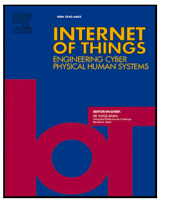
Internet of Things 28 (2024) 101339

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot



Research article

A fair multi-attribute data transaction mechanism supporting cross-chain

Kuan Fan [a,b], Chaozhi Zhou [a], Ning Lu [a,b,*], Wenbo Shi [a,b], Victor Chang [c,*]

[a] School of Computer Science and Engineering, Northeastern University, Shenyang 110004, China
[b] Hebei Key Laboratory of Marine Perception Network and Data Processing, Northeastern University at Qinhuangdao, 066004, Hebei Province, China
[c] Department of Operations and Information Management, Aston Business School, Aston University, Birmingham, UK

ARTICLE INFO

Keywords:
Fairness
Data transaction
IoT data
Multi-attribute
Cross-chain

ABSTRACT

Reliable storage and high-speed data networks enable individuals to access high-quality Internet of Things (IoT) data for scientific research through global transactions. Blockchain technology provides transparency for institutions to securely store and manage IoT data, while cross-chain transaction mechanisms facilitate the flow of IoT data. However, fairness issues may arise when it comes to cross-chain transactions of IoT data. This paper proposes a mechanism for multi-attribute data transactions to support cross-chain. The solution utilizes Vickrey–Clarke–Groves (VCG) auction, Paillier, Intel SGX, and other technologies to design a secure and equitable data seller selection scheme. The scheme ensures that the selection process for data sellers is both informed and private. Additionally, we generate a key pair for each attribute in the dataset to produce the corresponding attribute data signature. The dataset's legitimacy is verified through batch verification to ensure that the data seller's purchased attributes align with their requirements. The exchange of crypto assets and private keys between data sellers and buyers is designed to achieve fair payment. Our research suggests that the scheme meets the necessary security standards, and simulation results confirm its feasibility and effectiveness.

1. Introduction

Access to high-quality IoT data is a fundamental requirement for advancing scientific research across various disciplines, improving scholars' work efficiency. This data is typically obtained from experiments or real-world scenarios. Currently, the availability of reliable and cost-effective storage solutions, combined with high-speed interconnection networks, enables the worldwide use of IoT data in targeted markets [1,2].

Due to their integrity, accuracy, and immutability, IoT data is often stored in blockchain systems. This approach ensures traceability of any changes made to the data, thus attributing responsibility to potential tampering attempts. However, the insularity of blockchain networks creates barriers to collaborative operation between different blockchains. Therefore, implementing cross-chain technology is crucial for establishing a robust market for high-quality IoT data and has significant practical implications.

The most crucial aspect of a transaction is the principle of fairness. For the context of this research, a fair data cross-chain transaction is defined as one in which both buyers and sellers of the data are content with the transaction once it is concluded. Consequently, the data marketplace ensures that the data buyer has access to a satisfactory data set and that the data seller is able to match it with the appropriate data buyer. The seller and buyer then negotiate the price of the data. Once both parties have

* Corresponding authors.
E-mail addresses: luning@neuq.edu.cn (N. Lu), v.chang1@aston.ac.uk (V. Chang).

https://doi.org/10.1016/j.iot.2024.101339
Received 13 April 2024; Received in revised form 12 June 2024; Accepted 21 August 2024
Available online 26 August 2024

reached an agreement on the price, the transaction can be initiated. The data seller can receive the token, while the data buyer can download the data at any time. In cross-chain data transactions, both parties to the transaction conceal their identities, particularly during the selection process of data sellers, which can prevent the occurrence of malicious competition. In order to realize the above data cross-chain fair transaction, we have designed the data market architecture based on oracle, and use oracle nodes to match buyers and sellers and verify the rationality of data. However, achieving equity in the aforementioned cross-chain data transactions presents three challenges that need to be addressed.

Challenge 1 Fairness of opportunity: A robust data market must provide comprehensive search, discovery, and matching services. Compliance with industry regulations [3,4] is paramount, particularly in ensuring the security and privacy of patient data. The design of transaction opportunity fairness protocols presents the challenge of finding the most suitable data seller (DS) for the data buyer (DB), while protecting the information of both parties from leakage. Decentralized Exchanges (DEX) adopt the order book and liquidity pool models to improve the speed of transaction matching and the acceleration of cash flow [5]. The order book model uses smart contract buyers to match orders based on demand and execute asset transfers between parties quickly. Although this model boasts fast transaction speeds, optimizing its matching strategy for multiple orders meeting a buyer's criteria remains an area for improvement [6], given its vulnerability to single-point failures. In contrast, the liquidity pool model involves liquidity providers locking their tokens into smart contracts to create a pool from which they earn trading fees [7]. However, the sheer volume of data makes this model impractical for data and token exchange, as it is not possible to lock data into smart contracts. Furthermore, while both models use smart contracts for transaction processing, these contracts ensure transaction transparency but do not address privacy concerns. Some articles propose auctioning strategies to address equity concerns, but these protocols often overlook security and privacy issues during the auctioning process [8–10].

Challenge 2 Transaction dataset fairness: To attract DB, DS may sometimes exaggerate the size and attributes of their datasets in the data market. Unfortunately, this can result in buyers receiving datasets that differ from what was understood. To prevent this, DS must provide accurate IoT data metadata to the data market, which is then verified to ensure alignment with the actual data size, content, and attributes. This verification process aims to address the issues mentioned above. Current solutions mainly use Boneh–Lynn–Shacham (BLS) or RSA signature mechanisms to create homomorphic verifiable tags, coupled with bilinear technology to verify data content integrity [11–14]. Currently, there are no established consistency verification schemes specifically for data size or attributes. This limitation arises due to the nature of multi-attribute data, where data comprises attribute values. Digital signatures and similar technologies can verify data consistency if attribute values are present. However, the challenge arises from attributes being abstract characteristics derived from these attribute values. As there is no direct data representation for attributes, it is unfeasible to employ cryptographic primitives for attribute calculation and label verification. Furthermore, confirming the consistency of the data scale in isolation lacks contextual relevance. It is more meaningful to verify that a dataset's size, meeting specific conditions, aligns with the disclosed data size in the metadata. Therefore, developing a mechanism to ensure alignment across attributes, scale, and content presents a significant challenge in achieving fairness within transactional datasets.

Challenge 3 Fairness of payment: During the exchange of data and tokens, the DS usually stores the encrypted IoT data on the server, while the DB tokens are locked in a smart contract. Upon completion of the transaction, the seller has immediate access to the tokens, while the buyer can download the IoT data at any time. Therefore, DS can terminate the transaction early after receiving DB tokens, resulting in the loss of DB's token. Conversely, if DB launches a DDOS attack on DS after receiving the data address, DS may lose data. Solutions such as the Hash Time Lock Protocol (HTLC) and notary mechanisms aim to ensure fairness in multiple token exchanges [15]. HTLC uses a hash lock and time lock to guarantee the atomic execution of multiple operations, solving the problem of cross-chain asset exchanges [16,17]. However, due to the non-exclusive nature of HTLC's commit and rollback operations, the mechanism remains susceptible to DDOS attacks [18]. While setting longer time locks may resist such attacks in practical applications, it inevitably leads to delays and reduced efficiency [19]. This article advocates for IoT data and token swaps but recognizes the limitations of directly using HTLC, which introduces long delays and significantly reduces transaction efficiency. Alternatively, the notary mechanism involves cross-chain participants selecting a notary collectively to ensure transaction fairness [20]. However, this solution relies heavily on the selected notary and remains vulnerable to single points of failure. Although a multi-notary mechanism can mitigate this problem [20,21], the differences in node composition and security levels between different blockchains pose challenges in identifying multiple credible notaries. As a result, ensuring fair exchange specificity between data and tokens within transactions remains a significant challenge.

This paper proposes a cross-chain transaction fairness framework tailored for multi-attribute datasets. It introduces a keyword auction protocol based on the Vickrey–Clarke–Groves (VCG) mechanism to ensure fair trading opportunities [22]. The paper presents Paillier encryption, Intel Software Guard Extension (SGX), and Pedersen Promise, among other techniques, to protect IoT data from malicious eavesdropping or tampering during the auction process. A consistent verification protocol for data attributes, size, and content has been developed to ensure transaction fairness. The protocol operates throughout the registration and request phases. It validates data size against metadata during registration using bilinear pairing techniques and hash functions. In the data request phase, distinct key pairs are generated for each attribute, signing the respective attributes. While using batch verification, the protocol significantly enhances verification efficiency and confirms the consistency of data attributes and content. In terms of payment equity, we address the differences between data and token exchange mechanisms and formulate an impartial data and token exchange protocol. In this agreement, both parties encrypt assets, and use double hash locking technology to exchange encrypted assets and keys. This approach transforms IoT data and token exchange into a secure exchange of private keys and tokens, mitigating delay and data security issues. Recognizing the vulnerability of the hash time lock mechanism to DDOS attacks, the protocol links two private keys that are essential for decrypting the encrypted assets. Successful decryption of crypto assets requires both parties to be in possession of the secret key. The main contributions of this paper are as follows:

This paper proposes a cross-chain transaction fairness framework tailored for multi-attribute data sets. It introduces a keyword auction protocol based on the Vickrey–Clarke–Groves (VCG) mechanism to ensure fair trading opportunities [22]. The paper also discusses several techniques, including Paillier encryption, Intel Software Guard Extension (SGX), and Pedersen Promise, to protect IoT data from malicious eavesdropping or tampering during the auction process. A verification protocol has been developed to ensure transaction fairness by checking data attributes, size, and content. The protocol is consistently applied during both the registration and request phases. Data size is validated against metadata during registration using bilinear pairing techniques and hash functions. In the data request phase, distinct key pairs are generated for each attribute and used to sign the respective attributes. The protocol enhances verification efficiency through batch verification while ensuring consistency of data attributes and content. Regarding payment equity, we address the differences between data and token exchange mechanisms and formulate a data and token exchange protocol. Both parties encrypt assets and use double hash locking technology to exchange encrypted assets and keys. This paper presents an approach that transforms the exchange of IoT data and tokens into a secure exchange of private keys and tokens, which helps to address latency and data security concerns. To mitigate the vulnerability of the hash time lock mechanism to DDOS attacks, the protocol links two private keys that are necessary for decrypting the encrypted assets. Successful decryption of crypto assets requires both parties to possess each other's secret key. The paper's main contributions are:

(1) This paper presents a cross-chain transaction architecture for multi-attribute IoT data using oracles. Oracle is positioned as an off-chain data trading platform that offers essential services such as order matching and data availability verification. Once the order is successfully matched, the DS and DB can engage in data and token exchange, facilitated under the supervision of the trading platform.

(2) This section analyzes three fairness concerns related to the multi-attribute cross-chain data transaction architecture. To ensure fairness in trading opportunities, we propose a robust and verifiable auction mechanism based on the VCG mechanism. This mechanism will employ smart contracts, SGX, and various security primitives. In addition, a protocol for auditing multi-attribute data consistency has been designed. This protocol uses batch audits to ensure consistency across data attributes, scale, and content, thereby ensuring the availability of transaction data. A two-round hash time lock scheme is proposed by our research. The scheme supports IoT data and token trading while ensuring data privacy and transaction fairness.

(3) Furthermore, a comprehensive safety analysis and performance evaluation of the proposed system has been conducted. The security analysis shows that the system meets the expected security standards, while the performance evaluation confirms that the system is practical.

The remainder of this paper is organized as follows. Section 2 outlines the system and threat models, presents the privacy-preserving task assignment problem, and delineates the design goals. Section 3 addresses solutions to three distinct fairness issues. Our security and privacy analyses and experimental evaluations are presented in Sections 4 and 5. Section 6 reviews related works, culminating in the conclusions drawn in Section 7.

2. System overview

2.1. System model

As shown in Fig. 1, we present the general architecture of multi-attribute data transactions in cross-chain scenarios. There are four entities: (1) Sell Chain, (2) Buy Chain, (3) Data Market, and (4) Cloud Server. Their roles in the system are described as follows.

- Data Sell Chain (DS): An organization or business that possesses a multi-attribute IoT dataset and manages it through a blockchain can earn tokens by selling this data.
- Data Buy Chain (DB): Organizations or individuals who manage data using blockchain technology. They purchase IoT datasets through data marketplaces. Once payment is successful, the purchased data set can be downloaded at any time.
- Data Market (DM): An entity is responsible for recording the metadata of the IoT dataset, verifying the availability of IoT datasets, and helping DS and DB match orders without compromising dataset privacy.
- Cloud Server (CS): An entity with rich storage and computing capabilities that stores the ciphertext of IoT dataset.

Fig. 1 shows the transaction process of the IoT dataset in the cross-chain scenario. The DS sends the metadata to the DM, who then publishes it. He then encrypts the dataset and sends it to the cloud server. The DB sends the conditions for purchasing the data set to the DM. DM finds the appropriate DS for DB after receiving it. Once DB finds DS, they will exchange tokens for the download address of the dataset.

This paper utilizes blockchain technology to construct a data market. Therefore, the blockchain in this paper can also be interpreted as representing the data market.

2.2. Desired goals

The design goals of a fair multi-attribute data transaction mechanism supporting cross-Chain are listed as follows:

(1) Fairness of Opportunity. We assume that DM finds y datasets according to the requirements of DB. DM selects the lowest-priced dataset based on bidding value. The selection process should meet the following requirements:

- *Identity anonymous. During the process of selecting DS, the identity of the DS cannot be known.*

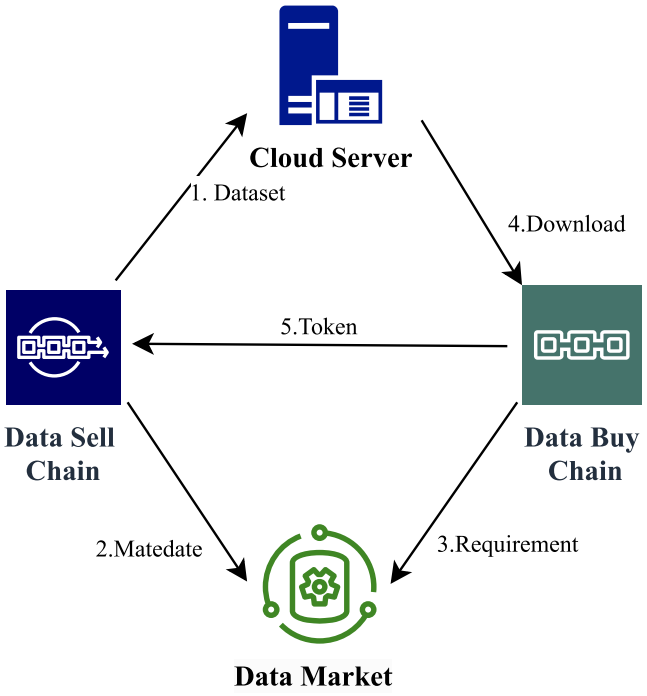


Fig. 1. System model.

- *Hide dataset prices. No one can calculate the bidding value for the dataset during the process of selecting DS.*
- *Verify matching results. After a successful selection of DS, anyone can verify these results.*

(2) Transaction Dataset Fairness. There are two scenarios for transaction dataset fairness:

- The metadata meets DB's requirements.
- The dataset received by the DB is consistent with the metadata published by the DM. This means that the dataset cannot be deleted or modified after the metadata has been published.

.

(3) Payment Fairness. The following requirements should be met for payment fairness

- DS can get the token paid by DB in time after successful payment.
- DB can download the purchased dataset anytime.
- During the transaction, the dataset cannot be seen by anyone other than DB and DS.

2.3. Threat and security model

Based on our above description, multi-attribute data transactions support cross-chain should satisfy the following security properties.

(1) Unforgeable: No one can forge data bidding during the auction. Once the auction is successful, anyone can verify the correctness of the auction results.
(2) Availability: If DS forges metadata, such as exaggerating data size or data attributes, DM can detect this malicious behavior through the verification mechanism, so that the IoT data received by DB is consistent with the data it requested.
(3) Atomicity: After successful payment, the DS can obtain the token, and the DB can download the IoT data at any time.
(4) Resist DDOS attacks: During the payment process, neither DB nor DS will launch DDoS attacks to undermine the fairness of payment.

We use a game between a challenger C and a probabilistic polynomial adversary A to demonstrate how the adversary A is against the proposed mechanism. This helps formalize the security model.

Definition 1. *During the transaction process, the DB cannot use forged datasets (size or attributes) to generate evidence and pass the consistency audit, destroying the availability of the mechanism*

Assuming that if A can forge the consistency proof with negligible probability and pass the availability audit, then the proof in the proposed mechanism is unforgeable.

1. **Initialization**: The proof unforgeable game between C and A is constructed. C constructs the algorithm B_A, and simulates the mechanism environment for A. A generates proof by inquiring B_A, and B_A verifies the proof. B_A and A are the verifiers and the prover, respectively.
2. **Query**: In this phase, A can make the hash queries and signature queries.
   - Hash queries: A queries B_A for the hash value of some data blocks(A_i, a_{ij}), and B_A calculates and sends it to A.
   - Signature queries: A queries the signature of the data block(A_i, a_{ij}), C runs the SigGen algorithm to calculate the signature of (A_i, a_{ij}), and sends to A.
3. **Challenge**: B_A checks A with a random challenge $Chal_t$ consisting of some blocks that have not been queried. According to $Chal_t$, A generates the corresponding signatures σ_t and proof $Proof_t$, and then returns them to C.
4. **Forged output**: If A can forge proof based on the challenge and pass the verification, then A wins the game.

2.4. System components

The transaction process can be divided into four stages: Registration, Order matching, Verification, and Payment. The transaction process is as follows:

Phase 1 Registration. DS generates dataset D's metadata $MData = (key_z, A_i, H(\prod_{j=1}^{m} a_{ij}), size)_{i\in[1,n]}$, where key_z represents the keywords of D, and $size$ can be expressed as $n \times m$, representing the size of the D. DS sends $MData$ to the DM. DM uses the "Challenge-Response" method to verify whether DS has the dataset of size $n \times m$. After verification, DM publishes $MData$.

Phase 2 Order Matching. After receiving the transaction request from data sellers, DM identifies y suitable data sellers. The Enclave service is then deployed by DM, which generates key (pk_i, sk_i) for each data seller. Data sellers encrypts the bidding value bv_i to $E_{pk_i}(bv_i)$, computes the bidding value's commitment $com(bv_i)$, and sends $E_{pk_i}(bv_i)$ and $com(bv_i)$ to DM. $E_{pk_i}(bv_i)$ is decrypted in Enclave by DM and the bv_i are sorted. The data seller with the lowest bv_i can trade with DB. The final transaction price is calculated by Enclave based on the impact factors β_i and bv_i. DB calls the Validation function $verification(com(bv_i), E(bv_i))$ to verify the correctness of the vb_i.

Phase 3 Verification. The data sellers who gain the right to trade generate n key pairs $\{sk_{A_i}, pk_{A_i}\}_{i\in[1,n]}$ for n attributes where private key is $sk_{Ai} = \alpha_i$, public key is $pk_{A_i} = g^{\alpha_i}$ and $\alpha_i \in [Z_p]$ is a random value. For each attribute, the data seller calculates the signature σ_{A_i} and generates the dataset signature set $\Phi = \{\sigma_i\}_{1\le i\le n}$. Once the signature set is received, the DM sends a challenge to the data seller, who then calculates the proof based on the challenge. The DM checks the availability of the dataset by verifying the correctness of the proof.

Phase 4 Payment. Once the dataset is available, data sellers and data buyers are ready to start the payment process. The payment process involves two rounds of asset exchange. In the first round, the data seller sends the encrypted dataset $E_{pk_{DS}}(E_{pk_{DB2}}(D))$ to a cloud server, while the data buyer sends the encrypted token $E_{pk_{DB1}}(Token)$ to the data seller. The public keys of the data buyer (pk_{DB1} and pk_{DB1}) and the public key of the data seller (pk_{DS}) are used in this process. Once the data seller verifies that the dataset is stored on the cloud server, the second round of exchange begins. The private key sk_{DS} is sent by the data seller to the data buyer, and the private key sk_{DB_1} is sent by the data buyer to the data seller. Once both parties have received their respective keys, they can access the token and the dataset.

The main notations and their descriptions are outlined in Table 1.

3. Proposed scheme

3.1. VCG-based secure data seller selection scheme

3.1.1. Overview

The system selects y DS that meets the requirements according to the search conditions. To choose the suitable DS, the scheme uses the Vickrey–Clarke–Groves(VCG) auction mechanism to determine the winner based on their impact factor and bidding value. This article deploys this solution into an oracle to demonstrate the transparency and legality of the auction process. The Paillier encryption scheme and Intel SGX technology [23,24] are used to protect the privacy of the bidding value, and Additionally, the Pedersen commitment is introduced to verify the correctness of the auction results.

The scheme framework is shown in Fig. 2. Each DS requests remote authentication services from Oracle nodes with TEE (TEE nodes) in DM and establishes a secure communication channel. DS sends the commitment of the bidding value to Oracle nodes through the Data Process Smart Contract(DP_{SC}) and the Oracle Smart Contract($Oracle_{SC}$). After the Oracle node listens to the Oracle contract event, DM obtains the encrypted bid value from the Oracle contract. It sends the ciphertext to TEE nodes for decryption, sorting, and calculation of the actual fee paid to the DS. The TEE node can determine the winner by calling the corresponding function written on the DM to execute the auction rules.

Table 1
Notations used in this paper.

Notations	Description
G, G_1	Multiplicative cyclic groups
p	Prime order of G, G_1
g	Generators ofG, G_1
$e : G \times G \rightarrow G_1$	The bilinear pairing
Z_p	Set of nonnegative integers less than p
$H : 0, 1^* \rightarrow G$	The secure map-to-point hash function
$h : 0, 1^* \rightarrow Z_p$	One way hash function
$r_i, u_i, rv_{ij}, w_{ij}, \mu$	random value
DP_{SC}	Data processing contract
D	Dataset
key_z	Keywords of the dataset
A_i	Dataset attributes
a_{ij}	Data block of A_i
n	Number of attributes
m	Number of data blocks for A_i
bv_i	Bidding value
F	Deposit
P	Dataset price
u_{id}/id	User real identity/User anonymous identity
τ_i	influence factor
k	Number of DB required attribute
b	Number of data blocks for attribute requested by DB
$chal_{size}/chal_{attr}$	Size challenge/Attribute challenge
$proof_{size}/proof_{size}$	Size proof/Attribute proof
$\Delta EC_{DB}/\Delta EC_{DS}$	Time delay
$\Delta Ek_{DB}/\Delta Ek_{DB}$	Time delay
ED/ET	Encrypt dataset/Encrypt token
$SC_{DB_{ET}}$	Lock encrypted token contract
$SC_{DS_{ED}}$	Lock encrypted dataset contract
$SC_{DB_{Ek}}$	Lock DB private key contract
$SC_{DS_{Ek}}$	Lock DS private key contract
χ and s	Secret value of Hash lock

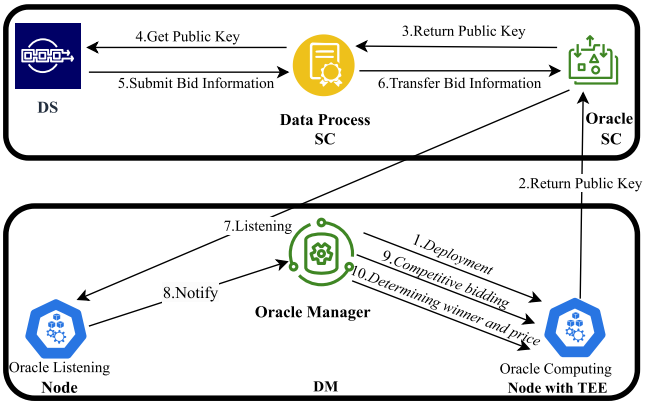


Fig. 2. Transaction dataset auction security model.

3.1.2. Protocol details

Assume that y data sellers $\{DS_1, DS_2, \ldots, DS_y\}$ participate in the bidding. After the auction ends, DM verifies the auction results and announces the final winner. Let τ_i represent the influence factor that affects DB's purchasing behavior, calculated by bid ranking. Suppose τ_i increases as the ranking decreases.

Step (1) Preparatory work

The preparation stage is mainly divided into two steps: deployment of smart contracts and remote authentication.

Step (1.1) Deployment of DM's auction service. First, g and h are the two generators of the cyclic group G, p is a large prime number, $q = 2p + 1$, which is also a large prime number. Second, the scheme initializes the auction parameters in DM's Oracle computing node. The scheme writes the auction and payment rules into the oracle of DM.

Step (1.2) Remote authentication. Each DS uses SGX's remote authentication mechanism to establish a secure communication channel and send information to the enclave security zone to compute sensitive information in a trusted execution environment. The implementation process of remote authentication is as follows:

- The DS can initiate remote authentication. When an enclave is created and started, we can use the Paillier to generate a key pair (epk_{DS_i}, esk_{DS_i}) in the enclave.
- The enclave issues authentication reports signed by the SGX hardware key. The report comprises two parts: the hash value of the enclave's initial contents and the list of data computed inside the enclave, including the public key, anonymous information of the DS, and encrypted bidding value. The data list and certification report are then sent to the DS.
- After receiving the certification report and the data list, the DS can publish them on the electronic bulletin board.
- Each DS can verify the certification reports and data list according to the Intel Attestation Service.

Step (2) Competitive bidding

During the bidding phase, DS's personally identifiable information is hidden using anonymous technology. Each DS's true identity can be expressed as u_{id}, while their anonymous identity id is calculated as follows:

$$id = E_{epk_{DS_i}}(u_{id}||pad)||epk_{DS_i}||y \tag{1}$$

where pad is a random padding bit that can pad the id to a certain length. DS_i encrypt the bv_i, and publishes $(E_{epk_{DS_i}}(bv_i), id)$ to the DM's electronic bulletin board. Then DS_i calculates the commitment value based on the extended Pedersen commitment using random value $r_i \in Z_p$.

$$Com(bv_i) = g^{bv_i} \ldots h^{r_i} \quad mod \quad p \tag{2}$$

DS_i keeps r_i by itself. Then the DS's anonymous identity id, the commitment value $com(bv_i)$ and deposit F_i are sent to data process smart contract DP_{SC} to save.

Step (3) Determining the winner and service price. The DM's oracle computing node is responsible for determining the winner and computing the price. There are three stages: decryption, sorting, and confirmation of service prices.

- Decryption. We use the private key esk_{DS_i} generated in the enclave to decrypt $E_{epk_{DS_i}}(bv_i)$ and get the bidding value bv_i.
- Sorting. After receiving $bv_1, bv_2, \ldots,$ and bv_y, DM sorts them in non-ascending order. The highest ranking is the auction winner.
- confirmation of service prices. When DM receives $bv'_1, bv'_2, \ldots, bv'_y$ and $y+1$ position impact factors $\tau_1, \tau_2, \ldots, \tau_y, 0$, it calculates the service price to be paid for all DS according to the final auction results. The calculation formula is as follows:

$$price = \frac{\sum_{i=1}^{y}(\tau_i - \tau_{i+1})bv'_{i+1}}{\tau_{winner}} \tag{3}$$

The DS_{winner} is required to publish the value of their bid bv_{winner} and random value r_{winner}. Any DS may be used to verify the accuracy of the auction results during the Public Verification Commitment phase.

Step (4) Public Verification Commitment The DS_{winner} publishes $(r_{winner}, bv_{winner})$. The DS computes the commitment value $Com'(bv_{winner}) = g^{bv_{winner}} \ldots h^{r_{winner}} mod p$, retrieves the commitment value $Com(bv_{winner})$ from the smart contract DP_{SC}. DS validates the accuracy of the auction results by comparing the two commitment values. During the verification process, the DS identity is anonymous, thus preventing the identification of the true identity of the DS_{winner}.

3.2. Multi-attribute data consistency audit scheme

3.2.1. Overview

The dataset obtained by DB is consistent with the metadata $MData$ on DM. The registration stage for data scale consistency adopts the "challenge-response" method. DM generates a challenge dataset $chal_{size}$ with the same size as the size in the $MData$. DS calculates the proof according to the $chal_{size}$ and sends it to the DM for verification. The success of the verification means that DS possesses a dataset of $n \times m$. The protocol defines the dataset structure as $D = (A_i : a_{ij})_{i\in[1,n],j\in[1,m]}$ to ensure the consistency of attributes and content. For each attribute A_i, the protocol generates a set of key pairs. The scheme utilizes A_i's private key to compute the BLS signature of a_{ij} and produce a homomorphically verifiable label. DM employs batch auditing techniques to verify the attribute and content consistency of datasets based on label homomorphism. Successful verification indicates that DS has a dataset of size $n * m$, and the attributes and content of the dataset are consistent with $MData$.

Fig. 3 depicts the dataset consistency audit model. The $MData$ is sent by the DB to the data processing contract, which then calls the oracle contract. The event is monitored by the listening node, which reports it to the Oracle management node. The management node sends the data size challenge task, $chal_{size}$, to the Oracle computing node. The size proof is calculated by the data processing contract based on $chal_{size}$. After listening to the evidence, the listening node sends it to the management node and assigns the computing node to verify the proof. Once the verification is successful, the management node records $MData$ in DM. When DM identifies a DS that satisfies the requirements of DB, the data processing contract computes the proof $Proof_{attr}$ using the attribute challenge $chal_{attr}$ generated by the computing node. Compute nodes audit data attributes and content consistency. The data obtained by the DB is available due to the consistency of data size, attributes, and content with $MData$.

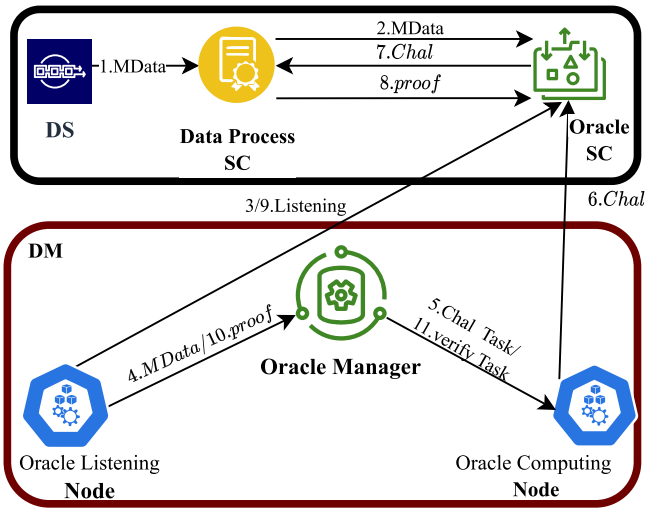


Fig. 3. Consistency audit model.

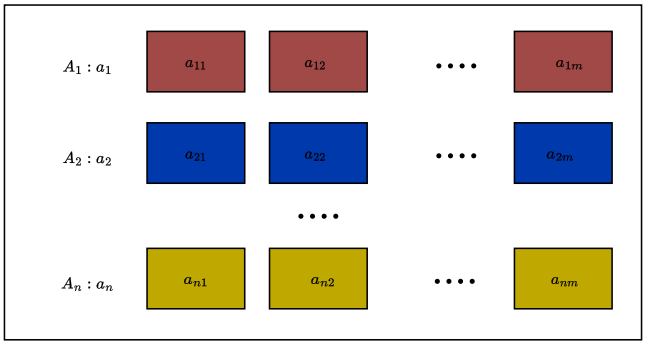


Fig. 4. IoT dataset structure.

3.2.2. Protocol details

The structure of the dataset involved in the scheme is defined as follows:

Definition 2 (*Multi-Attribute Dataset*)**.** Multi-attribute dataset $D = \{A_1 : a_1, A_2 : a_2, \ldots, A_n : a_n\}$, where A_i represents attributes such as age, sex, etc. a_i is the dataset under attribute A_i, which can be expressed as $a_i = a_{i1}, \ldots, a_{im}$, where m is the number of data items of a_i. n is the number of attributes of D, then its size is $n \times m$. The structure of the IoT dataset is illustrated in Fig. 4.

Fig. 5 shows the protocol sequence diagram, which has 9 phases. The protocol is described as follows:

Setup: Given a security parameter λ, it outputs the system public parameters $params = (p, g, G, G_1, H, h, e)$, where p is the large prime order of multiplicative cyclic groups G and G_1, g is the random generator of G and G_1, $e : G \times G \to G_1$ is a bilinear pairing, $H : \{0,1\}^* \to G$ is collision-resistant hash function.

KeyGen: For dataset $D = (A_i : a_{ij})_{i\in[1,n],j\in[1,m]}$, the DB randomly generates n signing key pair$(sk_i, pk_i)_{1\le i\le n}$, and chooses random values $\alpha_i \to Z_p$, calculate $v_i \to g^{\alpha_i}$. The private key for attribute A_i is α_i, and the public key is g^{α_i}.

SigGen: DB choose n random values $\{u_i \to Z_p\}_{1\le i\le n}$, and calculate the signature $\sigma_{ij} = (H(a_{ij}).u_i^{a_{ij}})^{\alpha_i}$ for each data block a_{ij}. The dataset D's signature set is $\Phi = \{\sigma_i\}_{1\le i\le n}$, where $\sigma_i = \prod_{j=1}^{m} \sigma_{ij}$.

RegisterChal: DB sends metadata $MData = \{attr_i, H(\prod_{j=1}^{m} a_{ij}), size\}_{1\le i\le n}$ to the data process smart contract DP_{SC}. DP_{SC} calls the oracle smart contract $Oracle_{SC}$. The Oracle listening node listens to the event and sends it to the oracle management node. Then, it delegates the oracle computing node to compute the challenge. it generates $n * m$ random number $\{rv_{ij}\}_{1\le i\le n,1\le j\le m}$ to form registration challenge $chal_{size}$. The oracle computing node writes $chal_{size}$ into the $Oracle_{SC}$ to register "Register Challenge event". The oracle node listens to the "Register Challenge event" and sends challenge $chal_{size}$ to the DP_{SC}.

RegisterProof: The DP_{SC} calculates dataset size proof $proof_{size}$ according to the following formula:

$$H_{attr_i} = \prod_{j=1}^{m} H(a_{ij})^{rv_{ij}} \tag{4}$$

The proof is $Proof_{size} = \{H_{attr_i}\}_{1\le i\le n}$. DP_{SC} returns $Proof_{size}$ to the $Oracle_{SC}$.

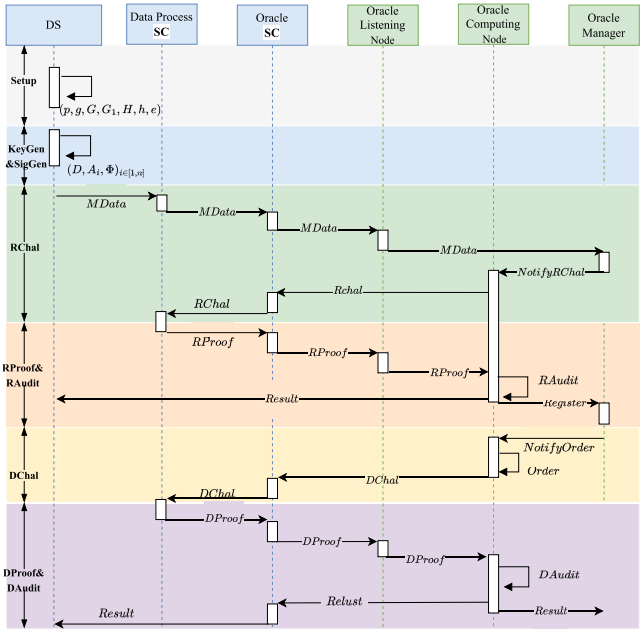


Fig. 5. Consistency audit sequence diagram.

RegisterAudit: The Oracle listening node monitors the evidence event and sends the event to the oracle management node. The management node sends the audit task to the oracle computing node. It calculates the verification equation according to the following formula:

$$e(\prod_{i=1}^{n}(\prod_{j=1}^{m} H(a_{ij})^{rv_{ij}}), g) \stackrel{?}{=} e(\prod_{i=1}^{n}(\prod_{j=1}^{m} H(a_{ij}), \prod_{i=1}^{n}(\prod_{j=1}^{m} g^{rv_{ij}}))) \tag{5}$$

If the equation is equal, it means that the DB have a dataset of size $n * m$.

ChalGen: The Oracle Manager receives the request $Reinfor = (attr_i, b)_{1\le i\le k}$, where x is the number of the data block under $attr_i$, and k is the number of required attributes. The oracle computing node randomly selects b integers and $k * b$ random values w_{ij} to generate challenge $chal_{attr} = (b, w_{ij})_{1\le i\le k, 1\le j\le x}$.

The oracle computing node registers the $Chal_{attr}$ as "Challenge Event" on the $Oracle_{SC}$. The oracle node listens to the challenge event and sends data requirements $Reinfor$ and $Chal_{attr}$ to the DP_{SC} according to the event requirements.

DataProofGen: After receiving $Chal_{attr}$ and $Reinfor$, DP_{SC} calculates the proof $proof_{attr}$ according to the following formula:

$$\begin{aligned} \mu_i &= \sum_{j=1}^{b} w_{ij}.a_{ij} \\ \xi_i &= \prod_{j=1}^{b} \sigma_{ij}^{w_{ij}} \end{aligned} \tag{6}$$

The proof is $proof_{attr} = (\mu_i, \xi_i)$.

DataVerify: The oracle computing node calculates the verification equation according to the following formula:

$$\begin{aligned} Pr1 &= e(\prod_{i=1}^{k} \xi_i, g) \\ Pr2 &= e(\prod_{i=1}^{k}(\prod_{j=1}^{b}(H(a_{ij})^{w_{ij}}).u_i^{\mu_i}), \prod_{i=1}^{k} g^{a_i}) \end{aligned} \tag{7}$$

If $Pr1 == Pr2$, it means that the attributes owned by DB are the same as the registered metadata, and it also means that the content of the registration data has not changed; otherwise, the transaction is terminated.

3.3. Fair and secure payment scheme for multi-attribute data transactions supporting cross-chain

3.3.1. Overview

Fig. 6 shows a payment model for data transactions. The gray line represents the smart contract called by DB, and the black line represents the smart contract called by DS. Fig. 6 contains a two-round hashing time-lock mechanism designed for exchanging assets.

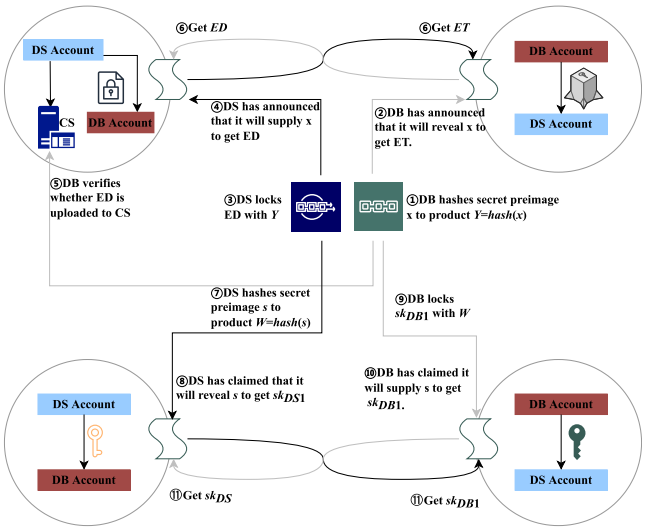


Fig. 6. Data transaction payment Model.

The first round involves the exchange of encrypted assets (steps 1–4), while the second round involves the exchange of decryption keys for ciphertext assets (steps 7–10). Both parties use the keys obtained in the second round to decrypt the encrypted assets after the two rounds of exchange. In addition, the DS sends the encrypted record to the CS, and the DB verifies that the record is sent to the CS before the second round.

3.3.2. Protocol details

Step (1) Initialization

Initialize system public parameters $params = (p, g, G, G_1, H, h, e)$, $h : \{0,1\}^* \rightarrow Z_p$ is homomorphic collision resistant hash function.

Pseudocode 1: Lock encrypted tokens(executed by the DB)

Input: Parameters $[params]$, secret χ, private key sk_{DB_i}

procedure Init($[params]$, χ,$[\alpha_i]_{i=1,2}$)
$sc_{DB\_ET} \leftarrow$ initialize($[params]$)
compute $pk_{DB} = [g^{\alpha_i}]_{i=1,2}$ ▷ generate public key
compute $sc_{DB\_ET}.ET = E_{pk_{DB1}}(Token)$
$sc_{DB\_ET}.Time_{Lock} = Time_{now} + \Delta_{EC_{DB}}$
pay(DB,$sc_{DB\_ET}, sc_{DB\_ET}.ET$) ▷ transfer DB encrypted tokens to sc_{DB_ET}
sc_{DB_ET}.commit $\leftarrow$ hash(χ) ▷ set secret commitment
sc_{DB_ET}.state $\leftarrow$ LOCK
return sc_{DB_ET}
end procedure

Step (2) Exchanging encrypted assets This stage comprises three functions deployed to smart contracts, as follows.

Step (2.1) Lock Encrypted Tokens: The secret value $\chi \in Z_p$, and the private key $[\alpha_i \in Z_p]_{i=1,2}$ are randomly selected by the DB, and the public key $pk_{DB_i} = [g^{\alpha_i}]_{i=1,2}$ is calculated. As it is shown in Pseudocode 1, DB encrypts the token with the public key pk_{DB_1}, and writes the encryption result into the smart contract sc_{DB_ET}. The sc_{DB_ET} assigns χ as a hash-locked secret value and sets a time lock, where $\Delta_{EC_{DB}}$ is the time delay of DB. Once the above operations are complete, sc_{DB_ET} is set to a locked state.

Step (2.2) Lock Encrypted Dataset: DS randomly selects $\beta \in Z_p$ as the private key sk_{DS}, and generates a public key $pk_{DS} = g^{\beta}$.

As shown in Pseudocode 2, when the state of SC_{DB_ED} is LOCK, DS initializes the smart contract SC_{DS_ED}. DS calculates the time lock $Time_{Lock}$, Where $\Delta_{EC_{DS}}$ is the time delay of DS. DS sets the commitment value $commit$ to be the same as in sc_{DB_ED}. DS encrypts the dataset $ED = E_{pk_{DS}}(E_{pk_{DB2}}(D))$ and computes the signature ω, where μ is a random number. DS uploads(ED, ω) to the CS, calculates the hash value of ED, and writes it into the smart contract sc_{DS_ED}. Once the above operations are complete, sc_{DS_ED} is set to a locked state.

Step (2.3) Obtain Encrypted Assets: Obtaining encrypted assets involves obtaining $H(ED)$ and ET. Pseudocode 3 outlines the process of obtaining $H(ED)$. When the state of SC_{DS_ED} is locked and the current time is within the legal range, DB uses χ to unlock SC_{DS_ED} and obtain $H(ED)$. The process of obtaining ET is similar to Pseudocode 3.

Pseudocode 2: Lock encrypted dataset(executed by the DS)

Input: Parameters [$params$], private key $sk_{DS} = \beta$
Output: sc_{DS_ED}

procedure Init([$params$],β)
$sc_{DS\_ED} \leftarrow$ initialize([$params$])
compute $pk_{DS} = g^{\beta}$ ▷ generate public key
if sc_{DB_ED}.state==LOCK **then**
  $sc_{DS\_ED}.Time_{Lock} = Time_{now} + \Delta_{EC_{DS}}$
  sc_{DS_ED}.commit=sc_{DB_ET}.commit
  compute $ED \leftarrow E_{pk_{DS}}(E_{pk_{DB2}}(D))$
  compute $\omega \leftarrow (H(ED) \cdot \mu^{h(EF)})^{\beta}$
  send (ED, ω) to cloud storage server
  pay(DS,$sc_{DS\_ED}, sc_{DS\_ED}.H(EF)$) ▷ Transfer hash value of ED to sc_{DS_ED}
  sc_{DS_ED}.state $\leftarrow$ LOCK
  return sc_{DS_ED}
end
end procedure

Pseudocode 3: Unlock encrypted token(executed by the DB)

Input: sc_{DS_ED}, secret χ

procedure WhistleblowSecret($sc_{DS\_EC,\chi}$)
if sc_{DS_ED}.state==LOCKED and time()< $sc_{DS\_ED}.Time_{Lock}$ **then**
  if $hash(\chi) = sc_{DS\_ED}.commit$ **then**
    obtain $sc_{DS\_ED}.H(ED)$
    sc_{DS_ED}.state $\leftarrow$ DISCLOSED
  end
end
end procedure

Step (3) Audit Dataset

After DB and DS obtain each other's encrypted assets, DB generates a random value $r \leftarrow Z_p$ and sends it to the CS. After receiving it, the CS calculates proof $\sigma = (H(ED) \cdot u^{h(ED)})^{\beta r}$, $\mu = r \cdot ED$, and sends (σ, μ) to the DB. DB verifies $e((H(ED) \cdot u^{h(ED)})^{\beta r}, g) \stackrel{?}{=} e(H(ED)^r \cdot u^{h(r \cdot ED)}, g^{\beta})$. If they are equal, it means that the CS has stored dataset, otherwise the transaction is aborted. **Step (4) Key Exchange**

This stage comprises three functions deployed to smart contracts, as follows.

Step (4.1) Lock encrypted dataset key: DS initializes the lock key smart contract sc_{DS_EK}, uses the public key pk_{DB1} to encrypt the key sk_{DS}, and writes the encrypted result into the smart contract sc_{DS_EK}. The sc_{DS_EK} assigns s as a hash-locked secret value and sets a time lock, where $\Delta_{Ek_{DS}}$ is the time delay of DS. Once the above operations are complete, sc_{DS_EK} is set to a locked state. The specific details are shown in Pseudocode 4.

Pseudocode 4: Lock dataset key(executed by the DS)

Input: SC parameters [$params$], secret s
Output: sc_{DS}

procedure Init([$params$], s)
$sc_{DS\_Ek} \leftarrow$ initialize([$params$])
compute $sc_{DS\_ES}.E_{sk_{DS}} \leftarrow E_{pk_{DB1}}(sk_{DS})$
$sc_{DS\_Ek}.Time_{Lock} = Time_{now} + \Delta_{Ek_{DS}}$
pay(DS,$sc_{DS\_Ek}, sc_{DS\_Ek}.E_{sk_{DS}}$) ▷ Transfer DB encrypted private key to sc_{DS_Ek}
sc_{DS_Ek}.commit $\leftarrow$ hash(s)
sc_{DS_Ek}.state $\leftarrow$ LOCK
return sc_{DS}
end procedure

Step (4.2) Lock encrypted token key:

As shown in Pseudocode 5, when the state of SC_{DB_EK} is LOCK, DB initializes the smart contract SC_{DB_EK}. DS calculates the time lock $Time_{Lock}$, Where $\Delta_{Ek_{DB}}$ is the time delay of DB. DB sets the commitment value $commit$ to be the same as in sc_{DS_Ek}. DS encrypts the private key of DB sk_{DB_1}, $E_{pk_{DS_1}(sk_{DB_1})}$ and writes it into SC_{DB_EK}. Once the above operations are complete, sc_{DB_Ek} is set to a locked state.

Pseudocode 5: Lock token key(executed by the DB)

Input: SC parameters [$params$]

procedure Init([$params$]) ▷ Create sc_{DS}
$sc_{DB\_Ek} \leftarrow$ initialize([$params$])
if sc_{DS_Ek}.state==LOCK **then**
    $sc_{DB\_Ek}.Time_{Lock} = Time_{now} + \Delta_{Ek_{DB}}$
    sc_{DB_Ek}.commit=sc_{DS_Ek}.commit
    compute $sc_{DB\_Ek}.E_{sk_{DB1}} \leftarrow E_{pk_{DS1}}(sk_{DB1})$
    pay(DB,$sc_{DB\_Ek}, sc_{DB\_Ek}.E_{sk_{DB1}}$)
    sc_{DB_Ek}.state $\leftarrow$ LOCK
end
end procedure

Step (4.3) Obtain private key:

Obtaining private key involves obtaining sk_{DB_1} and sk_{DS}. Pseudocode 6 outlines the process of obtaining sk_{DB_1}. When the state of SC_{DB_EK} is locked and the current time is within the legal range, DS uses s to unlock SC_{DB_EK} to obtain sk_{DB1}. The process of obtaining sk_{DS} is similar to Pseudocode 6.

Pseudocode 6: Unlock token key(executed by the DS)

Input: sc_{DB_ES}, secret s

procedure WhistleblowSecret(sc_{DB_ES},s)
if sc_{DB_ES}.state==LOCKED and time()< $sc_{DB\_ES}.Time_{Lock}$ **then**
    if $hash(s) = sc_{DB\_ES}.commit$ **then**
        obtain $sc_{DB\_ES}.E_{sk_{DB1}}$
        sc_{DB_ES}.state $\leftarrow$ DISCLOSED
    end
end
end procedure

4. Security analysis

4.1. Security analysis

In this section, we use mathematical techniques to demonstrate that our scheme is secure.

Theorem 1. *Under the computational CDH problem, our scheme is resistant to forgery attacks in data consistency audit.*

Proof. Suppose A can successfully forge some data blocks proof ξ, and the forgery-proof game is won by verification. In that case, B_A can solve the CDH hard problem.

Initialization: Given α_i is the private key of attribute A_i in the DS, g^{α_i} is the public key of A_i. Let $u = g^{\theta}$, where $\theta \in Z_p$ be the random value chosen by B_A. The input value of B_A are g^{α_i} and g^{β}, and then B_A can solve the CDH problem and output $g^{\alpha_i\beta}$.

Hash-Oracle: A queries B_A for the hash value $H(a_{ij})$ of data block a_{ij}:

1. If a_{ij} is in the hash list $H = \{a_{ij}, H(a_{ij})\}$, B_A obtains data $\{k_0, i, j, a_{ij}, h_{a_{ij}}\}$ from the list, and then reply A $H(a_{ij}) = h_{a_{ij}}$.
2. If a_{ij} is not in the hash list, B_A randomly selects $k_0 \in [0, 1]$, random value $r_{ij} \leftarrow Z_p$. When $k_0 = 0$, B_A calculates $h_{a_{ij}} = g^{r_{ij}}$. When $k_0 = 1$, B_A calculates $h_{a_{ij}} = (g^{\beta})^{r_{ij}}$. B_A store $\{k_0, i, j, a_{ij}, h_{a_{ij}}\}$ into the hash list, and reply $H(a_{ij}) = h_{a_{ij}}$

Signature-Oracle: To ensure that the interaction between B_A and A is identical to the actual attack, B_A maintains a signature list $sig = \{i, j, a_{ij}, \sigma_{ij}\}$ and responds to requests for the data block a_{ij} signature based on the signature list.

1. If the signature of a_{ij} is in the signature list, B_A obtains signature σ_{ij} from the list and then replies A the signature.
2. If the signature of a_{ij} is not in the signature list, B_A finds the hash list corresponding to $H(a_{ij})$. When the $H(a_{ij})$ does not exist in the hash table, the B_A queries the oracle again.

    - If the corresponding record is in the hash list and $k_0 = 0$, then B_A selects $H(a_{ij}) = g^{r_{ij}}$ according to the Hash-Oracle, and generates the signature as follows:

$$\sigma_{ij} = (H(a_{ij}) \cdot u^{a_{ij}})^{\alpha_i}$$

$$= (g^{\alpha_i})^{r_{ij}} \cdot (g^{\alpha_i})^{\theta(a_{ij})} \tag{8}$$

B_A adds the data $\{i, j, a_{ij}, \theta_{ij}\}$ to the signature list, and sends σ_{ij} to A.

- When $k_0 = 1$, the B_A refuses to respond to the corresponding signature to A.

Challenge: Suppose the B_A generation challenge $chal = \{(i, j, w_{ij}), 1 \le i \le n, 1 \le j \le k\}$. There is a tuple in the $chal$ that is not in the signature list.

Forged output: A generates a legal proof $\{\mu', \xi'\}$ depend on the $chal$. According to the formula (5) and formula (6), it can be known that:

$$e(\prod_{i=1}^{n} \xi', g) = e(\prod_{i=1}^{n}(\prod_{j=1}^{k} H(a_{ij})^{w_{ij}} \cdot u_i^{\mu'}), \prod_{i=1}^{n} g^{\alpha_i}) \tag{9}$$

At the same time, A does not request the signature of (i^*, j^*, w_{ij}^*) in the $chal$ from the Signature Oracle.

This means that the hash value $h_{a_{ij}}$ of (i^*, j^*, w_{ij}^*) can be found in the Hash list, and the signature list has no record for it. B_A queries the signature list to obtain signatures for other challenge values.

For other tuples from the $chal$, the B_A queries the signature list. If no corresponding record exists in the signature list, B_A queries the Hash-Oracle or Signature-Oracle. If $(i^*, j^*, w_{ij}^*), k_0 = 0$, B_A rejects the hash value $H(a_{ij})$, otherwise, B_A can solve the CDH problem.

In the Hash list, $k = 1$ for the challenge value (i^*, j^*, w_{ij}^*), and $k = 0$ for other challenge values, so the right side of formula (8) can be expressed as:

$$\begin{aligned} & e(\textstyle\prod_{i,j\in chal, i,j\neq i^*,j^*} (g^{r_{ij}})^{w_{ij}} \cdot \\ & \quad \textstyle\prod_{i,j\in chal} u_i^{\mu'} \cdot (g^{\beta r_{i^*j^*}})^{w_{i^*j^*}}, \textstyle\prod_{i,j\in chal} g^{\alpha^i}) \\ = \; & e((g^{\beta\alpha_i^* r_{i^*j^*}})^{w_{i^*j^*}}, g) \cdot \\ & e(g^{\sum_{i,j\in chal, i,j\neq i^*,j^*} r_{ij}\cdot w_{ij}\cdot\alpha_i} \cdot g^{\sum_{i,j\in chal, i\neq i^*, j\neq j^*} \theta_i\cdot\alpha_i\cdot\mu'}, g) \end{aligned} \tag{10}$$

The solution to the computational CDH hard problem is:

$$\begin{aligned} g^{\alpha_i^*\beta} = & (\xi' \cdot (g^{\sum_{i,j\in chal, i,j\neq i^*,j^*} r_{ij}\cdot w_{ij}\cdot\alpha_i} \cdot \\ & g^{\sum_{i,j\in chal, i,j\neq i^*,j^*} \theta_i\cdot\alpha_i\cdot\mu'})^{-1})^{\frac{1}{r_{i^*j^*}\cdot w_{i^*j^*}}} \end{aligned} \tag{11}$$

That said, B_A can solve the CDH problem with a non-negligible probability, but this contradicts the CDH difficulty problem, so our scheme can resist the forged proof attack initiated by A.

Theorem 2. *Under the computational DL problem, our scheme can prevent anyone from forging bidding value.*

Proof. The VCG-based secure data seller selection protocol can resist this attack during the public commitment stage. The commitment in random oracle mode is based on the assumption of discrete logarithmic difficulty. Even if an attacker has unlimited computing power, he cannot solve the corresponding discrete logarithm or tamper with the secret value. The secret value and the random number (b_i', r_i') are randomly selected. If there are two different open values for the commitment $com(b_i)$, namely (b_i', r_i') and (b_i, r_i), then:

$$\begin{aligned} com(b_i) &= g^{b_i'} \cdot h^{r_i'} mod\, p \\ &= g^{b_i} \cdot h^{r_i} mod\, p \\ &\Longrightarrow g^{b_i'-b_i} = h^{r_i-r_i'} mod\, p \\ &\Longrightarrow g = h^{\frac{b_i'-b_i}{r_i'-r_i}} mod\, p \end{aligned} \tag{12}$$

The above shows that $\log_h g$ can be calculated, which is contrary to the discrete logarithm problem. Therefore anyone cannot find the data tuple (b_i', r_i') to replace (b_i, r_i) to generate a valid commitment.

Theorem 3. *In the payment phase of the transaction, our scheme can resist DDOS attacks.*

Proof. Suppose that both DS and DB are rational traders. During the encrypted asset exchange process, DB uses the preimage χ to obtain DS's encrypted assets before the time lock expires. If DS encounters a DDOS attack, DB will not be able to obtain DS's encrypted assets. Since DB and DS exchange encrypted assets, but neither party has the key to decrypt the assets at this time, a rational DB will not launch an invalid DDOS attack. During the key exchange phase, we assume that the DB suffers a DDOS attack. According to the payment mechanism, DS cannot use DB's token before the time lock arrives. After the time lock expires, if DB does not obtain the one-time private key for DS encrypted data, the Token paid by DB will become invalid. Therefore, DB will not lose tokens, and DS data will not be illegally used. Moreover, if the third attacker obtains the preimage in a DDOS attack, then he can only obtain the encrypted assets or one-time key. Therefore, this protocol is resistant to DDOS attacks.

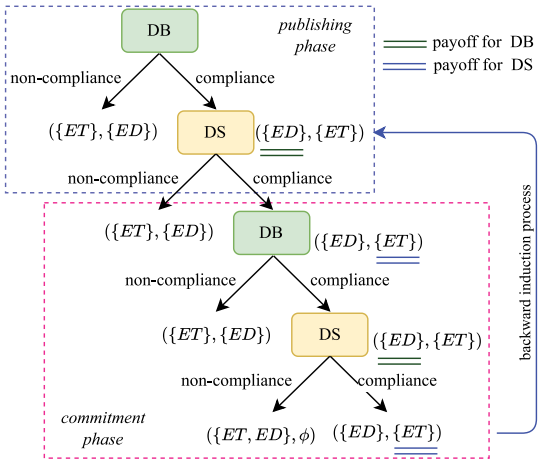


Fig. 7. Outcomes represent the payoffs owned by DB and DS, respectively.

Theorem 4. *The choice of "compliance" in our protocol is the only subgame perfect equilibrium.*

Proof. HTLC in our protocol is categorized into two phases: publishing and commitment phases. In addition, the publishing phase is required as a precondition for the commitment phase.

We consider two owners DB, DS swapping two assets ET, ED. DB, is the first player to select between "non-compliance" and "compliance". Whenever one of the two parties chooses "non-compliance" during the publishing phase due to the commitment requirement, the game terminates with the original ownership configuration as the outcome. If DB does not start the commitment phase, the outcome will coincide with the original ownership. When DS is called to decide, DB previous actions are known; if DS opts for "compliance", the swap occurs, otherwise DB acquires both the required asset and the originally owned one leaving DS empty-handed. The payoffs of DB and DS is presented in Fig. 7.

A backward induction process is applied to compute the subgame perfect equilibrium. Backward induction process is the reasoning from the end to the beginning of the game, the optimal payoff strategy is selected in each decision step. Then, considering the game associated to the protocol presented in this paper, the sequence of optimal actions is the one specifying compliance at each decision step (Fig. 7).

4.2. Security comparison

Compared to existing data transaction systems, our proposed cross-chain system has several security advantages. Table 2 shows that we use anonymous technology and VCG mechanisms to ensure fairness of opportunity. To ensure fairness of transaction data, we use bilinear technology to maintain data integrity and consistency. The system adopts cross-chain communication architecture, and to ensure the fairness of payment, the encrypted assets and the keys are exchanged through a two-round hash time-locking mechanism.

Table 2 shows that Chen et al.'s [25] work did not consider user and data privacy, Dai et al.'s [26] work did not guarantee data integrity, and He et al.'s [27] work primarily focused on tracking illegal data transactions rather than the data transaction itself. Guan et al. [28] and Guan et al. [29] proposed supporting smart contracts for big data trading. However, the data traded in these works are priced based on the degree of dataset matching, and the data buyer must pay for the corresponding output based on the size of the data, even if the data is invalid. None of the works mentioned implemented Oracle-based price bidding or fair trading of high-quality data for high data revenue.

5. Performance analysis

In this section, we evaluate the performance of the proposed scheme in terms of theory and experiment. We implement our scheme in Java and compare its performance with similar existing schemes. We build a simulation platform to supplement these analysis results and perform extensive experiments.

5.1. Theoretical analysis

In the VCG-based Secure Data Seller Selection Protocol, the primary computational overhead involves several key operations: Paillier encryption and decryption, utilization of the extended Pedersen commitment, and the frequency of function calls made by

Table 2
Security comparison.

	Anonymity	Data integrity	Fair trading	Bidding	High-quality data for high data revenue	Supporting cross-chain
Dai et al. [26]	✗	✗	✓	✗	✗	✗
Chen et al. [25]	✓	✓	✗	✗	✗	✗
He et al. [27]	✓	✓	✓	✗	✗	✗
Guan et al. [28]	✓	✓	✓	✗	✗	✗
Guan et al. [29]	✓	✓	✓	✗	✗	✗
Our proposed scheme	✓	✓	✓	✓	✓	✓

Table 3
Computational complexity.

Phases	Auction serve	DS
Preparatory work	Gen	–
Competitive bidding	–	$En, 2M_G$ $+1padd_G$
Determining the winner and service price	nDe	–
Public Verification Commitment	–	–

Table 4
Comparison of calculation costs.

Scheme	MHT	Our scheme
Initialization	$2n(H_{Z_p}) + E_{G+nk}$ $(H_G + M_G + 2E_G)$	$n(E_G + M_G) + 2E_G +$ $h_{Z_p} + nm(H_G +$ $M_G + 2E_G)$
Register auditing	$4P_G + 2H_G +$ $nm(H_G + 3E_G + 2M_G)$	$m(H_G + E_G) +$ $H_G + 2P_G + nm$ $(2H_G + 2E_G)$
Data auditing	$4P_G + 2H_G +$ $yb(H_G + 3E_G + 2M_G)$	$b(E_G + M_G) +$ $2P_G + 4E_G +$ $yb(H_G + E_G + M_G)$

the smart contract during the auction execution. We assume that there are n DS participating in the auction, and Gen represents Paillier key generation. En and De represent encryption and decryption, respectively.

The main computational overhead of the VGC-based secure data seller selection scheme includes encryption, decryption, extended Pedersen commitments, and the number of smart contract calls during auction execution. We assume that there are n DS participating in the auction, and Gen represents the Paillier key generation. En and De represent encryption and decryption, respectively. The auction server's enclave generates public and private keys by calling the Gen algorithm. Each DS calls the En algorithm to encrypt the bid value. The encrypted bid value will be decrypted n times before the winner and service price are determined. We implement the extended Pedersen commitment scheme using the elliptic curve secp256k1. Let the multiplication and addition operations on the elliptic curve be expressed as M_G and $padd_G$, respectively. As shown in Table 3, in the preparation work phase, the computational cost is mainly due to the auction server, which is Gen. In the competitive bidding phase, the computational cost mainly comes from the DS, each DS submits and performs the operation En and $2M_G + 1padd_G$. In determining the winner and service price phase, the auction server calculates the winner and the service price, the computational cost is nGe.

In Multi-Attribute Data Consistency Audit Protocol, We compare our scheme with the scheme MHT, which has the same functions. We divide the main process into three stages: initialization, register auditing, and data auditing. In Table 4, we compare the calculation costs of the MHT and our scheme. We assume that E_G, M_G, H_G and P_G represent the encryption overhead, multiplication overhead, hash calculation overhead and pairing calculation overhead on G, respectively. Hash cost of h_{Z_p} on Z_p. Our solution is to divide data blocks according to attributes, and use n keys to calculate the data block signature for each attribute, so the computational overhead in the initialization, registration audit, and data audit processes is slightly higher than MHT. In the initialization stage, our scheme requires $n[E_G + M_G - H_{Z_p}]$ more calculations than MHT. The computational overhead of the register auditing algorithm of our mechanism is $m(H_G + E_G) + H_G + 2P_G + nm(2H_G + 2E_G)$ and the data auditing overhead is $b(E_G + M_G) + 2P_G + 4E_G + yb$.

Payment latency denotes the duration between the initial locking of the first asset and the final unlocking of the last asset involved in a transaction. Yet, due to the absence of a dependable global clock within most blockchain systems, accurately measuring the

Table 5
Max latency.

Case \ Max Latency	DB	DS
DB and DS abide by the protocol	$\Delta_1 + \Delta_4 + t_v$	$\Delta_1 + \Delta_4 + t_v$
Encrypted asset stage DS stop payment after DB commit	$\Delta_1 + \gamma_{DB}$	None
Encrypted asset stage DB stop payment after DS commit	$\Delta_1 + \gamma_{DB}$	$\Delta_2 + \gamma_{DB}$
Key exchange phase DS stop payment after DB commit	$\Delta_1 + t_v + \Delta_3 + \gamma_{DB}$	$\Delta_1 + t_v + \Delta_4 + \gamma_{DS}$
Key exchange phase DB stop payment after DS commit	$\Delta_1 + t_v$	$\Delta_1 + t_v + \Delta_4 + \gamma_{DS}$

real latency of cross-chain payments presents challenges. This paper adopts a perspective centered on a participant blockchain client to measure latency. By doing so, apart from enhancing accuracy, the latency measurements offer a more precise reflection of the protocol's performance.

In this paper, the assumption is made that p represents a participant in the payment process. Three key stages are considered: Locked, Refund, and Redeem. When Locked = true, it signifies that p's assets are locked. Refund = true indicates that p has received the returned resource and can utilize it, while Redeem = true suggests p has received their reward and can access it. Let t_l denote the time when Locked = true, and t_r represent the time when Refund ∨Redeem = true. Consequently, the payment latency can be expressed as $\Delta_{latency} = t_r - t_l$.

Let Δ_1 be the $SC_{DB_{EC}}$ time-locked value, and Δ_2 be the $SC_{DS_{EC}}$ time-locked value, where $\Delta_1 > \Delta_2$. Let Δ_3 be the $SC_{DB_{EK}}$ time-locked value, and Δ_4 be the $SC_{DS_{EK}}$ time-locked value, where $\Delta_4 > \Delta_3$. We let γ_{DB} be the upper limit of DB payment confirmation time, γ_{DS} be the payment confirmation time of DS, and t_v be the verification time. We calculate the main time overhead in the verification process, then $t_v = 2H_G + 2H_{Z_p} + 2mul + P_G$. We assume that payments are maximally time-bound. We calculate the maximum latency of payment from two perspectives: DB and DS. Table 5 shows the maximum time latency of DS and DB. If both DS and DB obey the payment rules, then the maximum latency is $\Delta_1 + \Delta_4 + t_v$. If in the stage of exchanging encrypted assets, after DB obtains DS encrypted assets with secret value, DS terminates the payment, then he has to wait until Δ_1 expires before calling refund, and the refund will be in γ_{DB} confirmed in time. Suppose the phase of exchanging encrypted assets is when DB terminates the payment after DS uses the secret value to obtain DB encrypted assets. In that case, he has to wait until Δ_1 expires before calling refund, and the refund will be in γ_{DS} If it is confirmed within the specified time, DS will retrieve the refund after Δ_2 expires. For DB, the latency time is $\Delta_1 + \gamma_{DB}$, for DS, the latency time is $\Delta_2 + \gamma_{DS}$. If DS aborts the transaction during the key exchange stage, then DB has verified the authenticity of the data within t_v time, when DB stops the transaction after DS commits the secret value for DB and DS, the maximum latency is respectively For $\Delta_1 + t_v$ and $\Delta_1 + t_v + \Delta_4 + \gamma_{DS}$, when DB stops trading after DS confirms the secret value, for DB and DS, the maximum latency is $\Delta_1 + t_v + \Delta_3 + \gamma_{DB}$ and $\Delta_1 + t_v + \Delta_4 + \gamma_{DS}$.

5.2. Experimental analysis

This section describes the simulation experiments conducted to validate the proposed method's efficacy. The experiments were performed on an ASUS FX86SM laptop equipped with a 7th generation Intel(R) Core(TM) i7-8750H CPU and 16 GB of RAM. The operating system within the laptop environment was Ubuntu Linux 16.04, running on a VMware virtual machine. The project utilized the Intel SGX SDK version based on Linux 2.5, and the GCC compiler version 5.4.0. The primary development language used throughout the project was C.

The performance of our approach was evaluated using a well-recognized metric in cross-chain research, namely time cost. Specifically, we focused on the total time of a cross-chain transaction, which was $totalTime$. The process of cross-chain transactions involves three phases: $matchTime$, which is the time taken to find a suitable dataset for the database; $auditTime$, which is the time taken to verify the integrity of the dataset; and $swapTime$, which is the time taken for the exchange of assets between the database and the data source. These phases reflect the costs associated with cross-chain transaction protocols.

To assess Oracle's $matchTime$, the experiments employ the Intel SGX Autoconf development framework and leverage the trusted GMP Library project within this framework to construct a secure GMP library in SGX. It is imperative to ensure the reliability of the random source for the Paillier algorithm. Hence, the experiments utilize SGX's trusted hardware-based random numbers function $sgx\_read\_rand()$, which serves as the random source for the Paillier key. The Paillier algorithm relies on the pertinent arithmetic functions the trusted GMP library provides for the generation, encryption, and decryption operations involving the 1024-bit Paillier key. Furthermore, this paper introduces the libsecp256k1 library and implements a 512-bit extended Pedersen commitment scheme based on the $secp256k1$ curve. Detailed time consumption for phase $matchTime$ is presented in Table 7.

Fig. 8 depicts the consumption of enclave bidders during Paillier key generation, encryption, decryption, and extended Pedersen commitment. The experimental findings reveal that key generation demands more time compared to encryption and decryption, both of which consume nearly the same amount of time. The time taken for commitment, illustrated in Fig. 8, appears negligible when considering values of $n = 100, 200, 300, 400, 500$, registering times of 6.16 ms, 12.927 ms, 18.902 ms, 24.988 ms, and 31.376 ms, respectively.

Additionally, the time taken by VCG is depicted in Fig. 9. In the winner determination phase, the final winner is determined by finding the maximum value through a sorting function. Thus, as n increases, the time required for the winner determination phase increases accordingly.

Table 6
Comparison of different cross-chain solutions.

Experimental Environment	Max/s	Min/s	Mean/s
Simulation [30] (Ubuntu 20.04.1, AMD Ryzen5 4600H CPU with Radeon Graphics 3.00 GHz)	0.6	0.15	0.375
Fisco Bcos v2.8 (Ubuntu Server 20.04, AMD Ryzen7 5800H with Radeon Graphics 3.20 GHz, 16 GB memory) [31]	3.14	0.076	0.418
Simulation [32] (Intel core i5-4590, 3.30 GHz, 8.00 GB RAM)	7	3	5
Our mechanism	**14.9**	**2.7**	**8.36**
Cosmos sdk v0.42 to build blockchains [33]	11	9.94	10.47
Ethereum, truffle, Ganache [34] (Intel core i7 2.6 GHz CPU,16G RAM, Radeon Pro 555X 4 GB)	14	7	10.5
Ethereum [35] (DELL Power Edge T130, 4 IntelXeon CPU E3-1220 v5 @3.00 GHz 32 GB memory)	17	7	12
Consortium based on Ethereum [36], (IntelXeonE5 CPU 64 GB memory, Ubuntu64 os)	15	11	13

Table 7
The detailed time consumption for phase *matchTime* (ms)

Number of bidders \ Phase	Generation	Encryption	Decryption	Commitment	VCG
100	2353.757	107.863	105.963	6.16	0.013
200	4934.683	214.09	216.289	12.927	0.021
300	7113.625	319.747	316.756	18.902	0.033
400	9414.659	481.72	493.15	24.988	0.055
500	12 406.294	530.221	534.852	31.376	0.059

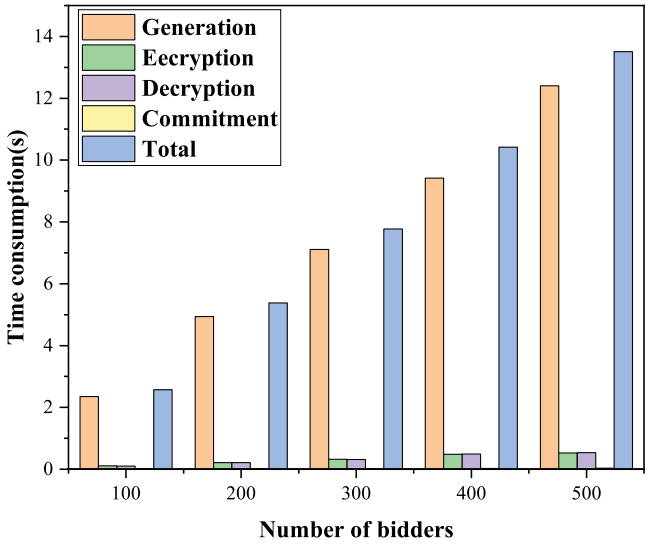


Fig. 8. The time consumption for the Paillier operations.

To evaluate Oracle's *auditTime*, we implemented an experiment using the PBC library. We chose the elliptic curve $BN254$ to construct a bilinear pair mapping with cyclic groups G_1 and G_2 of order 160 bits.

The experimental results show in Fig. 10 that **DataProofGen & DataVerify** is the primary time-consuming phase when the data volume is small. However, as the volume of data increases substantially, the **ChalGen** phase becomes the primary time-consuming phase. Since the **ChalGen** phase must generate challenges for each bidder, the time required increases with the number of bidders, i.e., it is close to $O(n)$, whereas in this paper we use a batch verification method, which is close to $O(1)$. These findings highlight the effectiveness of the auditing method used in this study.

Fig. 11 displays the time consumption variation for cross-chain scenarios with n values of 100, 200, 300, 400, and 500, and data sizes of 1 KB, 10 KB, 100 KB, and 1000 KB. Based on the analysis in Fig. 11, we can derive the effect of dataset size and the number of participating bidders on the total time. It is evident that the total time increases linearly with the increase of dataset and number of participating bidders.

We utilized OpenSSL's SHA256 to generate hash time locks for assessing Oracle's *swapTime*. The time taken for creation and redemption stood at 0.004 ms and 0.007 ms, respectively. Consequently, the cumulative time for two rounds of *swapTime* totaled 0.022 ms. Finally, we present the detailed data of *totaltime* under different conditions (numeber of bidder, datasize) in Table 8.

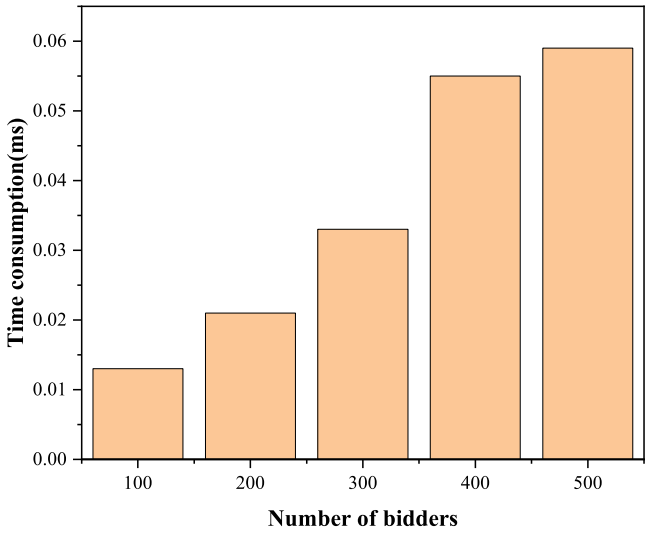


Fig. 9. The time consumption for VCG.

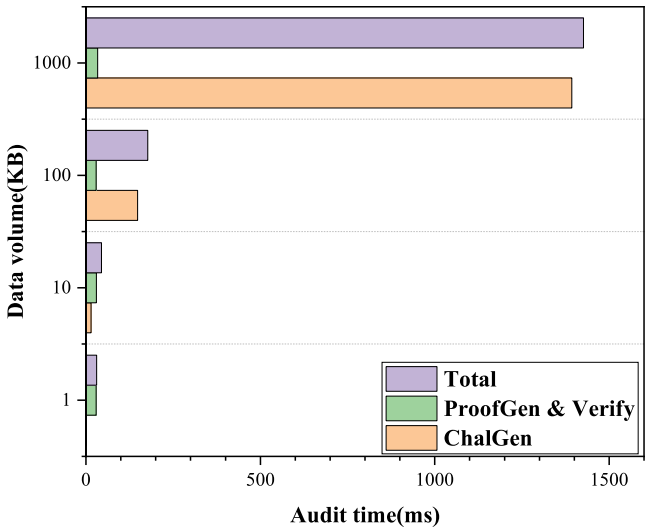


Fig. 10. The time consumption of the *auditTime*.

Table 8
The detailed time consumption for *totaltime* (s)

Datasize / Number of bidders	1 KB	10 KB	100 KB	1000 KB
100	2.7	2.71	2.85	4.1
200	5.4	5.42	5.55	6.8
300	7.8	7.81	7.94	9.2
400	10.44	10.5	10.59	11.84
500	13.53	13.55	13.68	14.93

5.3. Performance comparison

This paper systematically compiles and organizes results derived from various cross-chain methods to demonstrate the effectiveness of the proposed cross-chain transactions. The results are presented in Table 6, which shows that prevailing cross-chain methods exhibit an average time delay ranging from 0.375 s to 13 s across different blockchain platforms and experimental environments. Although the literature [30–32] reports shorter processing times than ours, they do not address the application of large-scale attribute sets in cross-chain transmission and are limited to transmitting small amounts of information. Our experimental results position the

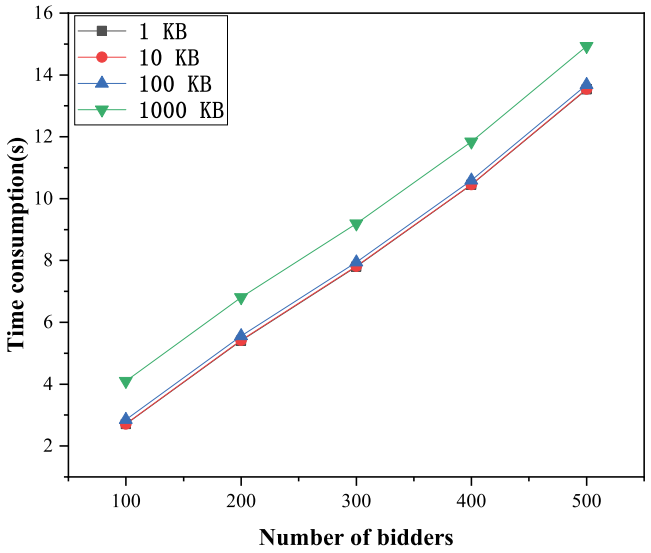


Fig. 11. The time consumption of the *totalTime*.

performance of the proposed cross-chain transaction system at the forefront, despite the diverse nature of blockchain underlying technologies and the variability inherent in cross-chain scenarios.

6. Related work

Fairness stands as the cornerstone of data transactions. Conventional approaches rely on a trusted third party (TTP) for tasks like data encryption or signing [37,38]. Nonetheless, this model exhibits vulnerabilities associated with a single point of failure. The emergence of blockchain technology has spurred the development of numerous fair data transaction frameworks proposed by scholars. These schemes aim to eliminate the reliance on a TTP within the data fair trading ecosystem [39]. To circumvent the TTP dependency, current research harnesses blockchain or smart contracts to establish a decentralized trading platform. This platform integrates cryptographic primitives like encryption, signatures, hashes, and more. Their implementation ensures transactional fairness, contributing to a more robust and equitable data exchange environment.

Xue et al. proposed a blockchain-based data transaction framework that emphasizes fairness and privacy protection [40]. The paper describes a model in which data buyers pay for the data they need, while data sellers receive certain tokens. However, the article highlights three critical concerns: data availability, privacy preservation, and ciphertext retrieval. Several studies focusing on transactional fairness also address usability, transactional privacy, and payment atomicity. For example, Liu et al. validated data availability through identity verification protocols [41]. Their solution includes a blockchain-driven authentication protocol to validate information contributors, complemented by a smart contract-based mechanism to detect and filter potential instances of fake information. This methodology effectively prevents unverified participants from disseminating erroneous data, thereby ensuring data authenticity and fairness. Numerous studies address the verification of data availability by authenticating the identity of transaction parties, as discussed in [42–44]. These works typically assume the legitimacy of the identity of the data sharer, and thus infer the legality and accessibility of the shared data.

An et al. developed a fair data trading framework tailored for Crowdsensing Data Trading (CDT) [45]. Their approach uses a blockchain-driven reverse auction (BRA) mechanism to allocate sensing tasks to data sellers, but lacks transactional privacy considerations. Islam et al. developed a scheme aimed at privacy-preserving news exchange [46]. To ensure transactional fairness, the article introduces an intermediary called “Block Cop (BC)”. If the seller fails to deliver documents after receiving the buyer’s funds, the buyer can report the incident to BC for fraud investigation. Li et al. constructed a fair transaction protocol that uses smart contracts to enhance participants’ privacy [47]. This protocol assumes the existence of a trusted data manager capable of verifying data authenticity. Using proxy re-encryption technology, the article delegates the task of double encryption of transaction ciphertext to the data manager, thereby ensuring enhanced privacy. Related studies [48–50] also use smart contracts to develop fair and privacy-centric data transaction mechanisms. These efforts consider aspects such as data and identity privacy for both transaction parties.

The exchange of data and funds must be atomic in fair transactions. This area of research has been extensively covered in existing literature. The concept of atomic swaps [51], which originates from HTLC introduced in the Lightning network [15], is often the basis of these studies. However, conventional atomic swaps mainly address the atomic exchange between cryptocurrencies and do not seamlessly integrate data with transactions. Chenli et al. proposed an atomic exchange framework based on secret sharing principles to achieve atomic data-token exchange. In their method, both the seller and the buyer perform atomic swaps during the

exchange phase. The buyer provides funds, and the seller provides the decryption key. However, the article does not address the importance of checking data availability, which can lead to the exposure of decryption keys during the exchange process.

The aim of our work is to address concerns about fairness in cross-chain data transactions. To achieve this, we have developed a data transaction platform that uses distributed oracles. Our research focuses on ensuring fairness in the selection of data sellers, verifying the availability of transaction data, addressing payment fairness concerns, and ultimately achieving fair cross-chain data transactions.

7. Conclusion

This paper presents a novel method for enabling equitable multi-attribute data transactions across various blockchain networks. Our framework establishes a decentralized marketplace that utilizes blockchain Oracle technology and devises an architecture that enables secure cross-chain trading of multi-attribute data. In our proposed scheme, data sellers compete fairly for trading opportunities, ensuring that data buyers receive information that is consistent with its representation in the data marketplace. After a successful purchase, buyers can download the acquired data at their convenience, while sellers promptly receive legitimate tokens. Our scheme has been substantiated through rigorous security validation to achieve fairness and robust security. The effectiveness of our proposed framework is supported by the theoretical scrutiny and empirical experiments.

Future research will concentrate on cross-chain transactions that facilitate the transfer of multiple data types, including image data and text data. The availability of these data types will be verified during the transaction. Furthermore, future work will also focus on the utilization of relay chains, witness networks, cross-chain bridges, and other methodologies to construct cross-chain transaction architectures, thereby enhancing the flexibility of the transaction model.

CRediT authorship contribution statement

Kuan Fan: Writing – original draft, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Chaozhi Zhou:** Writing – review & editing, Validation, Methodology, Investigation, Data curation. **Ning Lu:** Writing – review & editing, Supervision, Resources, Funding acquisition, Conceptualization. **Wenbo Shi:** Writing – review & editing, Supervision, Project administration, Methodology, Funding acquisition. **Victor Chang:** Writing – review & editing, Validation, Project administration, Funding acquisition, Formal analysis.

Declaration of competing interest

This is hereby certify that the paper is original, neither the paper nor a part of it is under consideration for publication anywhere else. Also, we have no conflicts of interest to disclose.

We confirm that the manuscript has been read and approved by all named authors.

Data availability

The authors do not have permission to share data.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Nos. 62072092, 62072093, 62102075); the Natural Science Foundation of Hebei Province, China (No. F2020501013); the Fundamental Research Funds for the Central Universities, China (No. 2023GFYD001, N2323023). Prof Chang's research is supported by VC Research (VCR 0000230) and the International Science Partnerships Fund (ISPF: 1185068545).

References

[1] H. Ghosh, Data marketplace as a platform for sharing scientific data, in: Data Science Landscape: Towards Research Standards and Protocols, Springer, 2018, pp. 99–105.
[2] R. Alvsvåg, A. Bokolo Jr., S.A. Petersen, The role of a data marketplace for innovation and value-added services in smart and sustainable cities, in: International Conference on Innovations for Community Services, Springer, 2022, pp. 215–230.
[3] A. Act, Health insurance portability and accountability act of 1996, Public Law 104 (1996) 191.
[4] A. Act, Health information technology (HITECH act), Index Excerpts Am. Recover. Reinvestment Act 2009 2009 (2009) 112–164.
[5] A. Lehar, C.A. Parlour, Decentralized exchanges, 2021, Available at SSRN 3905316.
[6] J. Xu, K. Paruch, S. Cousaert, Y. Feng, Sok: Decentralized exchanges (dex) with automated market maker (amm) protocols, ACM Comput. Surv. 55 (11) (2023) 1–50.
[7] Y. Wang, Y. Chen, H. Wu, L. Zhou, S. Deng, R. Wattenhofer, Cyclic arbitrage in decentralized exchanges, in: Companion Proceedings of the Web Conference 2022, 2022, pp. 12–19.
[8] B. An, M. Xiao, A. Liu, G. Gao, H. Zhao, Truthful crowdsensed data trading based on reverse auction and blockchain, in: Database Systems for Advanced Applications: 24th International Conference, DASFAA 2019, Chiang Mai, Thailand, April 22–25, 2019, Proceedings, Part I 24, Springer, 2019, pp. 292–309.
[9] J. Du, E. Gelenbe, C. Jiang, Z. Han, Y. Ren, Auction-based data transaction in mobile networks: Data allocation design and performance analysis, IEEE Trans. Mob. Comput. 19 (5) (2019) 1040–1055.

[10] D. An, Q. Yang, W. Yu, D. Li, Y. Zhang, W. Zhao, Towards truthful auction for big data trading, in: 2017 IEEE 36th International Performance Computing and Communications Conference, IPCCC, IEEE, 2017, pp. 1–7.
[11] K. Fan, M. Liu, G. Dong, W. Shi, Enhancing cloud storage security against a new replay attack with an efficient public auditing scheme, J. Supercomput. 76 (2020) 4857–4883.
[12] K. Fan, Z. Bao, M. Liu, A.V. Vasilakos, W. Shi, Dredas: Decentralized, reliable and efficient remote outsourced data auditing scheme with blockchain smart contract for industrial IoT, Future Gener. Comput. Syst. 110 (2020) 665–674.
[13] H. Han, S. Fei, Z. Yan, X. Zhou, A survey on blockchain-based integrity auditing for cloud data, Digit. Commun. Netw. 8 (5) (2022) 591–603.
[14] D. Liu, Z. Li, D. Jia, Secure distributed data integrity auditing with high efficiency in 5G-enabled software-defined edge computing, Cyber Secur. Appl. 1 (2023) 100004.
[15] J. Poon, T. Dryja, The bitcoin lightning network: Scalable off-chain instant payments, 2016.
[16] K. Narayanam, V. Ramakrishna, D. Vinayagamurthy, S. Nishad, Atomic cross-chain exchanges of shared assets, 2022, arXiv preprint arXiv:2202.12855.
[17] L. Lys, A. Micoulet, M. Potop-Butucaru, R-SWAP: Relay based atomic cross-chain swap protocol, in: Algorithmic Aspects of Cloud Computing: 6th International Symposium, ALGOCLOUD 2021, Lisbon, Portugal, September 6–7, 2021, Revised Selected Papers 6, Springer, 2021, pp. 18–37.
[18] I.S. Amiri, M.R.K. Soltanian, Theoretical and Experimental Methods for Defending Against DDoS Attacks, Syngress, 2015.
[19] S. Mathivanan, et al., A survey on resource inflated denial of service attack defense mechanisms, in: 2016 Online International Conference on Green Engineering and Technologies (IC-GET), IEEE, 2016, pp. 1–4.
[20] 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.
[21] 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.
[22] P.G. Sessa, N. Walton, M. Kamgarpour, Exploring the vickrey-clarke-groves mechanism for electricity markets, IFAC-PapersOnLine 50 (1) (2017) 189–194.
[23] J. Wang, N. Lu, Z. Gong, W. Shi, C. Choi, A secure double spectrum auction scheme, Digit. Commun. Netw. (2022).
[24] J. Wang, N. Lu, Q. Cheng, L. Zhou, W. Shi, A secure spectrum auction scheme without the trusted party based on the smart contract, Digit. Commun. Netw. 7 (2) (2021) 223–234.
[25] J. Chen, Z. Lv, H. Song, Design of personnel big data management system based on blockchain, Future Gener. Comput. Syst. 101 (2019) 1122–1129.
[26] W. Dai, C. Dai, K.-K.R. Choo, C. Cui, D. Zou, H. Jin, SDTE: A secure blockchain-based data trading ecosystem, IEEE Trans. Inf. Forensics Secur. 15 (2019) 725–737.
[27] Y. He, H. Zhu, C. Wang, K. Xiao, Y. Zhou, Y. Xin, An accountable data trading platform based on blockchain, in: IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE, 2019, pp. 1–6.
[28] Z. Guan, X. Shao, Z. Wan, Secure fair and efficient data trading without third party using blockchain, in: 2018 IEEE International Conference on Internet of Things (IThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), IEEE, 2018, pp. 1395–1401.
[29] Z. Guan, Y. Zhao, D. Li, J. Liu, Tbdct: A framework of trusted big data collection and trade system based on blockchain and tsm, in: 2018 IEEE International Conference on Smart Cloud (SmartCloud), IEEE, 2018, pp. 77–83.
[30] W. Wei, W. Juan, W. Ziyang, et al., A cross-chain solutions based on proxy network, in: 2021 18th International Computer Conference on Wavelet Active Media Technology and Information Processing, ICCWAMTIP, IEEE, 2021, pp. 89–93.
[31] S. Lu, J. Pei, R. Zhao, X. Yu, X. Zhang, J. Li, G. Yang, CCIO: A cross-chain interoperability approach for consortium blockchains based on oracle, Sensors 23 (4) (2023) 1864.
[32] Z. Wu, Y. Xiao, E. Zhou, Q. Pei, Q. Wang, A solution to data accessibility across heterogeneous blockchains, in: 2020 IEEE 26th International Conference on Parallel and Distributed Systems, ICPADS, IEEE, 2020, pp. 414–421.
[33] X. Wu, Cross-chain workflow model based on trusted relay, in: Proceedings of the ACM Turing Award Celebration Conference-China, 2021, pp. 49–53.
[34] C. Tan, S. Bei, Z. Jing, N. Xiong, An atomic cross-chain swap-based management system in vehicular ad hoc networks, Wirel. Commun. Mob. Comput. 2021 (2021) 1–14.
[35] H. Su, B. Guo, J.Y. Lu, X. Suo, Cross-chain exchange by transaction dependence with conditional transaction method, Soft Comput. (2022) 1–16.
[36] R. Qiao, X.-Y. Luo, S.-F. Zhu, A.-D. Liu, X.-Q. Yan, Q.-X. Wang, Dynamic autonomous cross consortium chain mechanism in e-healthcare, IEEE J. Biomed. Health Inform. 24 (8) (2020) 2157–2168.
[37] F. Bao, R.H. Deng, W. Mao, Efficient and practical fair exchange protocols with off-line TTP, in: Proceedings. 1998 IEEE Symposium on Security and Privacy (Cat. No. 98CB36186), IEEE, 1998, pp. 77–85.
[38] J. Zhou, D. Gollman, A fair non-repudiation protocol, in: Proceedings 1996 IEEE Symposium on Security and Privacy, IEEE, 1996, pp. 55–61.
[39] H. Pagnia, F. Gärtner, On the Impossibility of Fair Exchange Without a Trusted Third Party. Darmstadt University of Technology, Technical Report TUD-BS-1999-02, Department of Computer Science, 1999.
[40] L. Xue, J. Ni, D. Liu, X. Lin, X. Shen, Blockchain-based fair and fine-grained data trading with privacy preservation, IEEE Trans. Comput. (2023).
[41] Y. Liu, X. Hao, W. Ren, R. Xiong, T. Zhu, K.-K.R. Choo, G. Min, A blockchain-based decentralized, fair and authenticated information sharing scheme in zero trust internet-of-things, IEEE Trans. Comput. 72 (2) (2022) 501–512.
[42] S.F. Shahandashti, R. Safavi-Naini, Threshold attribute-based signatures and their application to anonymous credential systems, in: Progress in Cryptology–AFRICACRYPT 2009: Second International Conference on Cryptology in Africa, Gammarth, Tunisia, June 21-25, 2009. Proceedings 2, Springer, 2009, pp. 198–216.
[43] J. Blömer, J. Bobolz, Delegatable attribute-based anonymous credentials from dynamically malleable signatures, in: International Conference on Applied Cryptography and Network Security, Springer, 2018, pp. 221–239.
[44] P. Li, J. Lai, Y. Wu, Publicly traceable attribute-based anonymous authentication and its application to voting, Secur. Commun. Netw. 2021 (2021) 1–17.
[45] B. An, M. Xiao, A. Liu, Y. Xu, X. Zhang, Q. Li, Secure crowdsensed data trading based on blockchain, IEEE Trans. Mob. Comput. (2021).
[46] A. Islam, M.F. Kader, M.M. Islam, S.Y. Shin, Newstradcoin: A blockchain based privacy preserving secure news trading network, in: IC-BCT 2019: Proceedings of the International Conference on Blockchain Technology, Springer, 2020, pp. 21–32.
[47] Y.J. Galteland, S. Wu, Blockchain-based privacy-preserving fair data trading protocol, Cryptology ePrint Archive (2021).
[48] Y.-N. Li, X. Feng, J. Xie, H. Feng, Z. Guan, Q. Wu, A decentralized and secure blockchain platform for open fair data trading, Concurr. Comput.: Pract. Exper. 32 (7) (2020) e5578.
[49] T. Li, W. Ren, Y. Xiang, X. Zheng, T. Zhu, K.-K.R. Choo, G. Srivastava, FAPS: A fair, autonomous and privacy-preserving scheme for big data exchange based on oblivious transfer, ether cheque and smart contracts, Inform. Sci. 544 (2021) 469–484.
[50] Q. Miao, H. Lin, J. Hu, X. Wang, An intelligent and privacy-enhanced data sharing strategy for blockchain-empowered internet of things, Digit. Commun. Netw. 8 (5) (2022) 636–643.
[51] M. Herlihy, Atomic cross-chain swaps, in: Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, 2018, pp. 245–254.