectional pegging technique used for fund transfers between main and side chains. By leveraging smart contracts and hash time-locked contracts (HTLCs), we have designed a cross-chain interaction scheme that enables the decentralized execution of the locking, verification, and unlocking phases of cross-chain transactions without the need for any third-party entities. Moreover, this approach eliminates the waiting period for any confirmation cycles. This not only mitigates the centralized risks associated with third-party involvement in cross-chain transactions but also significantly improves the efficiency of fund transfers between the main chain and side chains. Furthermore, it ensures a degree of security for cross-chain transactions.

The main contribution of this paper is:

- Aiming at the problems in the existing cross-chain technology, an efficient cross-chain interaction method is proposed to realize a more secure and efficient asset transfer and data exchange mechanism.
- In the locking, verification, and unlocking phases of cross-chain transactions, smart contracts are utilized to avoid the centralization risk brought about by the third party, improve the efficiency of fund transfer, and ensure the security of cross-chain transactions to a certain extent.
- Employing smart contracts for data storage on side chains facilitates secure and efficient cross-chain data exchanges, allowing various blockchain systems to seamlessly share information about transactions and smart contracts.

The remainder of this document is structured as follows: Section 2 provides an overview of the background related to blockchain interoperability, including existing solutions for cross-chain technology and pertinent research; Section 3 outlines an effective model for cross-chain interactions within a multi-chain setting, detailing the specific process involved in such interactions; Section 4 presents an experimental evaluation of the approach proposed in this paper; and Section 5 offers the concluding remarks of the study.

2. Background & related work

The number of global blockchains is increasing [21], and the mutual isolation of different blockchain networks leads to the inability of the chains to effectively perform operations such as digital asset transfer and cross-chain communication. In recent times, as the variety and complexity of blockchain applications have grown, an increasing number of blockchain projects have started to demand and propose solutions for cross-chain functionality, leading to the swift advancement of the concept of blockchain interoperability.

2.1. Blockchain interoperability

Blockchain interoperability involves the transfer of information between specific blockchain systems, known as blockchain system instances, and various other types of systems. These other systems can encompass applications built on top of the blockchain, different blockchain networks, and data systems that operate outside of the blockchain infrastructure. By exchanging information, these system instances can share transaction data, smart contract information, etc., and realize the interaction and operation between each system.

By formally defining and modeling blockchain interoperability, a cross-chain interoperability process with precision and clarity can be provided to ensure a clear understanding of blockchain interoperability concepts, goals, and requirements. In addition, formal definitions can serve as specifications or standards that unify the way blockchain interoperability is understood and implemented, which makes it possible to build a cross-chain interoperability platform that takes into account various heterogeneous blockchains.

Blockchain interoperability includes exchange interoperability and transfer interoperability. The concept of swap interoperability is centered around the capacity for asset exchanges between distinct blockchain ecosystems. Specifically, it describes a process by which blockchain systems A and B can reach a new, consistent system state in a finite amount of time through the transaction T_{swap}. Transfer interoperability, on the other hand, refers to the ability of an asset in one blockchain system to be verified and validated in another system. This type of interoperability involves the transfer of assets between different blockchain systems. In the following paper, we will provide a formal definition of blockchain interoperability and build a blockchain interoperability model.

Definition 1 *(Swapping interoperability)*. For assets $a_1 \in A$ and $a_2 \in B$ in any blockchain system A and B, the operations of Eq (1) are executed through a transaction T_{swap}:

$$T_{swap} \rightarrow \mathrm{A} \Leftrightarrow T_{swap} \rightarrow \mathrm{B} \tag{1}$$

where the transaction T_{swap} is able to transform the blockchain systems A and B into a new consistent system state (i.e., $a_1 \in B \wedge a_2 \in A$) within a bounded time interval ΔT; and there exists a consistency mechanism that is able to check the integrity condition of T_{swap} on both A and B at the same time.

Definition 2 *(Transferring interoperability)*. For any asset a in A, there exists a $Proof \in B \Leftrightarrow a \in A$ that, through a transaction T_{trans}, perform the operation in Eq (2):

$$T_{trans} \rightarrow \mathrm{A} \Leftrightarrow T_{trans} \rightarrow \mathrm{B} \tag{2}$$

where transactions T_{trans}, capable of converting blockchain systems A and B to a new consistent system state (i.e. $\exists a(a \in B \wedge a \notin A)$) within a bounded time interval ΔT.

Blockchain interoperability model: For any two existing blockchain systems A and B, the underlying ledgers are L_1 and L_2 respectively. Where M processes are running on A and N processes are running on B. The state of a blockchain system can be altered through the action of processes either by committing transactions T to the foundational ledger L or by ceasing any interaction with the system. Where can write transactions T_M to the ledger L_1 and N has transactions T_N that can be written to the ledger L_2. The validation function $Verify$ can validate the result of the transaction against expectations, e.g., by specifying information such as the value of the transaction and the addresses of the interacting parties. Mowing the result of Verify V_N is a validation of the transaction, while owning the result of $Verify$ is used to validate the transaction T_N. In short, M expects to T_N be written to L_2, N expects to T_M be written to L_1. Thus $V_M = Verify(T_M)$ implies that T_M is valid on A, otherwise it would not be possible to write to $L_1(V_N$ ditto).

A formal definition of blockchain interoperability: In order to synchronize the process M and N, i.e., N writes a transaction T_N to L_2 when and only when M writes a transaction T_M to book L_1, it must satisfy $V_N = Verify(T_M)$ and $V_M = Verify(T_N)$. Since T_M and T_N both transactions must be contained in both ledgers L_1 and L_2 to guarantee the atomicity of the asset exchange. For this purpose, M has to persuade N that it has generated a transaction T_M included in L_1. In other words, the process N must confirm that the state of ledger L_1 exists within a specified timeframe t. The correct blockchain interoperability to achieve this goal must have the following properties:

Definition 3 *(Availability)*. Should M and N perform their operations without error, ensuring that T_M and T_N are correctly validated and align with anticipated outcomes (meaning both transactions are legitimate), then T_M will be incorporated into L_1 and T_N into

L_2. Conversely, if any of the transactions fail to meet the validation criteria or diverge from expected results, both processes will terminate prematurely. This is specified as indicated in Eq (3):

$$\begin{aligned}&\left(Verify\left(T_M\right) = V_N \wedge Verify\left(T_N\right) = V_M = \perp \Rightarrow T_M \in L_1 \wedge T_N \in L_2\right)\\&\wedge\left(Verify\left(T_M\right) \neq V_N \vee Verify\left(T_N\right) \neq V_M \Rightarrow T_M \notin L_1 \wedge T_N \notin L_2\right)\end{aligned} \tag{3}$$

Definition 4 *(Atomicity)*. To guarantee the atomicity of cross-blockchain interactions, it is critical to avoid scenarios in which M successfully commits to L_1 at time t without N managing to record T_N prior to t, or wherein N accomplishes a write to L_2 at time t, while M does not successfully commit T_M before t. Specifically as expressed in Eq (4):

$$\neg\left(\left(T_M \in L_1 \wedge T_N \notin L_2\right) \vee \left(T_M \notin L_1 \wedge T_N \in L_2\right)\right) \tag{4}$$

Definition 5 *(Promptness)*. Ultimately, if process M operates as intended, it will successfully record a legitimate transaction T_M onto its ledger, L_1.

Definition 6 *(Blockchain interoperability)*. Imagine two processes, where M resides on system A and N on system B, each having transactions T_M and T_N that require synchronization. Then, the correct blockchain interoperability implements $T_M \in L_1$ and $T_N \in L_2$ with validity, atomicity and timeliness.

2.2. Introduction to cross-chain technology

The core of cross-chain blockchain technology lies in employing technical measures across different chains to enable the movement of value from one blockchain to another, as outlined by Ren et al. [22]. This facilitates the unrestricted movement of assets across various blockchains, whether they are of similar or different structures, paving the way for a truly interconnected blockchain ecosystem. In their work, Han et al. [23] conduct an in-depth analysis of current cross-chain technologies, focusing particularly on the challenges related to security, privacy, and operational efficiency. They suggest an interoperability framework for blockchains, evaluating the possible risks and problems, and organizing them by the type of technology and the intended purpose of cross-chain interactions. Currently, the core technologies enabling cross-chain functionality within the blockchain sphere are categorized into four primary types based on their foundational principles and implementation methods: notary schemes, sidechains along with relay systems, hash time-locked contracts, and distributed key management techniques. This document will focus on elucidating the fundamental principles and associated research concerning hash time-locked contracts and sidechain methodologies as employed in our study.

2.2.1. Hash timelock contract

As the technical basis for realizing the current atomic cross-chain digital asset transaction scheme, Nolan proposed the idea of atomic transfers in the BitcoinTalk forum in May 2013 [24]. Following refinement and advancement, Nolan's methodological innovation found its first application in the architecture of the Bitcoin Lightning Network [25]. This technique stands as a cross-chain solution that enables the movement of capital between diverse blockchain platforms, bypassing the need for any external trusted intermediaries, through the synergistic use of hash and time lock mechanisms. This approach offers a cross-chain mechanism designed to enable the transfer of assets across various blockchain networks without requiring third-party notarization, relying on the combined functionality of hash and time locks [26]. The hash house and time lock in the hash time lock are two different locking mechanisms. Time locks are used to restrict the execution of a specific operation within a certain period of time and will not be able to be executed beyond the specified time. The hash lock mechanism demands the submission of the accurate preimage for resource unlocking, whereas the hash time lock's utilization focuses on safeguarding transaction security during the cross-platform exchange of assets between different blockchains. During the execution of the hash time lock, the counterparty who wants to unlock the locked asset needs to provide the original hash image for verification. Only after the verification is legitimate can the asset be successfully unlocked. Should there be no withdrawal of the encumbered asset within the set timeframe, the time lock mechanism ensures the return of all assets to the initiating account. Known as hash locking, this protocol is extensively applied in cross-chain technologies due to its capacity to facilitate asset exchanges that are decentralized and free from intermediaries. The parties to the transaction do not need to trust each other to complete the exchange of assets. Vitalik Buterin [27] has described the problem of blockchain interoperability and introduced the application of the hash time locking algorithm in cross-chain technology. This algorithm provides a reliable technical means to achieve secure asset exchange. Jinzhong Li and colleagues [28] introduced a universally adaptable cross-chain transaction protocol that leverages hash time-locking technology. This protocol employs cross-chain tech to address issues of network isolation and facilitate interoperability within the energy sector's blockchain. Blerim Rexha and his team [29] explored the integration of blockchain bridges into electronic voting systems, underscoring their capability to enhance both trust and interoperability within these systems. They utilize several techniques such as atomic swaps, sidechains, cross-chain bridges, token wrapping, and connectivity protocols to illustrate how blockchain bridges facilitate the exchange of data and support dual decentralization models.

2.2.2. Sidechain

Sidechain, also known as pegged sidechain, are a kind of cross-chain technology that realizes asset transactions or data-information interactions between different blockchains through two-way peg technology [30]. At the heart of sidechain functionality lies the two-way peg system, a mechanism that allows for the transfer of tokens between the main blockchain and a sidechain according to a predetermined or formulaic exchange rate, with the provision for these tokens to be returned [31]. Three key methods have been developed to facilitate this two-way pegging within sidechain operations:

Centralized Method: In this approach, a single trusted authority oversees the locking and unlocking of funds on both the main and side chains. Its simplicity and quick transaction capability come at the cost of heightened centralization risk, potentially compromising the security of the transfers.

Federated or Multi-Signature Method: Building on the centralized model, this method distributes fund management across a consortium or group, requiring consensus (typically a majority) for transactions. While this mitigates some centralization concerns and retains transaction speed, the reliance on a select group may still pose risks to trust and security [32].

Simplified Payment Verification (SPV) Method: SPV confirms a transaction's validity on the blockchain through proof of computational effort. Transfers to the main chain involve locking funds to an address, which are then unlocked on the sidechain using SPV proofs, allowing for the transaction without further main chain interaction. This process, which mirrors for transfers back to the main chain, supports decentralization by eliminating third-party involvement. However, the necessity for chain synchronization introduces significant delays—usually a day or two—slowing down the transaction process. The need for SPV proofs also increases the size of transactions, thereby consuming more blockchain space [33].

Blockstream, a company joined by the core developers of Bitcoin, proposed the concept of pegged side chains in a white paper released in October 2014 [33], with the aim of realizing the cross-chain transfer of assets from different blockchains as well as the possibility of implementing more technological and financial innovations without affecting the main chain. Two-way pegging, fundamental to sidechain technology, is characterized by its role in facilitating the bidirectional exchange of assets across different blockchain networks. This mechanism is categorized into two forms: symmetric two-way peg, where the asset conversion rate is equal in both directions, and asymmetric two-way peg, where the conversion rates vary between the two chains. In December 2016, Blockstream introduced the idea of sidechains equipped with strong federations [32]. This concept employs addresses that require multiple signatures for transactions, managed by several entities, aiming to decrease transaction times and enhance the interoperability between different blockchains. In May 2016, the blockchain technology firm ConsenSys, based in the United States, unveiled BTCRelay [34], a mechanism for enabling Ethereum to access Bitcoin blockchain data directly. Following that, in June 2016, Kwon outlined Cosmos [35], a new network design aimed at facilitating the integration and interoperability of different blockchain networks. Later, in November 2016, Wood proposed a diverse multi-chain framework in the Polkadot white paper [36], which is crafted to allow decentralized and trustless interactions among various consensus mechanisms.

The Zcash (ZEC) team developed the XCAT (Cross-Chain Atomic Transaction) tool [37], which realizes cross-chain atomic transactions between ZEC and BTC. Bancor builds a cross-chain decentralized liquidity network based on smart contracts to realize automatic valuation and exchange of blockchain assets without counterparties; OneLedger provides a set of solutions for enterprise systems to connect with the blockchain system, which connects to various types of private, alliance, and public chains through side links and communicates with the enterprise-level systems through intermediate protocol layers.

3. Efficient cross-chain interaction solution in multi-chain environment

3.1. System overview

The system model in this paper uses one main chain and multiple side chains to complete the construction of the main side multi-chain structure and uses a main side chain two-way pegged trading system to transfer funds between the main side chains. All the users involved in the transaction have an account in the main chain; in addition, each user also has an account in its corresponding side chain. The model structure of the entire master sidechain system is shown in Fig. 1.

The master chain is the first chain generated by the system, which is responsible for the confirmation of the side chains and guarantees that the side chains can run well. Each side chain is categorized according to its specific application scenarios and functions, and in each category, the side chain nodes handle the same type of transactions. Sidechains in different categories exist due to the existence of special functions or sidechains without monetary functions, so the transactions between them cannot be sent directly and need unified management by the cross-chain transaction system. Within distinct namespaces, side chains transmit their blocks to the main chain, where the main chain's nodes are responsible for verifying the side chains' nodes. This process is crucial for confirming the authenticity and security of the side chains' blocks.

The two-way pegged cross-chain transaction system for the transfer of funds from the main side chain is realized entirely through the smart contract on the chain without introducing any third-party entity, and the entire cross-chain fund transfer process is carried out in accordance with the process set by the smart contract without waiting for confirmation from other parties. By implementing this approach, it not only mitigates the risk of centralization to a degree and enhances the overall security of cross-chain transactions but also enables users to swiftly execute fund transfers. This markedly boosts the efficiency of cross-chain interactions between the main chain and side chains.

Beyond just facilitating fund transfers between the main chain and side chains, the methodology outlined in this document also encompasses the exchange of data across chains. This is achieved via smart contracts deployed on the main chain, which enable the transfer of data from side chains to the main chain. Side chains push the required data to the main chain, where it is validated and

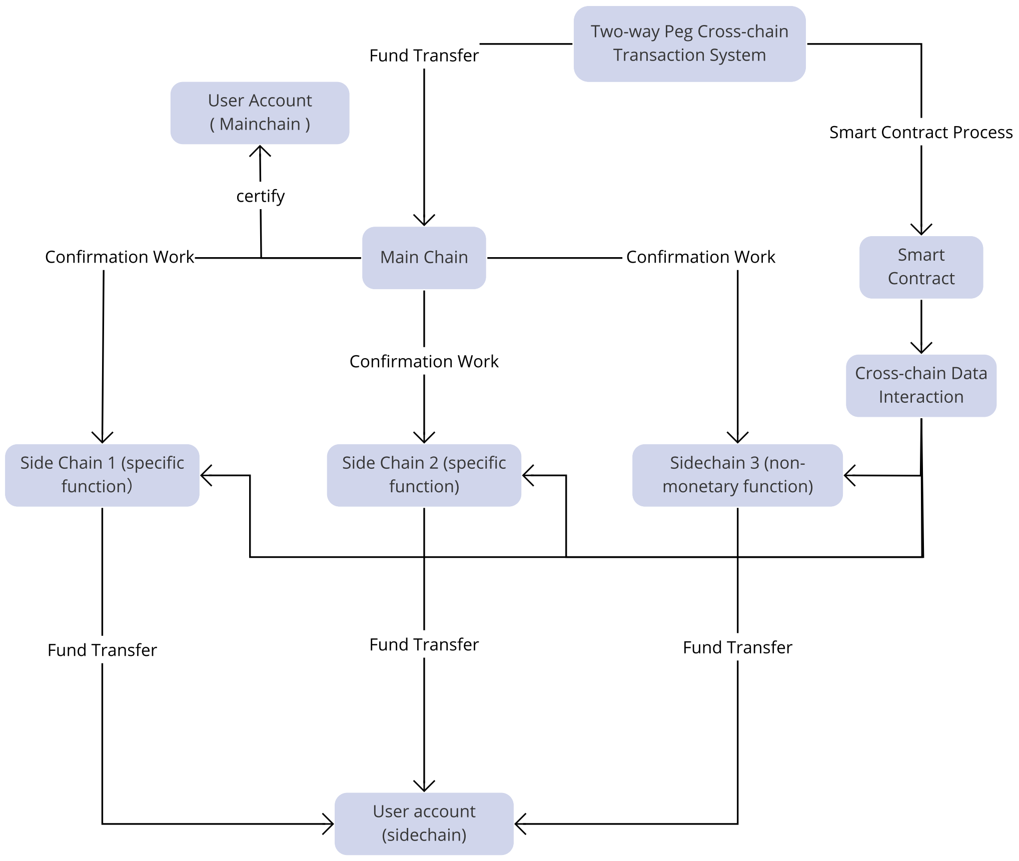
Two-way Peg Cross-chain Transaction System
Fund Transfer
Smart Contract Process
User Account (Mainchain)
certify
Confirmation Work
Main Chain
Confirmation Work
Smart Contract
Confirmation Work
Cross-chain Data Interaction
Side Chain 1 (specific function)
Side Chain 2 (specific function)
Sidechain 3 (non-monetary function)
Fund Transfer
Fund Transfer
Fund Transfer
User account (sidechain)

Fig. 1. Cross-chain interaction system model.

stored by a smart contract. This setup allows other side chains to access and retrieve this data through smart contract calls, thus enabling a comprehensive cross-chain data sharing mechanism.

3.2. Cross-chain transaction process

The framework for main-to-side chain cross-chain transactions proposed in this study is segmented into three distinct phases: initiation, confirmation, and completion. Initially, a user triggers the transaction by applying to the cross-chain mechanism, which, upon approval, issues a unique key based on predefined criteria. This key enables the user to secure their assets for transfer. During the confirmation phase, the system evaluates the user-provided information to ascertain the proper sequestration of funds. Successful verification progresses the transaction to the final phase. Here, the user employs the previously received key on the intended chain to release the assets, thereby finalizing the transfer process. The specific flow of the whole cross-chain fund transfer is shown in Fig. 2.

3.2.1. Lock-in phase

After confirming the need for fund transfers, the user signs the current account address, target chain account address, transfer amount, and other information and sends it to the cross-chain system. After receiving the signature, the system verifies it and generates the key for locking the funds after confirming that the information is correct. The following initial service Algorithm 1 includes the processes of signature verification and key generation.

Which (m, σ) is the user sent to the system by the message signature pair for the user's public key pk. After the system receives the message signature pair, it calls the versign function to verify the signature, and if the verification passes, i.e., result is 1, it calls the initialization function to initialize the service, and if the verification of the whole signature does not match the information, i.e., result is 0, it returns Verification Failed and ends the algorithm. In the initialization function, the information in the message sent by the user can be extracted by calling, including the source blockchain identifier ID_S, target blockchain identifier ID_T, and transaction amount. The system then generates the key r through a random function and calculates the hash value H_{lock} used to lock the funds through a hash function. In this context, the initial image r is formed from a sequence that includes several components and a random value, as defined by Eq (5) and (6):

$$r_0 = Random(i)_{i\in[0,n+1]} \tag{5}$$

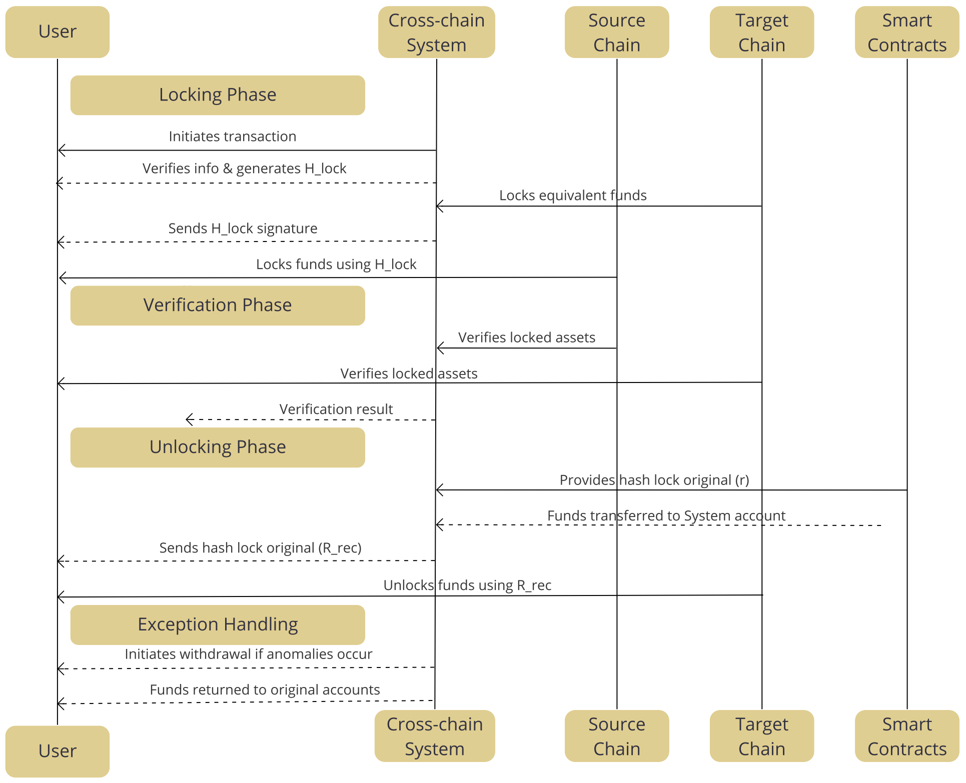

Fig. 2. Cross-chain transaction process.

Algorithm 1 Initialization service.

```
Input: $m, \sigma, pk$
Output: $(ID_S, ID_T, accunt); H_{lock}$
 1: function VERSIGN($m, \sigma, pk$)
 2:     $result \leftarrow ver(m, \sigma, pk)$
 3:     if $result = 1$ then
 4:         $initialization()$
 5:     else
 6:         return Verification Failed
 7:     end if
 8: end function
 9: function INITIALIZATION($m$)
10:     $(ID_S, ID_T, accunt) \leftarrow getInformation(m)$
11:     $r \leftarrow Random()$
12:     return $(ID_S, ID_T, accunt), H_{lock} = H(r)$
13: end function
```

$$r = (Multielements) + r_0 \tag{6}$$

In this scenario, *Multielements* represents a composite value obtained through a multielement mapping process, which could include various parameters such as geographical coordinates (e.g., latitude and longitude) of a location, along with a unique identifier for an enterprise organization. Additionally, r_0 denotes the corresponding random number generated within the blockchain system.

After completing the initialization service, the cross-chain system will lock the same amount of currency in the target chain as the user needs to transfer. The steps performed by the smart contract in this process are shown in the Algorithm 2.

Where $accunt$ is the number of funds to be locked, the value of the hash lock used to lock the asset, Acc_{t1} is the fund account of the cross-chain system in the target chain, and t1 is the time interval of locking. First the system obtains the current time from the timestamp on the chain as the time to start locking $Time_{start}$, and then calls the function $getBalance()$ to query the balance on Bal_{sys} the account Acc_{t1}, if the balance is greater than or equal to the number of funds to be locked, the locking starts, otherwise it returns locking failure. The system calculates the locking time and deducts the number of funds locked by the user from the account to complete the locking. In the subsequent locking time, if the system receives the key of the hash value unlocked by the user, it runs the unlocking program, and if it is still not unlocked at the end of the locking time, the funds will be withdrawn to the original account.

Algorithm 2 Systematic locking of assets.

```
Input: $accunt; H_{lock}; Acc_{t1}; t_1$
Output: $lock\ result$
 1: function LOCK()
 2:     $Time_{start} = getTimestamp()$
 3:     $Bal_{sys} \leftarrow getBalance(Acc_{t_1})$
 4:     if $Bal_{sys} >= accunt$ then
 5:         $Time_{lock} = Time_{start} + t_1$
 6:         Decrease the assets in the $Acc_{t1}$ according to the accunt
 7:         return Lock successfully
 8:     end if
 9:     for i ← 0 to $t_1$ do
10:         if get key for unlocking $H_{lock}$ then
11:             Call $Unlock()$
12:         else
13:             return Lock failed
14:         end if
15:     end for
16: end function
```

After the currency in the target chain is locked, the system signs the hash value H_{lock} and sends it to the user. After receiving the signature, the user verifies the validity of the signature, and after confirmation, the user uses the hash value H_{lock} to lock the asset in the source chain for the amount he or she wants to transfer, and the locking time is t_2. The specific process is shown in the Algorithm 3.

Algorithm 3 User locked assets.

```
Input: $accunt; H_{lock}; Acc_{s1}; t_2$
Output: $lock\ result$
 1: function LOCK()
 2:     $Time_{start} = getTimestamp()$
 3:     $Bal_{user} \leftarrow getBalance(Acc_{s_2})$
 4:     if $Bal_{user} >= accunt$ then
 5:         $Time_{lock} = Time_{start} + t_2$
 6:         Decrease the assets in the $Acc_{s1}$ according to the accunt
 7:         return Lock successfully
 8:         for i ← 0 to $t_2$ do
 9:             if get key for unlocking $H_{lock}$ then
10:                 Call $Unlock()$
11:             end if
12:             $Time_{end} = getTimestamp()$
13:             if $Time_{end} > Time_{lock}$ then
14:                 Call $Withdraw()$
15:             end if
16:         end for
17:     else
18:         return Lock failed
19:     end if
20: end function
```

Where accunt is the number of funds to be locked, H_{lock} is the value of the hash lock used to lock the asset, Acc_{s2} is the user's fund account in the source chain, t_2 is the time interval for locking, and $t_2 < t_1$. After obtaining the current time through the timestamp, the function $getBalance()$ is called to query the balance Bal_{user} on the account Acc_{s2} and compare it with the number of funds to be locked, if the balance of the user's account meets the requirements, the locking will be started, or else return to the locking failure. After the system deducts the corresponding funds from the user's account, the lock is completed and returns the result. If the system receives the hash key for unlocking within the t_2 time, the unlocking program is run, and if it is still not unlocked at the end of the locking time, the funds are returned in the same way.

3.2.2. Validation phase

After the user and the system have completed locking the assets in the source chain and the target chain respectively, the whole fund transfer process enters the verification stage. At this time, the user and the cross-chain system will verify the assets previously locked by the other party to ensure that the funds are locked according to the requirements, so as to ensure the security of the entire cross-chain fund transfer process. Firstly, the system will find the relevant two blockchains according to the source blockchain identifier ID_S and target blockchain identifier ID_T in the information submitted by the user, and then confirm whether the user has completed the locking of the funds on these two chains and whether the transaction amount accunt is equal to the number of funds locked in the smart contract; if the verification passes, then the unlocking phase will be carried out in the next step; if the verification fails, then the transaction will be canceled. In the transaction validation phase, how the system confirms that the funds have been correctly locked is a key part of ensuring the security and atomicity of cross-chain transactions. Specific validation criteria and logic can include the following.

- **Smart contract state checking:** The system first checks the status of the smart contract associated with the transaction to ensure that the contract has been correctly activated or is in the expected wait-locked state. This step ensures that the initialization of the transaction has been completed and that the smart contract is ready to receive and lock funds.
- **Validation of the amount of funds:** The system reads the amount of funds recorded by the smart contract and compares it with the amount of funds declared in the transaction request submitted by the user. The validation ensures that the funds the user has locked in the source chain correspond to the amount they wish to transfer to the target chain. This step prevents under or over locking of funds.
- **Reconciliation of hash lock information:** For cross-chain transactions that use a hash-time locking mechanism, the system needs to verify the hash-lock information provided by the user. This includes checking that the key of the hash submitted by the user matches the hash value recorded in the smart contract, and that it was submitted within the specified time-lock period. This step ensures that both parties to the transaction have complied with the pre-agreed transaction conditions.
- **Reconciliation of hash lock information:** For cross-chain transactions that use a hash-time locking mechanism, the system needs to verify the hash-lock information provided by the user. This includes checking that the key of the hash submitted by the user matches the hash value recorded in the smart contract, and that it was submitted within the specified time-lock period. This step ensures that both parties to the transaction have complied with the pre-agreed transaction conditions.
- **Verification of the accounts of both parties to the transaction:** The system checks the source and target account information of the locked funds to ensure that the funds were transferred from the correct account and will be transferred to the intended account. This step prevents misdirection of funds and unauthorized transactions.

Through these specific verification criteria and logic, the cross-chain transaction system can ensure that each transaction has strictly followed the preset rules and conditions, thus guaranteeing the security and atomicity of the transaction. Such a validation process helps build trust between the two transaction participants, while reducing the risk of fund loss due to misuse or malicious behavior.

3.2.3. Unlocking phase

After passing the verification stage of the information, such as the locking of funds, the unlocking of funds will begin. First, the system will provide the key of the hash lock in the smart contract in the source chain, and then the funds will be transferred directly to the locked account address of the system according to the setting of the smart contract. The specific unlocking process is shown in the Algorithm 4.

Algorithm 4 System unlocked assets.

```
Input: $r; H_{lock}; Acc_{s1}; Time_{lock2}$
Output: $unlock\ result$
 1: function UNLOCK()
 2:     $Time_{now} = getTimestamp()$
 3:     if $Time_{now} < Timelock2$ then
 4:         if $H_{lock} = H(r)$ then
 5:             Unlock contract assets
 6:             Add the assets in the $Acc_{s1}$ according to the accunt
 7:             return Unlock successfully
 8:         else
 9:             return Unlock failed
10:         end if
11:     else
12:         return Unlock failed
13:     end if
14: end function
```

Where H_{lock} is the hash value of the locked funds, Acc_{s1} is the account address of the system in the source chain, and $Time_{lock2}$ is the time of the time lock set in the source chain smart contract. First, the current time is obtained through the timestamp to verify whether it is within the range of the time lock, and if it is within the range, then the next step of verification is carried out; otherwise, the unlocking fails and the fund withdrawal process is carried out. Then determine whether *r* is the correct original hash-lock image by comparing the hash value $H(r)$ of the key *r* with H_{lock}. The asset in the smart contract can be unlocked after the verification is passed, and the funds will be transferred to the system's account address Acc_{s1} in the source chain. If the original image provided does not always agree with the hash lock then the unlocking fails.

After the system completes the unlocking, the user will get the original image R_{rec} of the hash lock through the smart contract, at which time the user needs to unlock the locked assets in the target chain before the specified time. As shown in Algorithm 5.

Where H_{lock} is the hash value of the locked funds, Acc_{t2} is the user's account address in the target chain, and $Time_{lock1}$ is the time of the time lock set in the target chain smart contract. After obtaining the current time through the timestamp and verifying that it is at a valid time, use the acquisition of the original image R_{rec} to calculate its hash value H_{lock} and compare it with to

Algorithm 5 Users unlocking assets.

```
Input: $R_{rec}; H_{lock}; Acc_{t1}; Time_{lock1}$
Output: $unlock\ result$
 1: function UNLOCK()
 2:     $Time_{now} = getTimestamp()$
 3:     if $Time_{now} < Timelock1$ then
 4:         if $H_{lock} = H(R_{rec})$ then
 5:             Unlock contract assets
 6:             Add the assets in the $Acc_{t2}$ according to the accunt
 7:             return Unlock successfully
 8:         else
 9:             return Unlock failed
10:         end if
11:     else
12:         return Unlock failed
13:     end if
14: end function
```

determine its validity. If the comparison is consistent, then the unlocking is successful, and the funds will be transferred to the user's account address Acc_{t2} in the target chain, and then the entire cross-chain fund transfer is successfully completed. If the original image provided is inconsistent with the hash lock, then the unlocking fails.

3.2.4. Handing of exceptions

During the transaction process, there are some abnormal situations that do not follow the expected process, such as when the user did not unlock the asset within the time specified or failed to unlock it in time due to the system being affected by network fluctuations and equipment failures. First of all, the system will monitor whether the transaction is completed within the specified time through the time lock mechanism built into the smart contract. If the locked funds are not correctly unlocked within the set time window, the smart contract will automatically trigger a timeout exception. During the initialization phase of the transaction, the system verifies the signature submitted by the user to ensure the authenticity of the transaction request and the identity of the user. If the signature verification fails, the system will immediately mark the transaction as abnormal. In addition, during the locking phase, the system will check whether there are sufficient funds in the user's account to complete the transaction. If there are insufficient funds, the system will block the transaction from execution and log the insufficient funds exception.

In response, the system will perform a withdrawal algorithm for these situations that lead to transaction failure. The specific process is shown in Algorithm 6.

Algorithm 6 Asset withdrawal.

```
Input: $Time_{lock1}; Time_{lock2}$
Output: $Withdraw\ result$
 1: function WITHDRAW ()
 2:     $Time_{now} = getTimestamp()$
 3:     if $Time_{now} > Timelock1$ then
 4:         Unlock contract assets
 5:         Return the assets to the original account
 6:         return Withdraw successfully
 7:     else
 8:         if $Time_{now} > Timelock2$ then
 9:             Unlock contract assets
10:             Return the assets to the original account
11:             return Withdraw successfully
12:         else
13:             return Withdraw failed
14:         end if
15:     end if
16: end function
```

Where $Time_{lock1}$, $Time_{lock2}$ is the time of the time lock set in the target and source chains respectively, and $Time_{lock1} > Time_{lock2}$. If the current time $Time_{lock1}$ is exceeded, it is determined that the time lock in the target chain has timed out, and then the system will unlock the funds locked by both parties and return them to the original account. If the current time $Time_{lock1}$ is not exceeded, but $Time_{lock2}$ is exceeded, it is determined that the time lock in the source chain has timed out, and the system will also unlock the funds locked by both parties and return them to the original account.

3.3. Cross-chain data interaction

The cross-chain data interaction scheme in this paper utilizes side-chain data storage smart contracts to achieve efficient and secure cross-chain data interaction. The scheme includes steps such as the definition of data formats and contract interfaces, data submission on the sidechain, a data verification mechanism on the mainchain, and data synchronization between the mainchain and the sidechain. By adopting smart contracts, trigger mechanisms, and regular updates, the scheme proposed in this paper ensures the reliability, integrity, and consistency of cross-chain data interactions.

3.3.1. Data formate definition

Data format definitions define the structure and fields of cross-chain data, allowing data to be accurately understood and parsed between different chains. Current cross-chain data interaction schemes can use different data structures for representation, such as JSON (JavaScript Object Notation), XML (eXtensible Markup Language), or binary formats. In this scheme, we choose to use JSON as the data structure representation due to its simplicity, readability, and wide support. The following are the data structure field definitions for the cross-chain data in this paper:

(1) $DataId$: Unique identifier of data for data access and synchronization in the sidechain data store contract.
- Add_{sender}: The sender of the data, recording the address from which the data originated.
- $Add_{receiver}$: The recipient of the data identifies the destination address of the data.
- $Timestamp$: Timestamp of data, recording when the data was generated.
- $Payload$: The payload of data contains the actual data content. The specific content of the data is located in the payload field, which can be defined and extended according to specific needs, and the header of the payload field contains the hash value of the data content, which is used for integrity verification of the data.
- $Signature$: Digital signatures of data are used to verify the integrity of the data and the trustworthiness of the source.

To ensure data integrity, we use the following hash calculation formula Eq (7) to generate a hash of the data:

$$H(data) = Hash(Payload) \tag{7}$$

where, $H(data)$ represents the hash value of the data, which is calculated by applying the hash function $Hash$ to the $Payload$. This ensures the integrity and consistency of the data during transmission.

3.3.2. Data submission

When there is data generated that needs to be shared, the sender of the data can invoke the sidechain data storage contract's $contractInstance.submitData(dataId, Add_{sender}, Add_{receiver}, timestamp, payload, signature)$ function for data submission, the incoming parameters include all the data structure fields defined earlier. The data is passed in as a string and is parsed and stored according to the agreed data format. After the contract receives the data, it is verified and processed accordingly. The data validation mechanism can include verification of the legitimacy, integrity, digital signature, etc. of the data to ensure that only valid data can be accepted and stored. The contract can use the storage mechanism to save the data persistently for subsequent access and synchronization operations.

3.3.3. Data validation

After the sidechain submits the data to be shared, the data storage contract will verify the uploaded data through the data validation mechanism to ensure that only valid data can be accepted and stored. The process of data validation can include a data legitimacy check, integrity verification, and digital signature verification.

After the data is submitted, the contract first checks the legitimacy of the submitted data to verify that the data meets the agreed format and field requirements. The contract uses methods such as conditional statements and regular expressions to validate the data and ensure that the data meets the agreed format or field requirements. If the validation fails, the contract will refuse to accept and store the data and return a corresponding error message to the sender.

The contract then verifies the integrity of the data to ensure that it has not been tampered with or lost during transmission. The contract verifies the integrity of the data by performing hash calculations on the data using a hash function and comparing it to the hash value of the data provided by the sender in the header of the payload field. We verify the integrity of the data by using the Eq (8):

$$IntegrityCheck(DataHash, StoredHash) = True/False \tag{8}$$

If the hash value of the data does not match the hash value provided by the sender, the contract determines that the data has been tampered with and refuses to accept and store the data.

Finally, the contract will verify the digital signature of the data using the Eq (9):

$$Verify(signature, Data, PK) = True/False \tag{9}$$

where $Signature$ is the digital signature of the data, $Data$ is the original data, PK is the public key of the sender. If the signature verification is successful, then return $True$, otherwise return $False$, indicating that the data may have been tampered with or the sender is not trustworthy.

The sender can sign the data using a private key, and the receiver can verify the validity of the signature using the sender's public key. The contract can verify the digital signature of the data when it is submitted to ensure the trustworthiness of the data. If the signature verification fails, the contract can refuse to accept and store the data.

Table 1
Configuration information of the nodes.

Experimental environment	Configuration
CPU	Intel(R) Core(TM) i5-12400F CPU @ 2.50GHz
Memory	16GB
Hard disk	935GB SSD
Operating system	Linux Ubuntu 22.04
Network bandwidth	1000M

3.3.4. Data access control

Cross-chain data access is an important part of the cross-chain data interaction scheme. By calling the interface function of the side-chain data storage contract, the access and acquisition of cross-chain data can be realized on the main chain. The smart contract on the main chain can call the contract function and pass the corresponding parameters to obtain specific data. The contract can decide whether to provide access to the data based on the permissions and verification mechanisms.

In order to ensure data security and privacy, cross-chain data access should be set up with appropriate access control mechanisms. Contracts set different levels of access privileges and decide whether to provide access to data based on user roles or authentication mechanisms. Only users or contracts with appropriate permissions can successfully access cross-chain data.

The cross-chain data interaction system will define different roles and corresponding permission levels, for example, multiple levels of access roles such as administrator roles (admin) and ordinary user roles (users), and at the same time, roles in different side chains will be assigned corresponding permission levels for each role based on relevance.

Permission modifiers (modifiers) are used in contracts to restrict access to functions. Permission modifiers can be used to determine whether or not to execute a function based on a role's permission level. Only roles with the appropriate permission level can successfully execute functions modified by modifiers. The access rights to the corresponding functions are restricted through the onlyAdmin and onlyUser permission modifiers. Only the contract deployer (admin) can call the PerformRestrictedOperation function, while only addresses added to the user list can call the PerformNormalOperation function. Before accessing cross-chain data, the system will perform data validation on the access request. The contract can verify the validity, integrity, and legitimacy of the data to ensure that only validated data can be accessed and used. In addition, the contract can record the access log of cross-chain data, including the identity information of the visitor, access time, access data, etc., so as to trace and monitor the data access behavior.

4. Experimental analysis

To validate the effectiveness of the cross-chain interaction model proposed in this paper and assess its performance, a blockchain experimental environment is simulated and constructed using an Ethernet platform. Three sets of experiments are conducted within this environment by establishing two private Ethernet blockchains. The methodology outlined in this paper is implemented within the experimental setup, with the algorithmic code stored locally on all Ethernet nodes to facilitate verification. Configuration details for the nodes are provided in Table 1.

This paper conducts experiments with three distinct variable groups. Firstly, we establish one main chain and several side chains of varying types to assess the performance of the proposed scheme against traditional methods under different transaction loads. Secondly, we analyze the interaction success rates of our approach and traditional methods in the presence of numerous malicious nodes attempting attacks. Lastly, we compare the communication overhead between our proposed cross-chain data interaction scheme and traditional methods when facilitating data exchange between the main and side chains.

For the purpose of comparison, we selected two widely recognized traditional schemes that have been instrumental in facilitating cross-chain interactions prior to our innovation: Notary Schemes: These involve a set of trusted validators (notaries) that oversee and facilitate transactions between different blockchain networks. While offering simplicity and direct interoperability, their reliance on centralized entities for validation introduces potential security vulnerabilities and centralization risks. Sidechains: Sidechains are auxiliary blockchains that run parallel to the main blockchain, allowing for asset and data transfer between the sidechain and the main chain. This method leverages a two-way peg mechanism, enabling interoperability but often at the cost of increased complexity and security challenges related to the sidechain's security and consensus mechanism. We will use these traditional cross-chain schemes in our experiments to compare with the scheme in this paper to highlight the superiority of the scheme in this paper.

4.1. Performance testing

Since the two-way pegged cross-chain transaction system used for the fund transfer of the main and side chains in this paper's scheme is realized entirely through the smart contract on the chain without waiting for the confirmation of other parties, users can quickly complete the fund transfer and realize the efficient cross-chain interaction of the main and side chains. In order to verify the fund transfer efficiency of this paper's scheme and its performance under different transaction loads, we build a main chain and multiple side chains with different purposes through simulation experiments, and configure the two-way pegged transaction system between the main chain and the side chains to ensure that the fund transfer and data interaction functions are available, and then carry out fund transfers between the main and side chains, and record the submission time and confirmation time of each transaction, and then start to gradually increase the number of transactions to simulate the performance of the system under different loads, and

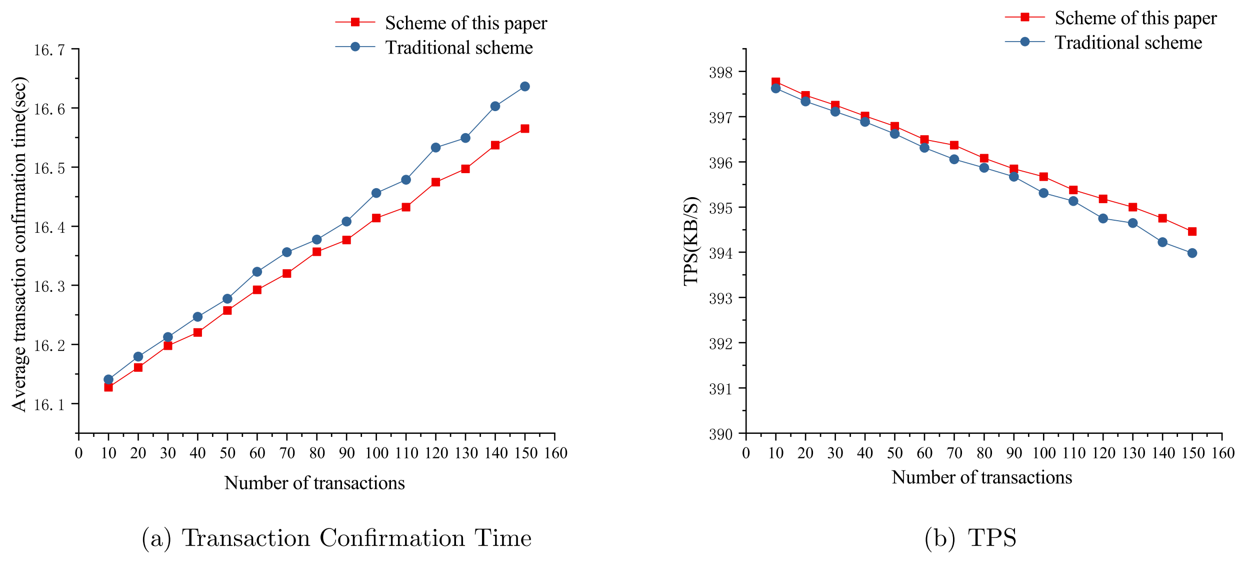

Fig. 3. Performance test experimental results.

Table 2
Malicious attack test results.

Test Scenario	Proposed Method	Traditional Method
Malicious Data Submission Test		
Number of Attempts	20	20
Successful Submissions	0	10
Malicious Data Detected	20	0
Data Tampering Test		
Number of Attempts	15	15
Successful Tampering	0	8
Tampering Detected	15	0
System Stability Test		
Number of Malicious Nodes	50	50
Normal Transaction Time	Normal	Slightly Delayed
Defensive Test		
Number of Attempts	20	20
Successful Attacks	0	12

finally obtain the system's transaction confirmation time, throughput and other indicators under different loads. The experimental results for transaction confirmation time are shown in Fig. 3(a), and the experimental results for throughput are shown in Fig. 3(b).

From the experimental results, it can be seen that the transaction confirmation time gradually increases with the increase in the number of transactions, and the system has a high throughput in the initial stage, but the throughput gradually decreases under high load. The scheme in this paper shows lower transaction confirmation time and reflects higher efficiency under each transaction number compared to the traditional scheme. In addition, this paper's scheme is able to maintain high throughput under high load and shows better scalability.

4.2. Malicious attack test

In order to verify the security of this paper's scheme in the process of cross-chain interaction, we test whether the system can recognize it effectively by setting up a malicious node in the blockchain and attempting to submit malicious data, as well as tampered data. In this paper, we first simulate the experiments of trying to submit malicious data using malicious nodes in this paper's scheme and the traditional scheme, and record the number of successful submissions. In addition this paper also simulates the data tampering test where the malicious node tries to tamper with the submitted data and records the results of the system detection. The results of the experiment are shown in Table 2.

Experimental results show that this paper's scheme successfully identifies and blocks malicious data submission, detects data tampering, and performs well in defending nodes. In contrast, the traditional scheme has a high level of malicious data submission and data tampering, exposing its lower security performance. This scenario will verify the reliability and security of the scheme in this paper and provide practical data support for the security performance of cross-chain interactions.

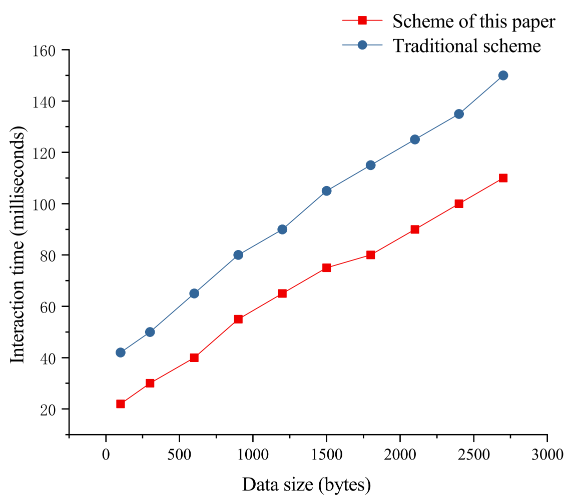

Fig. 4. Data interaction overheads.

4.3. Data interaction overhead test

In order to verify the effectiveness of this paper's scheme in cross-chain data interaction, this paper records the communication overhead of the two schemes in different cases by simulating data interaction between the main side chains. First, we will choose different sizes of data for testing, submit these data to the main chain smart contract for verification and storage, and record the time required for each interaction. Then, we will simulate scenarios with different interaction frequencies, measure the time for the main chain smart contract to process the data and the side chain to submit the data, and calculate the average communication overhead. The experimental results are shown in Fig. 4.

From the above data, it can be observed that the proposed scheme in this paper has relatively low transaction size and transaction cost during data interaction, while the traditional scheme performs poorly in these aspects. This indicates that the scheme in this paper has an advantage in terms of communication overhead and is able to realize cross-chain data interaction between the main chain and side chains more effectively. This result is consistent with our expectations and proves the superiority of the cross-chain data interaction scheme proposed in this paper in terms of communication overhead.

5. Conclusion

This paper focuses on resolving the challenge of enabling interaction between digital content and assets across various blockchains within a multi-chain ecosystem. By building upon traditional sidechains and incorporating technologies such as smart contracts and hash time locks, it not only resolves security and efficiency concerns but also achieves decentralized transaction processes, mitigating third-party risks and improving transaction speeds. Additionally, the solution emphasizes data interchange security and consistency, ensuring data trustworthiness and privacy protection during cross-chain processes. Through this technological implementation, distinct blockchain systems can efficiently share data and assets, achieving true blockchain interoperability. Finally, this paper validates the proposed solution through performance testing and malicious attack experiments, demonstrating its effectiveness. In the future, we will focus on optimizing the scalability of the solutions in this paper and conducting comprehensive tests in real-world scenarios to address these challenges. And more in-depth research on optimizing data exchange protocols and security frameworks to ensure robust protection against evolving cyber threats and to adapt to the increasingly complex blockchain ecosystem. We will also target scalability enhancements to ensure that our solutions can handle a wide range of data interactions without compromising performance.

CRediT authorship contribution statement

Lili Cheng: Software, Methodology, Writing – original draft. **Zhiying Lv:** Writing – review & editing, Writing – original draft, Methodology, Investigation, Data curation. **Osama Alfarraj:** Resources, Funding acquisition, Writing – review & editing. **Amr Tolba:** Writing – original draft, Software, Data curation. **Xiaofeng Yu:** Supervision, Project administration, Funding acquisition. **Yongjun Ren:** Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant No. 72372074, L2324126 and 62072249) and the Researchers Supporting Project Number (RSP2023R102) King Saud University, Riyadh, Saudi Arabia.

References

[1] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, H. Wang, Blockchain challenges and opportunities: a survey, Int. J. Web Grid Serv. 14 (4) (2018) 352–375.
[2] M. Belotti, N. Božić, G. Pujolle, S. Secci, A vademecum on blockchain technologies: when, which, and how, IEEE Commun. Surv. Tutor. 21 (4) (2019) 3796–3838.
[3] Q. Wu, Z. Han, G. Mohiuddin, Y. Ren, Distributed timestamp mechanism based on verifiable delay functions, Comput. Syst. Sci. Eng. 44 (2) (2023).
[4] M. Krichen, M. Ammi, A. Mihoub, M. Almutiq, Blockchain for modern applications: a survey, Sensors 22 (14) (2022) 5274.
[5] Y. Ren, Y. Leng, J. Qi, P.K. Sharma, J. Wang, Z. Almakhadmeh, A. Tolba, Multiple cloud storage mechanism based on blockchain in smart homes, Future Gener. Comput. Syst. 115 (2021) 304–313.
[6] W. Ou, S. Huang, J. Zheng, Q. Zhang, G. Zeng, W. Han, An overview on cross-chain: mechanism, platforms, challenges and advances, Comput. Netw. 218 (2022) 109378.
[7] Y. Jiang, C. Wang, Y. Wang, L. Gao, A cross-chain solution to integrating multiple blockchains for iot data management, Sensors 19 (9) (2019) 2042.
[8] B. Pillai, K. Biswas, V. Muthukkumarasamy, Cross-chain interoperability among blockchain-based systems using transactions, Knowl. Eng. Rev. 35 (2020) e23.
[9] Y. Ren, F. Zhu, J. Wang, P.K. Sharma, U. Ghosh, Novel vote scheme for decision-making feedback based on blockchain in Internet of vehicles, IEEE Trans. Intell. Transp. Syst. 23 (2) (2021) 1639–1648.
[10] H. Jin, X. Dai, J. Xiao, Towards a novel architecture for enabling interoperability amongst multiple blockchains, in: 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), IEEE, 2018, pp. 1203–1211.
[11] M. Madine, K. Salah, R. Jayaraman, Y. Al-Hammadi, J. Arshad, I. Yaqoob, appxchain: application-level interoperability for blockchain networks, IEEE Access 9 (2021) 87777–87791.
[12] Monika, R. Bhatia, A. Jain, B. Singh, Hash time locked contract based asset exchange solution for probabilistic public blockchains, Clust. Comput. 25 (6) (2022) 4189–4201.
[13] A. Xiong, G. Liu, Q. Zhu, A. Jing, S.W. Loke, A notary group-based cross-chain mechanism, Digit. Commun. Netw. 8 (6) (2022) 1059–1067.
[14] R. Huo, S. Zeng, Z. Wang, J. Shang, W. Chen, T. Huang, S. Wang, F.R. Yu, Y. Liu, A comprehensive survey on blockchain in industrial Internet of things: motivations, research progresses, and future challenges, IEEE Commun. Surv. Tutor. 24 (1) (2022) 88–122.
[15] A.S. Yadav, N. Singh, D.S. Kushwaha, Sidechain: storage land registry data using blockchain improve performance of search records, Clust. Comput. 25 (2) (2022) 1475–1495.
[16] Y. Ren, Y. Leng, Y. Cheng, J. Wang, Secure data storage based on blockchain and coding in edge computing, Math. Biosci. Eng. 16 (4) (2019) 1874–1892.
[17] Y. Ren, F. Zhu, P.K. Sharma, T. Wang, J. Wang, O. Alfarraj, A. Tolba, Data query mechanism based on hash computing power of blockchain in Internet of things, Sensors 20 (1) (2019) 207.
[18] A. Alkhodair, S. Mohanty, E. Kougianos, D. Puthal, Mcpora: a multi-chain proof of rapid authentication for post-blockchain based security in large scale complex cyber-physical systems, in: 2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), IEEE, 2020, pp. 446–451.
[19] Y. He, C. Zhang, B. Wu, Y. Yang, K. Xiao, H. Li, Cross-chain trusted service quality computing scheme for multi-chain model-based 5g network slicing sla, IEEE Int. Things J. (2021).
[20] P.P. Ray, N. Kumar, D. Dash, Blwn: blockchain-based lightweight simplified payment verification in iot-assisted e-healthcare, IEEE Syst. J. 15 (1) (2020) 134–145.
[21] Y. Ren, K. Zhu, Y. Gao, J. Xia, S. Zhou, R. Hu, X. Feng, Long-term preservation of electronic record based on digital continuity in smart cities, Comput. Mater. Continua 66 (3) (2021).
[22] Y. Ren, D. Huang, W. Wang, X. Yu, Bsmd: a blockchain-based secure storage mechanism for big spatio-temporal data, Future Gener. Comput. Syst. 138 (2023) 328–338.
[23] P. Han, Z. Yan, W. Ding, S. Fei, Z. Wan, A survey on cross-chain technologies, Distrib. Ledger Technol., Res. Pract. 2 (2) (2023) 1–30.
[24] T. Nolan, Alt chains and atomic transfers, online forum post, https://bitcointalk.org/index.php?topic=193281.0, 2013.
[25] J. Seo, J. Kim, Enhancing scalability with payment requests aggregation in lightning network, in: 2022 IEEE International Conference on Blockchain (Blockchain), 2022, pp. 340–347.
[26] Y. Ren, Z. Lv, N.N. Xiong, J. Wang, Hcnct: a cross-chain interaction scheme for the blockchain-based metaverse, ACM Trans. Multimed. Comput. Commun. Appl. (2023).
[27] B. Vitalik, Chain interoperability, R3 Research Paper 9, https://allquantor.at/blockchainbib/pdf/buterin2016chain.pdf, 2016.
[28] J. Li, X. Duan, W. Qiu, P. Cao, Z. Wang, Y. Li, A cross-chain transaction model for power blockchain based on hash-locking mechanism, in: 2023 3rd International Conference on Energy, Power and Electrical Engineering (EPEE), 2023, pp. 1273–1278.
[29] B. Rexha, V. Neziri, R. Dervishi, Enhancing trustworthiness and interoperability of electronic voting systems through blockchain bridges, HighTech Innov. J. 4 (4) (2023) 749–760.
[30] H. He, Z. Luo, Q. Wang, M. Chen, H. He, L. Gao, H. Zhang, Joint operation mechanism of distributed photovoltaic power generation market and carbon market based on cross-chain trading technology, IEEE Access 8 (2020) 66116–66130.
[31] G. Scaffino, L. Aumayr, Z. Avarikioti, M. Maffei, Glimpse: on-demand pow light client with constant-size storage for defi, in: 32nd USENIX Security Symposium (USENIX Security 23), 2023, pp. 733–750.
[32] J. Dilley, A. Poelstra, J. Wilkins, M. Piekarska, B. Gorlick, M. Friedenbach, Strong federations: an interoperable blockchain solution to centralized third-party risks, arXiv preprint, arXiv:1612.05491, 2016.
[33] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, P. Wuille, Enabling blockchain innovations with pegged sidechains, 72 (2014) 201–224, http://www.opensciencereview.com/papers/123/enablingblockchain-innovations-with-pegged-sidechains.
[34] E. Foundation, Consensys, BTC-relay: Ethereum contract for bitcoin spv, https://github.com/ethereum/btcrelay, 2015.
[35] J. Kwon, E. Buchman, Cosmos whitepaper, A Netw. Distrib. Ledgers 27 (2019).
[36] G. Wood, Polkadot: vision for a heterogeneous multi-chain framework, White Paper 21 (2327) (2016) 4662.
[37] Z. Zhang, W. Li, H. Liu, J. Liu, A refined analysis of Zcash anonymity, IEEE Access 8 (2020) 31845–31853.