*electronics*

MDPI

*Article*

# Securing Big Data Exchange: An Integrated Blockchain Framework for Full-Lifecycle Data Trading with Trust and Dispute Resolution

**Chuangming Zhou [1], Zhou Yang [1], Shaohua Yue [1,*], Bona Xuan [1] and Xi Wang [2]**

[1] College of Air and Missile Defence, Air Force Engineering University, Xi'an 710051, China; zcmcldl@163.com (C.Z.); gyyz2024@163.com (Z.Y.); afeunbx219318@163.com (B.X.)
[2] Xi'an Satellite Control Center, Xi'an 710043, China; wxxawc@163.com
* Correspondence: sxxaysh@163.com

**Abstract:** In the era of big data, facilitating efficient data flow is of paramount importance. Governments and enterprises worldwide have been investing in the big data industry, promoting data sharing and trading. However, existing data trading platforms often suffer from issues like privacy breaches, single points of failure, data tampering, and non-transparent transactions due to their reliance on centralized servers. To address these challenges, blockchain-based big data transaction models have been proposed. However, these models often lack system integrity and fail to fully meet user requirements while ensuring adequate security. To overcome these limitations, this paper presents an Ethereum-based big data trading model that establishes a comprehensive and secure trading system. The model aims to provide users with more convenient, secure, and professional services. Through the utilization of smart contracts, users can efficiently match data and negotiate prices online while ensuring secure data delivery through encryption technologies. Additionally, the model introduces a trusted third-party entity that offers professional data evaluation services and actively safeguards user data ownership in the event of disputes. The implementation of the model includes the development of smart contracts and the necessary machine learning code, followed by rigorous testing and validation. The experimental results validate the effectiveness and reliability of our proposed model, demonstrating its potential to ensure effective and secure big data trading.

**Keywords:** big data trading model; blockchain; dispute resolution mechanism; data ownership protection; Ethereum; machine learning



**Citation:** Zhou, C.; Yang, Z.; Yue, S.; Xuan, B.; Wang, X. Securing Big Data Exchange: An Integrated Blockchain Framework for Full-Lifecycle Data Trading with Trust and Dispute Resolution. *Electronics* **2023**, *12*, 4894. https://doi.org/10.3390/electronics12244894

Academic Editors: Alberto Fernandez Hilario, Muhammad Khurram Khan, Wei Liang and Xiong Li

Received: 2 November 2023
Revised: 24 November 2023
Accepted: 1 December 2023
Published: 5 December 2023



## 1. Introduction

Big data has become incredibly important in our society because of the constant advancements in information technology. This has led to the widespread use of various information technology applications worldwide, resulting in billions of people generating massive amounts of data every day. At the same time, the rapid development of storage and data processing technology has greatly increased the efficiency of data utilization. As a result, scholars and industry professionals now consider data to be a crucial resource in the information age.

The application of big data has been accelerated by the development of new technologies such as cloud computing, the Internet of Things (IoT) [1], artificial intelligence, and blockchain. The widespread use of IoT not only enhances user experiences but also enables IoT application developers to provide better services through data analysis. Moreover, the recently popular global phenomenon-level application ChatGPT (Chat Generative Pre-trained Transformer) has sparked a wave of usage, research, and entrepreneurship in artificial intelligence. ChatGPT is a natural language processing tool that uses the transformer neural network model, which also requires massive amounts of data to train the model for better results. The use of big data gives companies and organizations a more

competitive advantage [2], and governments can use big data to provide faster and more efficient government services.

There is a lack of smooth data flow between data generation and usage, especially as enterprises grow and develop multiple business units. Each unit often stores its own data separately, creating physically isolated "data islands" that struggle to connect and interact with each other. The presence of such data islands poses risks to enterprises, including data duplication, incorrect decision-making, poor collaboration, inefficiency, and unsatisfactory customer experience. Data isolation between multiple enterprises can have even more damaging effects. In the era of big data, the inability to circulate and interact with data can hinder the development of entire industries and even society. As a result, building a bridge between data generation and usage and achieving fair and open data transactions has become a significant concern.

To address this concern, several platforms, such as Terbine, Datatrading, and GBDEX, have emerged to provide big data trading services as an initial exploration of the industry. However, the data sources of these platforms mainly come from agencies, and the amount of data provided by individual data providers is often insufficient to meet the needs of enterprises or organizations. Directly providing data to end users proves challenging. Therefore, designing and implementing a secure, fair, and accessible data trading platform for all users holds great research and practical significance in the era of big data.

While the industry has already applied, some data trading platforms and the research community have proposed various big data trading approaches. However, most current platforms still rely on centralized third parties to provide services. Unfortunately, this centralized approach is prone to single points of failure, lack of traceability, privacy breaches, and non-transparent transactions. Once someone gains higher privileges in a centralized server, they can freely manipulate the data stored on the server, including personal information, account balance, transaction records, etc., thereby infringing on user privacy. Moreover, opaque transaction records can potentially lead to tampering with users' account balances and transaction records, causing direct harm to users' interests. Although there are policies and legal regulations in place to address these issues, they still occur frequently. However, the decentralized network architecture of blockchain [3,4], along with its security and transaction transparency features, is ensured by the inherent structure and algorithms of blockchain itself, making it more secure and reliable than centralized servers. On blockchain platforms, various smart contracts can be written to constrain and regulate the behavior of all parties involved in big data transactions, thereby creating a more secure, fair, and universally reliable trading environment [5,6].

In this paper, we propose a model called the Big Data Trading Marketplace (BDTM) based on Ethereum smart contracts. In BDTM, data buyers publish demand messages, and data sellers respond accordingly. Through this marketplace, both parties can efficiently achieve their buying and selling goals. The data-matching process is completed using smart contracts. Users can apply to the Data Pricing and Evaluation Center (DPEC) to assess the pricing and quality of data provided by sellers. With the proposed pricing as a reference, both parties can negotiate the price using smart contracts. The data provided by the seller is stored in the InterPlanetary File System (IPFS). Only after the seller agrees to the data pricing and evaluation from the DPEC can the buyer obtain the encrypted file hash value provided by the seller and decrypt it to download the data from IPFS, ensuring the security of the seller's data. To maintain ownership of the user's data, the model establishes the Dispute Handling Organization (DHO). The DHO utilizes machine learning methods to compare the similarity of two pieces of data, preventing buyers from reselling data for profit. Additionally, data buyers can provide comments and ratings on the data they have purchased and on the data sellers, which serves as a reference for other buyers in the trading market. The above strategies encourage users to improve data quality, comply with trading rules, and protect data security and user privacy. This, in turn, attracts more data buyers and sellers to participate in big data trading.

The main problem studied in this paper is how to build a secure, whole-process, reliable, and practical big data trading system in the blockchain system. Therefore, in the Section 2, this paper introduces the current blockchain-based big data trading systems and analyzes their existing problems. The Sections 3 and 4 introduce the technical background needed for this system and design the model architecture and security requirements. In Sections 5 and 6, the implementation details of the system and experimental simulation are introduced in detail to verify the reliability and effectiveness of the designed system.

## 2. Related Work

Due to advances in blockchain technology and the advent of the big data era, trading big data resources on blockchain platforms has become a popular direction to explore. Zyskind et al. [7] proposed a scheme that uses blockchain to protect personal privacy in a decentralized data environment. Chen et al. [8] proposed an incentive mechanism based on a federated blockchain with quality-driven auctions for data sharing on the Internet of Vehicles (IoV). The mechanism ensures trust in on-chain and off-chain data and maximizes social welfare. Jung et al. [9,10] proposed a set of accountability protocols for big data trading. However, they assume that brokers are trustworthy, which leads to privacy breaches and risks in practical application scenarios. Chen et al. [11] proposed a blockchain-based big data people management system to protect big sensitive data while providing information management operations such as querying, adding, and modifying. However, the system does not offer big data transactions between multiple parties and does not analyze the system's security. Hu et al. [12] proposed a big data trading model that separates transactions from data with the help of a sidechain mechanism, protects data security, and provides methods to improve the usability of the data trading model, such as bargaining and quality assessment. Wei et al. [13] proposed a data resource protection scheme by simulating honest users to report the dumping behavior of malicious users. With the help of machine learning methods, a trusted resolution authority determines whether the user's behavior constitutes malicious data dumping. Park et al. [14] proposed using smart contracts to publish data comments in the IoT data trading market.

Guan et al. [15] proposed two secure data trading schemes using blockchain that do not rely on third parties. Their first scheme performs direct raw data exchange of large amounts of data, while the second scheme performs statistical data trading. They implement both schemes with smart contracts. Gao et al. [16] designed differentiated private crowd-aware data trading mechanisms that protect consumers' identity privacy and crowd workers' task privacy during data collection. Dai et al. [17] proposed a blockchain and SGX-based data trading ecosystem. In the ecosystem, neither data brokers nor buyers can access the seller's raw data, as they only obtain access to the analysis findings that they require. Li et al. [18] proposed a blockchain-based fair and responsible trading scheme for educational multimedia data that combines BIBD-ACC's anti-obfuscation code and asymmetric fingerprinting technology to achieve relatively strong copyright protection while implementing a smart contract with a reasonable pricing model to facilitate fair trading. Zhao et al. [19] proposed a new blockchain-based fair data trading protocol to verify data availability for data users, protect the privacy of data providers, and achieve fairness in payments between data providers and data consumers. Guan et al. [20] proposed a framework called Trusted Big Data Collection and Transaction System (TBDCT) to provide a fair and trustworthy platform for participants in the Big Data world. Their framework utilizes a physical unclonable feature technology to tie data to a unique sensor "fingerprint", ensuring the authenticity of the data source. Additionally, private attached network storage with embedded trusted security modules is employed to guarantee the trustworthiness of the data collection process. Michael et al. [21] proposed the application of blockchain technology to implement a data marketplace for the IoT. Within the proposed marketplace, smart contracts are used to implement various functionalities and enforce the rules of the data exchange. The marketplace also includes a proxy, a broker, and (GUIs) to enable data trading. An et al. [22] proposed that BCDT uses blockchain and a smart contract

(BCDToken) to ensure trust in data trading. BCDToken employs a Blockchain-based Reverse Auction (BRA) for task assignment and a Secure Truth Discovery and Reliability Rating (STDR) mechanism based on homomorphic cryptography deployed on the Ethereum test network for practicality and performance demonstration. Interventionary studies involving animals or humans, and other studies that require ethical approval, must list the authority that provided approval and the corresponding ethical approval code. However, the abovementioned works all have problems, such as a lack of system security and a lack of overall integrity. In order to make up for the above deficiencies, this paper designs a big data transaction model with the security of the whole process. The innovative contributions of this paper are summarized as follows:

(1) Compared to existing big data trading models based on blockchain platforms, our proposed model offers a more comprehensive and systematic big data trading process.

(2) By introducing trusted third-party entities managed by professionals, the model provides more professional services in data negotiation and dispute resolution.

(3) On the Ethereum platform, we have successfully built a complete big data trading system. The key functionalities, including data matching, evaluation, pricing, and negotiation, have been implemented. Additionally, the ownership of user data is protected off-chain using machine learning techniques.

## 3. Technical Background

In this chapter, we mainly introduce and discuss the key platforms and technologies used in this paper.

### 3.1. Ethereum

Blockchain technology [23] is a decentralized and traceable distributed ledger technology. It links data together in the form of blocks to create a complete and tamper-proof record of transactions, ensuring the reliability and security of information. The core features of blockchain include decentralization, immutability, transparency, and security. Ethereum is a cryptocurrency platform built on blockchain technology. It is considered a milestone of WEB 2.0 and was designed to address the scalability issue of Bitcoin [24]. At the core of the Ethereum platform are smart contracts, which enable the deployment of applications with various functions. With a current market capitalization second only to Bitcoin, Ethereum has amassed a large user base, with over 1 million active users daily. In September 2022, Ethereum upgraded its consensus algorithm from Proof-of-Work (POW) [3] to Proof-of-Stake (POS) [25]. This transition to a more environmentally friendly algorithm eliminates the excessive consumption of electricity and energy associated with POW.

### 3.2. Smart Contract

The concept of a Smart Contract (SC) was first introduced by cryptographer Nick Szabo in 1994 [26]. Smart contracts are essentially shareable distributed databases deployed in event-driven, stateful computer programs. They operate similar to if-then statements in other computer programs, enforcing contract terms when specific conditions are triggered. To function properly, smart contracts must be deployed in the blockchain. The blockchain is a tamper-proof, decentralized, open, and transparent platform, providing an ideal environment for executing smart contracts. In Ethereum, the contract layer incorporates the Ethereum Virtual Machine (EVM) and a Turing-complete language execution environment. Solidity, a programming language, is used to write smart contracts on Ethereum. This Turing-complete language provides complete freedom, allowing for the implementation of a wide range of contract functionalities. Currently, the number of smart contracts deployed on Ethereum has exceeded 50 million.

### 3.3. InterPlanetary File System

IPFS [27] is a peer-to-peer network transfer protocol that aims to address the limitations of the HTTP protocol, such as slow speed and low efficiency caused by time, access,

and operation speed constraints. Unlike HTTP, which relies on centralized servers, IPFS enables the creation of shared files with persistent distributed storage. It is a hypermedia distribution protocol based on content addressing. In an IPFS network, nodes collaborate to form a distributed file system.

### *3.4. Machine Learning*

Machine learning is a branch of artificial intelligence that aims to study and develop algorithms and models capable of learning and improving from data. It utilizes various techniques such as statistics, mathematics, and computer science to train computer models, enabling them to perform tasks like prediction, classification, clustering, and more without the need for explicit programming instructions. The core idea behind machine learning is to enable computers to automatically learn, extract patterns and rules from large amounts of data, and make autonomous learning and intelligent decisions. Machine learning finds wide applications in various industries, including natural language processing, image recognition, financial risk analysis, and more. Through machine learning, we can create intelligent systems and services that enhance efficiency and accuracy.

## 4. Model Architecture

In this chapter, the system architecture and security requirements of BDTM are mainly designed.

### *4.1. System Model*

The system model architecture designed in this paper is shown in Figure 1.

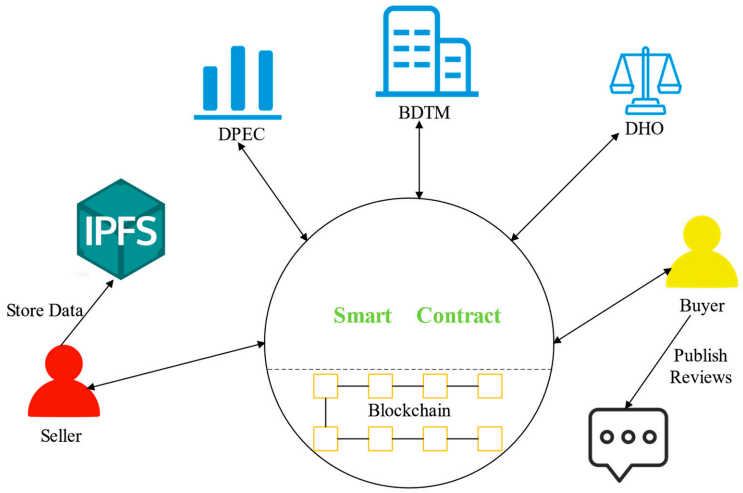



**Figure 1.** The architecture of BDTM.

Big Data Trading Marketplace (BDTM): The BDTM is a trusted entity that deploys and owns the SC (Smart Contract). It broadcasts data for sale and demand messages to the nodes in the blockchain, facilitating data trading between sellers and buyers. The BDTM establishes marketplace rules that balance the interests of all parties, and these rules are stored on IPFS (InterPlanetary File System), accessible to all users.

Data sellers: Data sellers can be private users or organizations, responding to demand from data buyers by selling data. They can actively post data for sale through the data exchange. A data seller's profit is determined by the quantity and quality of data provided, as well as the negotiated price with the buyer.

Data buyers: Data buyers are usually individuals and organizations seeking large-scale data analysis for scientific research, risk assessment, and business research purposes. They post their demand messages in the blockchain through the BDTM.

Data Pricing and Evaluation Center: Operated by the BDTM, it is reliable and fair. Employ professional data analysts, establish data evaluation and pricing models to evaluate and price data submitted by users, and provide professional real-time services during the running time of the data trading market.

Dispute Handling Organization: Run by the BDTM, the DHO is also a reliable and fair entity. DHO is to supervise and punish users who maliciously resell data in the process of data transaction, and it also employs professional service personnel to judge whether the reported user really has the behavior of reselling data according to the evidence provided by the whistleblower and punish them if it is true. DHO will use the cosine similarity algorithm and LMNN algorithm to assist in judging the similarity of two data set files.

### *4.2. Security Model*

This subsection discusses the fundamental security requirements that BDTM must fulfill.

Anonymity: Anonymity is crucial in preventing the correlation of a data seller's identity with the data they provide, as this could lead to a breach of privacy.

Data Confidentiality and Integrity: To ensure data confidentiality and integrity, data sellers' information should be encrypted during both storage and transactions. The original data should only be accessible to the data seller, authorized data buyers, and authorized DPEC.

Transaction fairness: It is essential to guard against data sellers pricing their data unreasonably to maximize profits. The system should ensure that data pricing is based solely on its quality and relevance.

Security against malicious data sellers: Malicious data sellers may attempt to provide irrelevant and unsuitable data or aim to receive payment without contributing any valuable data. Measures should be in place to prevent and detect such behavior.

Security against malicious data buyers: Malicious data buyers may attempt to evade payment after obtaining the desired data. The system must guarantee that data buyers fulfill their payment obligations once they receive the data they require.

Data ownership protection: Preventing malicious data buyers from reselling data for profit is crucial. These buyers may misuse compliant sellers' data by altering data summaries or content, thereby damaging both the interests of compliant sellers and the overall trading market order. The system should be capable of detecting data ownership violations and punishing malicious users.

## 5. Implementation Details of Big Data Trading Marketplace

This chapter describes in detail the various functional definitions of BDTM and the implementation details of each functional module, including the definition, pseudo-code, algorithm process, schematic diagram, etc.

### *5.1. User Registration and Data Storage*

Users can register for an Ethereum account through multiple channels, which consist of a Private Key (PrK), a Public Key (PuK), and an Address of Ethereum (AOE). The obtained account identities are anonymous. It is crucial for individuals to strictly keep their PrK confidential, as any loss or theft can result in irreparable damage. The PuK and the AOE can be publicly disclosed.

It is important to note that storing data on the blockchain can be costly. To overcome this, data sellers utilize IPFS for data trading purposes. Users only need to deploy IPFS services locally, which becomes an IPFS network node. This allows them to store and download data for free. When users store data, they will receive a unique string of hash value, denoted as "$h$" returned by IPFS. Later on, users can use this hash value to access and download the stored data on IPFS.

### *5.2. Posting Data Request Message and Sale Message*

Both buyers and sellers can post data messages to buy and sell items as needed.

Data sellers submit their demand messages to the BDTM, where the message is saved in the chain and published in the marketplace to match with potential data buyers. Data buyers post their DemandMsg based on their specific requirements.

**Definition 1:** *DemandMsg is the buyer's demand message, which is defined in the following formula:*

$$DemandMsg = (Dusage, Dsize, dl)$$

where *Dusage* is the industry in which the data can be used, *Dsize* is the size of the data, and *dl* is the deadline for data demand. The *DemandMsg* defined here is the most basic message provided by the buyer for trading, and the buyer can provide a more detailed message about the requirement so that the seller can offer it on demand.

Algorithm 1 *now* represents the current time, *msg.sender* is the Ethereum address of the SC caller, *Status* is the current contract status, *BuyerList* is a list of buyers, *BuyerID* is the number assigned to the buyer by SC, *DemandList* is a list of *DemandMsg*, and *DemandMsgID* is the number assigned to the *DemandMsg* by SC.

**Algorithm 1:** Buyer demand message release

```
Input: DemandMsg
Output: list of buyer BuyerList, list of DemandMsg DemandList
1    Status ← true
2    if now > DemandMsg.dl then
3       Status ← false
4       return erorrs
5       AOE ← msg.sender
6    end if
7    while Status = true do
8       BuyerList = BuyerList + this.BuyerID
9       DemandList = DemandList + this.DemandMsgID
10  end while
11  return BuyerList, DemandList
```

Data sellers submit their data information to the BDTM, which stores it in Ethereum and publishes it on the trading marketplace to match data buyers. As the owners of the data, sellers should provide comprehensive product details to assist buyers in finding and filtering data more efficiently.

**Definition 2:** *Ddetail is the detailed message provided by the seller about the product data, which is defined in the following formula:*

$$DdetailMsg = (Ddescription, Dtype, Dtuple, Dsize, Raddres, dl)$$

where *Ddescription* is the data description, *Dtype* is the data type, *Dtuple* is the data tuple, *Dsize* is the data size, *Raddress* is the address of the account used to receive the data reward, and *dl* is the sale deadline.

In Algorithm 2, *Reward_Address* stores the Ethereum addresses that receive data rewards, *SellerList* is a list of record sellers, *SellerID* is the number assigned to the seller by SC, *DetailList* is a list of record *DetailMsg*, and *DetailMsgID* is the number assigned to the *DetailMsg* by SC.

**Algorithm 2:** Sellers list data items

**Input:** DdetailMsg
**Output:** list of seller SellerList, list of DdetailMsg DetailList

```
Status ← true
if now > DdetailMsg.dl then
    Status ← false
    return errors
    AOE ← msg.sender
    Reward_Address ← Raddress
end if
while Status = true do
    SellerList = SellerList + this. SellerID
        DetailList = DetailList + this.DetailMsgID
end while
return DetailList, SellerList.
```

### *5.3. Price Negotiation*

The most crucial step after the initial agreement on a transaction is to negotiate the price of the data transaction. The model allows buyers and sellers to negotiate prices through a bargain contract. Table 1 highlights the key functions of these contracts.

**Table 1.** Key functions of the bargain contract.

| Name of Function | Description of Function |
|---|---|
| Bargain() | Both parties use this function to bid in turn according to the bargaining stage. |
| DPEC() | Give reference rating and price. |
| Encrypting() | Encrypting *h*. |
| Calculate_price() | Predicted closing price based on bids from both sides. |

The bargaining process is a game between two parties. In this game, we make the following basic assumptions:

(1) Both buyers and sellers follow the basic rules of the transaction, and each bid falls within a reasonable range.

(2) The bargaining process is dynamic. If one party rejects the other party's offer, the next round of bargaining will take place, ensuring flexibility and fairness.

#### 5.3.1. Data Pricing and Evaluation Center

The price negotiation process, based on the above flexible and fair assumptions, may pose challenges as not every user has a reasonable prediction of the data price. Information asymmetry can jeopardize the interests of one party, resulting in a lack of consensus during multiple rounds of bargaining and inefficient transactions. To solve this problem, this paper proposes a DPEC that involves a trusted third party to evaluate the transaction data and provide a reference price. DPEC employs professionals to establish a comprehensive data evaluation mechanism by incorporating advanced data evaluation models and pricing standards from existing data trading platforms. In general, the data evaluation mechanism of DPEC will assess various aspects of the data itself, including data quality, size, attributes, type, timeliness, and other elements, to determine a reference evaluation level and then combine market factors, industry characteristics, the system's historical transaction price of similar data to provide a reference price. This evaluation mechanism is transparent, open to all users, and subject to user supervision. The corresponding specification file is stored on IPFS, accessible to all users.

Users can apply to DPEC at any stage of the bargaining process. DPEC requests *h* from the seller, which, if agreed upon, are encrypted and sent to DPEC. After downloading the data samples from IPFS, DPEC evaluates and prices them based on the provided samples.

Both parties can then refer to the estimated price and continue their negotiation. Typically, the requester is responsible for paying a fee to DPEC for its services.

5.3.2. Bargaining Model

This paper presents a game process of pricing based on the Rubinstein bargaining model [28]. The application of the Rubinstein bargaining model relies on an understanding of the following three key points:

(1) Sequential Game: In each round, only one party makes a bid while the other party responds with {accept, reject}. For instance, suppose the seller initiates the bidding in the first round. If the buyer accepts the bid, the game ends. If the buyer rejects the bid, the game enters the second round, where the buyer proposes a counteroffer, and the seller decides whether to accept or reject it. This process continues until one participant accepts the other's bid. Specifically, the seller bids in rounds 1, 3, 5,..., while the buyer bids in rounds 2, 4, 6,...

(2) Set the discount factor $\delta$: Unlike the discount rate in economics, $\delta$ reflects the bargaining participants' level of patience.

(3) Principle of early acceptance: During the bargaining game, both sides incur negotiation costs, including the cost of time and opportunity costs. As the number of negotiations increases, these costs mount, and the benefits decrease. Thus, it is advisable for both sides to accept reasonable offers as early as possible to minimize these costs.

Model parameters are set: $\delta_s(0 \leq \delta_s \leq 1)$ is the seller's discount factor; $\delta_b(0 \leq \delta_b \leq 1)$ is the buyer's discount factor; $P_t^s(t \in [1,3,5,\cdots])$ is the seller's bid in round t, $P_t^b(t \in [2,4,6,\cdots])$ is the buyer's bid in round $t$; $U_t^s(t \in [1,3,5,\cdots])$ is the seller's subgame refined Nash equilibrium payoff in round $t$, $U_t^b(t \in [2,4,6,\cdots])$ is the buyer's subgame refined Nash equilibrium payoff in round $t$; and E is the Nash equilibrium price that both parties agree on.

The study of sequential games typically employs reverse induction, whereby the actions of the second bidder simulate the behavior of the first bidder, and the entire game process is simulated inductively. The seller's first bid ($P_1^s$) is denoted as V. The inverse induction method first examines $P_2^b$, which is the buyer's bid in round 2. For the buyer to maximize the bid and for the seller to accept this price in round 2, then $P_2^b$ should satisfy the following inequality:

$$V - P_2^b \geq \delta_s U_3^s \tag{1}$$

For $P_2^b$ to take the maximum, it should satisfy the following:

$$\text{argmax}\left\{ P_2^b \middle| V - P_2^b \geq \delta_s U_3^s \right\} = V - \delta_s U_3^s \tag{2}$$

This can be obtained from $U_2^b$ as

$$U_2^b = V - \delta_s U_3^s \tag{3}$$

The same can be obtained from $U_1^s$ as

$$U_1^s = V - \delta_b U_2^b \tag{4}$$

Because the process of a sequential game is repetitive, the participants receive the same in each round of the game, $U_1^s = U_3^s$. In the case of a user applying to DEPC, it is assumed that both parties have symmetric information and are patient enough to engage in bargaining. Therefore, the discount factor is set to $\delta = \delta_s = \delta_b$, which is calculated according to the Rubinstein bargaining game model as

$$(U_1^s, U_1^b) = \left(\frac{V}{1+\delta}, \frac{\delta V}{1+\delta}\right) \tag{5}$$

$$E = \frac{(1-\delta)(P_t^s - P_{t\pm1}^b)}{1-\delta^2} + P_{t\pm1}^b \tag{6}$$

In the absence of an application to the DPEC, there is an asymmetry of information between the parties involved, with the seller typically having more patience than is generally assumed ($\delta_s > \delta_b$). This is calculated according to the Rubinstein bargaining game model, which gives

$$(U_1^s, U_1^b) = (\frac{(1-\delta_b)V}{1-\delta_s\delta_b}, \frac{(1-\delta_s)\delta_b V}{1-\delta_s\delta_b}) \tag{7}$$

$$E = \frac{(1-\delta_b)(P_t^s - P_{t\pm1}^b)}{1-\delta_s\delta_b} + P_{t\pm1}^b \tag{8}$$

In Algorithm 3, t represents the tth round in the bargaining process. The call to abandon() indicates that a party has decided to give up the bargaining, thereby signaling the end of the price negotiation. The call to DPEC() represents a party's decision to apply to DPEC for evaluation and pricing of the current data commodity, where *r* stands for the reference evaluation level given by DEPC, and *V* stands for the reference price given by DEPC. The choice variable is used to indicate whether the user chooses to accept or reject.

**Algorithm 3:** Bargaining

**Input:** $P_t^s$, $P_t^b$, $\delta_s$, $\delta_b$
**Output:** success or failure

```
Bargain_flag = 0
//Bargaining success flag
 while Bargain_flag = 0 do
     if Seller.AOE call abandon() or Buyer.AOE call abandon() then
//The user abandons the transaction
         return failure
     end if
     if Seller.AOE call DPEC() or Buyer.AOE call DPEC() then
         return r,V by DPEC
          P_t^s = V
//Users can apply to DPEC for pricing of data at any stage in the bargaining process.
      if (t% 2 = 1) and msg.sender = Seller.AOE then
     //The Seller bidding stage.
          update P_t^s
          compute (U_1^s, U_1^b), E
          if Buyer.choice = accept then
            Bargaining success.
            Bargain_flag = 1
          else if Buyer.choice = reject then
            t = t + 1
          end if
      eles if (t% 2 = 0) and msg.sender = Buyer.AOE then
       //The Buyer bidding stage.
          update P_t^b
          compute (U_1^s, U_1^b), E
          if Seller.choice = accept then
            Bargaining success.
            Bargain_flag = 1
          else if Seller.choice = reject then
            t = t + 1
          end if
      end if
 end while
 return success.
```

The theoretical model calculates the results as $(U_1^s, U_1^b)$ and E. However, in actual negotiations, the information between the two parties involved may not be completely symmetrical and $\delta$ may constantly change. Therefore, the model provides reference results. The system will improve the bargaining model by referring to the deviation of the actual price of each transaction from the theoretical price calculated by the model to provide a more reasonable reference price.

### *5.4. Data Delivery*

After both parties have negotiated the price, the seller will deliver to the buyer a hash value string, $h$, for downloading data on IPFS. Anyone who gets $h$ can download this data on IPFS, so both parties need to confirm each other's identity before sending $h$. The data delivery process is as follows:

(1) The seller combines the Time Stamp of Seller (TSS), the negotiated price E, and the product data name (DN) into a message, $SellerMsg = TSS\&E\&DN$. The seller then signs the message with his private key and sends it to the buyer. The formula is

$$\text{SellerSinged}(SelerMsg) = \text{Seller.Prk}(\text{Hash}(SelerMsg))$$

(2) The buyer receives the message from the seller and verifies its authenticity. Upon successful verification, the buyer replies to the seller with their own message. The content of the message includes the Time Stamp of Buyer (TSB), E, and DN, that is $BuyerMsg = TSB\&E\&DN$, and it is sent to the seller after signing with their private key. The formula is

$$\text{BuyerSinged}(BuyerMsg) = \text{Buyer.Prk}(\text{Hash}(BuyerMsg))$$

(3) After receiving the buyer's message and confirming the identity of both parties, the seller encrypts $h$ with the buyer's public key and sends it back to the buyer. The formula for the encrypted hash value is

$$EncryetedMsg = \text{SellerEncrypt}(h) = \text{Buyer.PuK}(\text{Hash}(h))$$

Buyer receives EncryptedMsg and decrypts it with his private key to obtain $h$. The formula for the encrypted hash value is

$$\text{BuyerDecrypt}(EncryptedMsg) = \text{Buyer.Prk}(EncryetedMsg)$$

(4) The buyer uses $h$ to go to IPFS to download data.

The process of data delivery is shown in Figure 2.

### *5.5. Data Ownership Dispute Handling*

Data ownership protects the value of data. Ownership of data by the user ensures that its value is captured. Unclear ownership can lead to misuse, theft, or destruction of data, greatly diminishing its value [29]. Therefore, it is essential to establish and safeguard data ownership in the development of BDTM to protect the interests of users. Our model achieves this by implementing a DHO, which operates as follows:

(1) Reporting: Users who suspect data dumping by sellers can utilize the Whistleblower's Address of Ethereum (W.AOE) to trigger a smart contract that sends a *ReportMsg* to the DHO. The *ReportMsg* is defined by the following formula:

$$ReportMsg = (\text{S.AOE}, DataA, DataB)$$

where S.AOE is the Suspect's Address of Ethereum (S.AOE), *DataA* is the data in his possession, and *DataB* is the suspected data. To prevent malicious reporting, the whistleblower is required to provide a deposit to DHO.

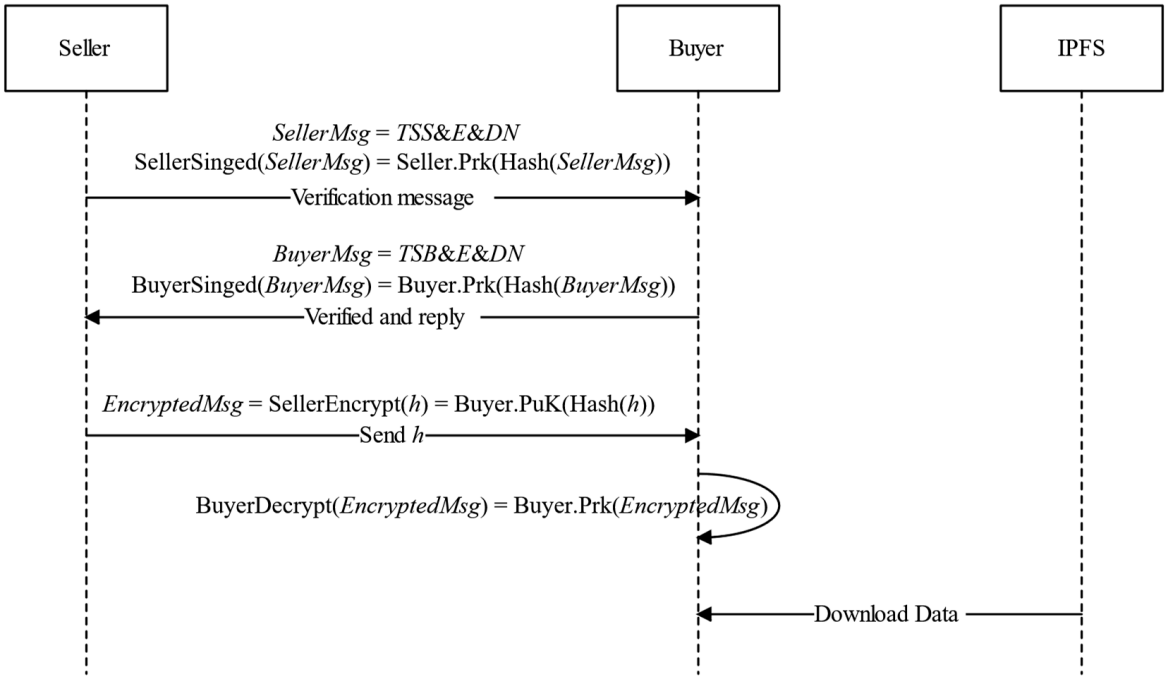



**Figure 2.** The process of data delivery.

(2) Acceptance: The DHO receives user reports and sends requests to download *h* and *h*' of *DataA* and *DataB* to the informant and suspect, respectively, as specified in the report message. Once the DHO obtains *h* and *h*', it downloads the data from IPFS and compares the similarity between *DataA* and *DataB* using the machine learning method.

(3) Processing result: Based on the similarity obtained in step 2, the DHO determines whether the reported individual has indeed dumped the data.

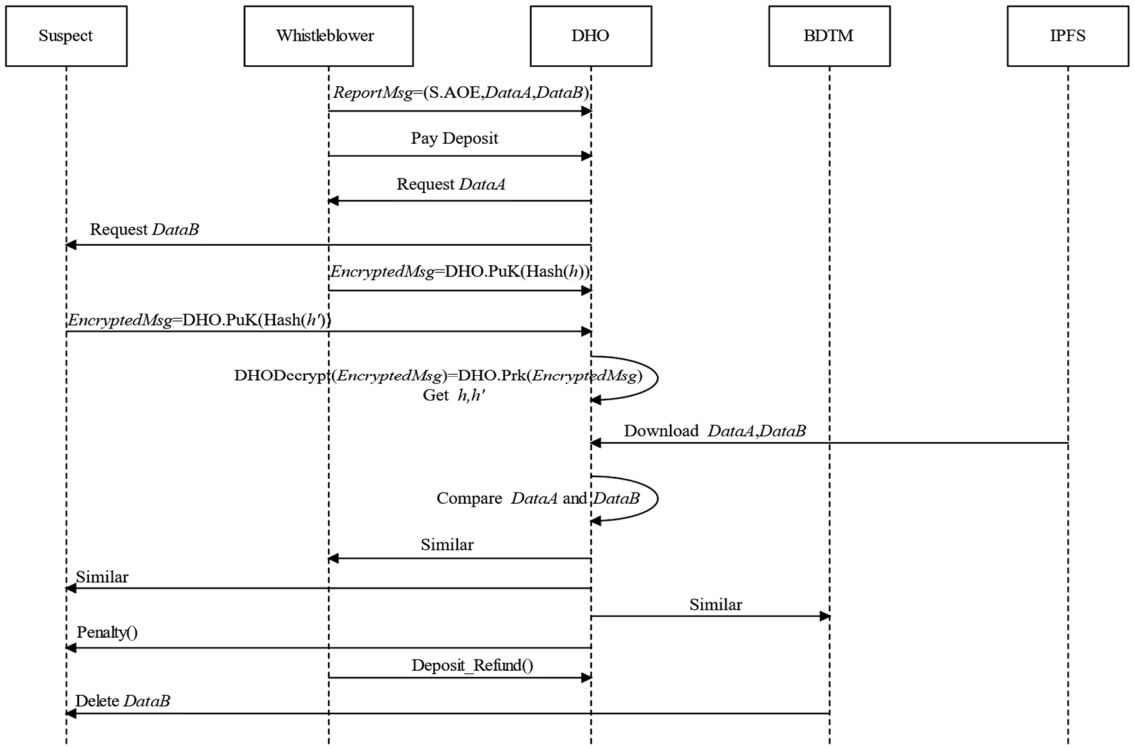
If the suspect is found innocent, the deposit paid by the whistleblower is deducted and distributed proportionally among the BDTM, the suspect, and the DHO.

If the suspect is confirmed to have sold the data, they are required to pay a penalty, which is then distributed proportionally among the BDTM, the whistleblower, and the DHO. Simultaneously, the BDTM removes the sale of the data posted by the suspect through a smart contract and returns the whistleblower's deposit.

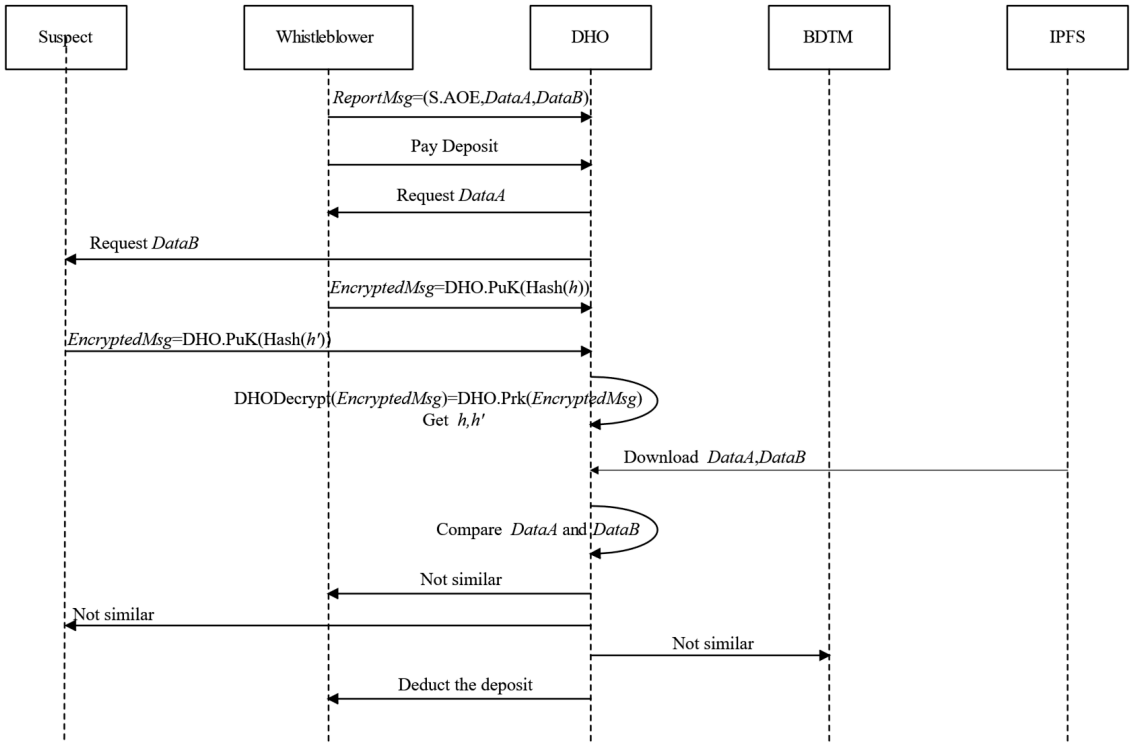
The dispute-handling process is shown in Figure 3.

### *5.6. Comment*

Like other online trading platforms, data trading marketplaces also face the issue of user trust in products and sellers. The purpose of implementing a commenting contract is to enable buyers to understand the reputation of the user and the quality of the goods through comments posted by other users. This enhances trust between unfamiliar parties, reduces concerns, and increases the desire to purchase products. The commenting system on blockchain platforms offers more advantages compared to server-centered platforms. In server-centered platforms, there is a possibility of administrators using their privileges to manipulate user comments or engage in fraudulent transactions using false identities. However, in blockchain platforms, users' identities are peer-to-peer and anonymous, and the data stored on the blockchain cannot be tampered with. Therefore, users can confidently post and view product comments. The functions of the comment contract are shown in Table 2.




(**a**)




(**b**)

**Figure 3.** (**a**) Dispute processing process (similar). (**b**) Dispute processing process (Not similar).

**Table 2.** The main functions used to implement the comment contract.

| Name of Function | Description of Function |
| --- | --- |
| SetComment() | Post a comment. |
| GetComment() | View comments. |
| GetRating() | View ratings. |
| Reward() | Sends rewards to users who post comments according to the rules. |

Buyers can share their feedback and ratings for a specific data item by using the SetComment() function. This function verifies that the user's address matches the address of the buyer who purchased the data, ensuring that only those who have completed the transaction can leave comments. SetComment() allows users to access both positive and negative comments, ensuring honest feedback.

For users who prefer to view only the ratings instead of all the comments, the SetRating() function is available.

To incentivize buyers to post comments and enhance interaction between the marketplace and sellers, the Reward() function can be utilized. Posting comments on the Ethereum platform incurs gas costs, making some buyers hesitant to provide feedback. By using Reward(), sellers can send rewards to buyers who have posted comments, encouraging more users to share their experiences and increasing the visibility and sales of their products.

## 6. Security Analysis and Experimental Testing

This chapter first analyzes the security of BDTM to correspond to the previous security requirements, then introduces cosine similarity and LMNN to calculate the similarity between the two data sets, and finally realizes the two algorithms and the whole BDTM system and verifies their effectiveness.

### *6.1. Security Analysis*

#### 6.1.1. Anonymity

Ethereum account registration utilizes asymmetric cryptography to generate random public–private key pairs, eliminating the need for users to provide any personal real identity information. Moreover, it is also almost impossible for other users to correspond the user's Ethereum address to the user's real identity in real life while using the user's account.

#### 6.1.2. Data Confidentiality and Integrity

IPFS incorporates both decentralized and content-based addressing features. Content-based addressing ensures that even minor alterations to the stored data in IPFS would result in a change of the addressing location, represented by a different hash value ($h$). This characteristic guarantees the data's integrity, as it remains unaltered by any unauthorized source.

When the seller stores the data for sale in IPFS, it remains inaccessible for download without obtaining the corresponding $h$. The seller encrypts $h$ when delivering the data, thus ensuring that only authorized users, including the seller, can access the data.

#### 6.1.3. Transaction Fairness

The system facilitates transparency and fairness by allowing users to review previous comments and ratings of product data. Additionally, users can evaluate purchased data, engage in negotiations with sellers, and apply to the DPEC to evaluate product data and receive reference prices. These measures help maintain symmetric information exchange between the parties involved, ensuring transactional fairness.

#### 6.1.4. Responding to the Security of Malicious Data Sellers

In cases where a malicious data seller delivers data that does not match the buyer's expectations, the buyer has the option to report both the data item and the seller to the

DHO. The DHO will then verify the authenticity of the reported fraud and impose penalties on the malicious seller if the fraud is validated. Conversely, if the report is false, the buyer will be penalized for misreporting.

#### 6.1.5. Responding to the Security of Malicious Data Buyers

To prevent malicious data buyers from acquiring data without payment, the system requires buyers to deposit funds into a smart contract before initiating price negotiations. The deposit amount is then adjusted accordingly based on the negotiated price. If no agreement is reached, the buyer can request a refund of their deposit. The use of blockchain technology ensures traceability of the funds' source and destination, enhancing security measures.

#### 6.1.6. Data Ownership Protection

DHO uses two machine learning algorithms to help measure the similarity between two dataset files in order to accommodate different dataset file metrics tasks. The cosine similarity algorithm is used to measure the similarity of simple text datasets; the Large Margin Nearest Neighbor metric learning algorithm is used to measure more complex, logically related datasets.

Cosine Similarity

Cosine similarity is a measure of similarity between two text files by calculating the angle between two text vectors. Its algorithm flow is as follows:

Two pieces of text, A and B, are segmented to obtain two-word lists:

$$A = [t_1, t_2, \cdots, t_i]$$

$$B = [t_1, t_2, \cdots, t_j]$$

The two-word lists are merged and de-weighted to obtain all the words in the input sample:

$$T(A, B) = T(A) + T(B) = [t_1, t_2, \cdots, t_k]$$

Selecting word frequencies as feature values:

$$F(A) = [f_{A_1}, f_{A_2}, \cdots, f_{A_k}],$$

$$F(B) = [f_{B_1}, f_{B_1}, \cdots, f_{B_k}]$$

Vectorization of eigenvalues:

$$\vec{A} = (f_{A_1}, f_{A_2}, \cdots, f_{A_k})$$

$$\vec{B} = (f_{B_1}, f_{B_2}, \cdots, f_{B_k})$$

Calculating cosines similarity:

$$\cos\theta = \frac{\sum_{i=1}^{k} f_{A_i} \cdot f_{B_i}}{\sqrt{\sum_{i=1}^{k} (f_{A_i})^2} \cdot \sqrt{\sum_{i=1}^{k} (f_{B_i})^2}} = \frac{f_A \cdot f_B}{||f_A|| \cdot ||f_B||} \in [-1, 1] \quad (9)$$

The value of $\cos\theta$ close to 1 means that $\theta$ is close to 0, and the angle between the two text vectors is close to 0, indicating that the two text files are more similar. Conversely, the value of $\cos\theta$ away from 1 indicates that the two text files are less similar.

Large Margin Nearest Neighbor Metric Learning

For some complex, logically related, and labeled datasets, we usually use data classification algorithms in machine learning, such as KNN. However, since the Euclidean distance generally used by the K-NearestNeighbor (KNN) algorithm does not always characterize the statistical features of the labeled samples well, Weinberger et al. further proposed Large Margin Nearest Neighbor metric learning (LMNN) [30] to be used to pre-train a distance metric. The key to LMNN is first to obtain the matrix L from the training data. For a given data point $x_i, x_j \in R^d$, compute the new distance metric:

$$D(x_i, x_j) = \| L(x_i - x_j) \|^2 \tag{10}$$

Let the semi-positive definite matrix $M = L^T L$, then D can be rewritten as the Mahanalobis distance between two points:

$$D(x_i, x_j) = (x_i - x_j)^T M(x_i - x_j) \tag{11}$$

After obtaining the new distance metric, the sample points are divided into ideal cases using a loss function, which intuitively feels like a part of the loss function responsible for clustering sample points of the same category, and the function is constructed:

$$\varepsilon_{\text{pull}}(L) = \Sigma_{i,j \to i} \| L(x_i - x_j) \|^2 \tag{12}$$

The other part of the loss function is responsible for pushing the sample points of different classes farther away, and the function is constructed as follows:

$$\varepsilon_{\text{push}}(L) = \Sigma_{i,j \to i} \Sigma_l (1 - y_{il}) [1 + \| L(x_i - x_j) \|^2 - \| L(x_i - x_l) \|^2]_+ \tag{13}$$

Here, $j \to i$ denotes the traversal of the k nearest points to the sample point i under the Euclidean distance calculation, and $y_{il}$ is the indicator variable, which is 1 when the indicators $y_i = y_l$ and 0 otherwise, where $y_i$ and $y_l$ are the indicator variables of $x_i$ and $x_l$ respectively.$[z]_+ = \max\{z, 0\}$.

The final objective function is expressed as a weighted sum of the two terms, balanced and adjusted using the parameter $\mu$.

$$\varepsilon(L) = (1 - \mu)\varepsilon_{\text{pull}}(L) + \mu\varepsilon_{\text{push}}(L), \ \mu \in [0, 1] \tag{14}$$

LMNN dramatically improves the classification accuracy and the credibility of similarity comparison by the above strategies.

### *6.2. Experiment Testing*

#### 6.2.1. Validation of Data Ownership Protection Algorithm

In this section, we implemented the cosine similarity algorithm and LMNN using Python and validated the performance of each algorithm with different datasets. We have our code, and data sets will be uploaded to https://github.com/yangzhou-code/LMNN (accessed on 7 October 2023).

Validation of Cosine Similarity Algorithm

To validate the cosine similarity algorithm, we utilized Python's CountVectorizer class to convert the text data into a numerical vector representation. We generated a word frequency matrix by assigning each word in the text as a feature and calculating its frequency of occurrence.

Initially, we tested the similarity between two identical samples from different document types. Next, we randomly deleted a portion of characters in one sample and conducted the similarity test again. The experimental results are presented below.

According to the findings in Table 3, it is evident that the cosine similarity algorithm achieves 100% similarity when comparing identical files. Furthermore, when randomly removing certain characters from a sample, the obtained similarity decreases, which indicates that the cosine similarity algorithm we use can be used for the similarity detection of text files.

**Table 3.** Text file similarity experiment.

| Document Type | Number of Characters in the Doument | Similarity |
|---|---|---|
| docx | same | 100% |
| txt | same | 100% |
| docx | 14,379 and 20,486 | 85.62% |
| txt | 27,602 and 46,439 | 84.45% |
| docx | 8569 and 20,486 | 66.69% |
| txt | 12,841 and 46,439 | 69.91% |

Validation of Large Margin Nearest Neighbor Metric Learning

We used a smart contract vulnerability dataset to verify the validity of LMNN in the following format:

The file format can be xlsx or csv. The last five columns of Table 4 are the types of vulnerabilities contained in this smart contract file, where 0 means that there is no such vulnerability and 1 means that there is such a vulnerability.

**Table 4.** Format of smart contract vulnerability dataset content.

| File | Opcode | Denial_Service | Arithmetic | Access_Control | Reentrancy | Time_Manipultion |
|---|---|---|---|---|---|---|
| Smart Contract Filename | opcode sequence | 0 or 1 | 0 or 1 | 0 or 1 | 0 or 1 | 0 or 1 |

In the implemented LMNN code, the opcode sequences are converted into numeric feature vectors using the TfidfVectorizer. Further, the TripletMarginLoss and Stochastic Gradient Descent optimizer are utilized to enhance feature learning and model optimization. After the completion of training, the Mahalanobis distances between the two dataset files are computed, which are subsequently transformed into an overall similarity degree. Two identical files are still used during validation; however, the best result achieved after multiple experiments is 95.82%. This discrepancy is due to the fact that while the LMNN model endeavors to minimize the loss and bring similar samples closer, it does not ensure the exact equivalence of embedding spaces for identical samples; the optimization process and loss function can introduce slight variations. Consequently, even if the files are identical, there might be subtle disparities in the learned embedding space, resulting in a similarity score slightly below 100%. Nonetheless, this outcome provides an adequate reference for the determination of dataset similarity by the DHO.

6.2.2. Model Experiment Preparation

We developed the system based on the Ethereum platform and Solidity programming language. Specifically, we built an Ethereum private chain using Ganache and managed the private chain account through Metamask. The hardware and system environment configurations for the devices are provided in Table 5. The hexadecimal strings denote the addresses of the Ethereum private chains assigned to the three important entities BDTM, DHO, and DPEC in this experiment.

**Table 5.** Hardware and model system environment used for experiments.

| Module | Description |
|---|---|
| CPU | AMD Ryzen 5 5600H |
| Memory | 16 GB DDR4 |
| Ganache | v2.7.1 |
| BDTM_Address | 0 × 3dBFea0e00De0B4bcD5F0BF72Ce232401dd27360 |
| DHO _Address | 0 × 0a88bd1fBC60d7535319D4CB0387beCffDa094aD |
| DPEC_Address | 0 × C64b2F15DF0641DEE028eEbec7216C7D0Fa81788 |

6.2.3. Simulation Testing

We built an Ethereum private chain using Ganache, and Figure 4 shows that we created 20 accounts and allocated 100 ETH per account for testing.

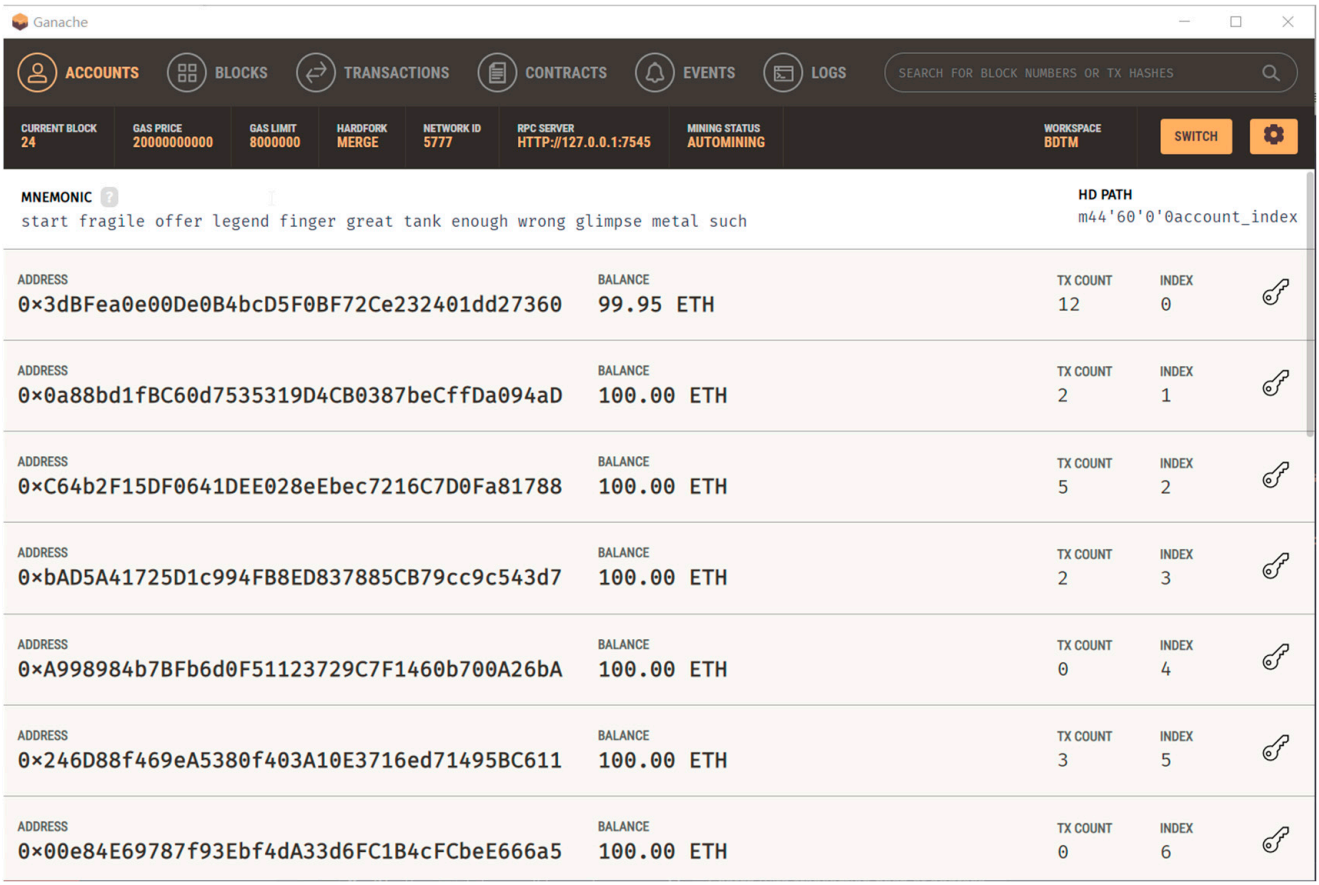
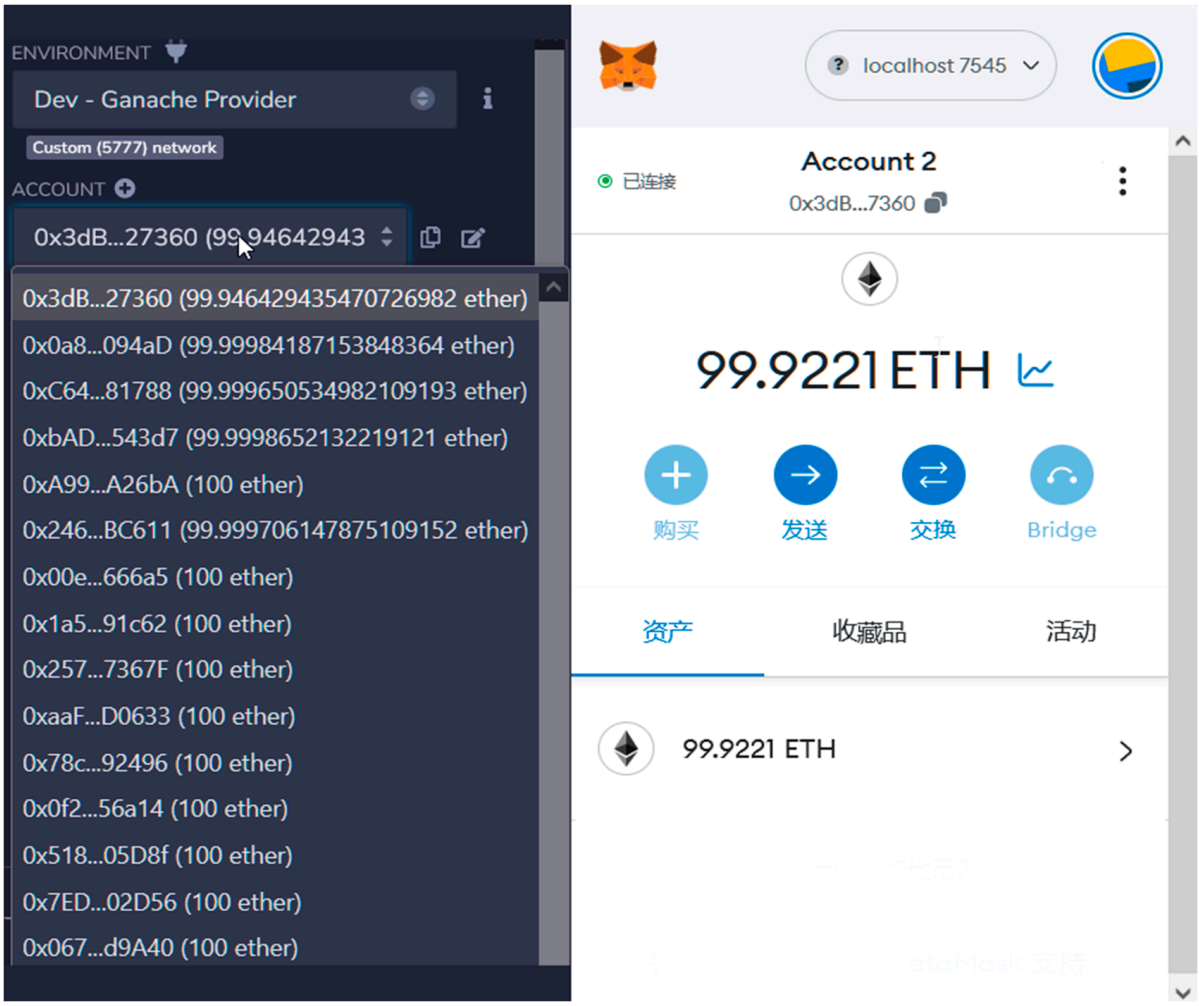



**Figure 4.** Some of the accounts created by Ganache.

After creating the accounts, we deployed the SC using BDTM_Address, connected the local private chain, managed the accounts via Metamask, and tested the SC functions with different accounts. As shown in Figure 5.




(**a**) (**b**)

**Figure 5.** (**a**) The account used for contract testing. (**b**) Metamask connects to the local IP and manages the account.

The function shown in Figure 6 is the main functional function needed to realize the whole BDTM system. The function shown in yellow requires the user to fill in the required parameter value when calling. The function shown in red not only requires the user to fill in the required parameter value when calling, but also has the function of ether transaction function, which needs to be carefully operated when calling, and the ether transfer operation is irreversible once completed.

Figure 7 shows the process of mining blocks after a transaction is completed and the hash value of each transaction.

### *6.3. Gas Consumption Calculation*

We have calculated the gas required to call the BDTM function, as shown in Table 6. According to the calculation rule, 1 Gas = 1 Gwei, i.e., 1 Gas = $1 \times 10^{-9}$ Ether, and the price of Ether on that day is USD 182.09 per Ether. Compared to [12], our gas consumption is slightly lower, and it is difficult to avoid an increase in gas consumption when calling certain string functions. Therefore, we can conclude that the cost of BDTM to execute smart contracts in the Ethereum network is not high, which indicates the practicality of the smart contracts.

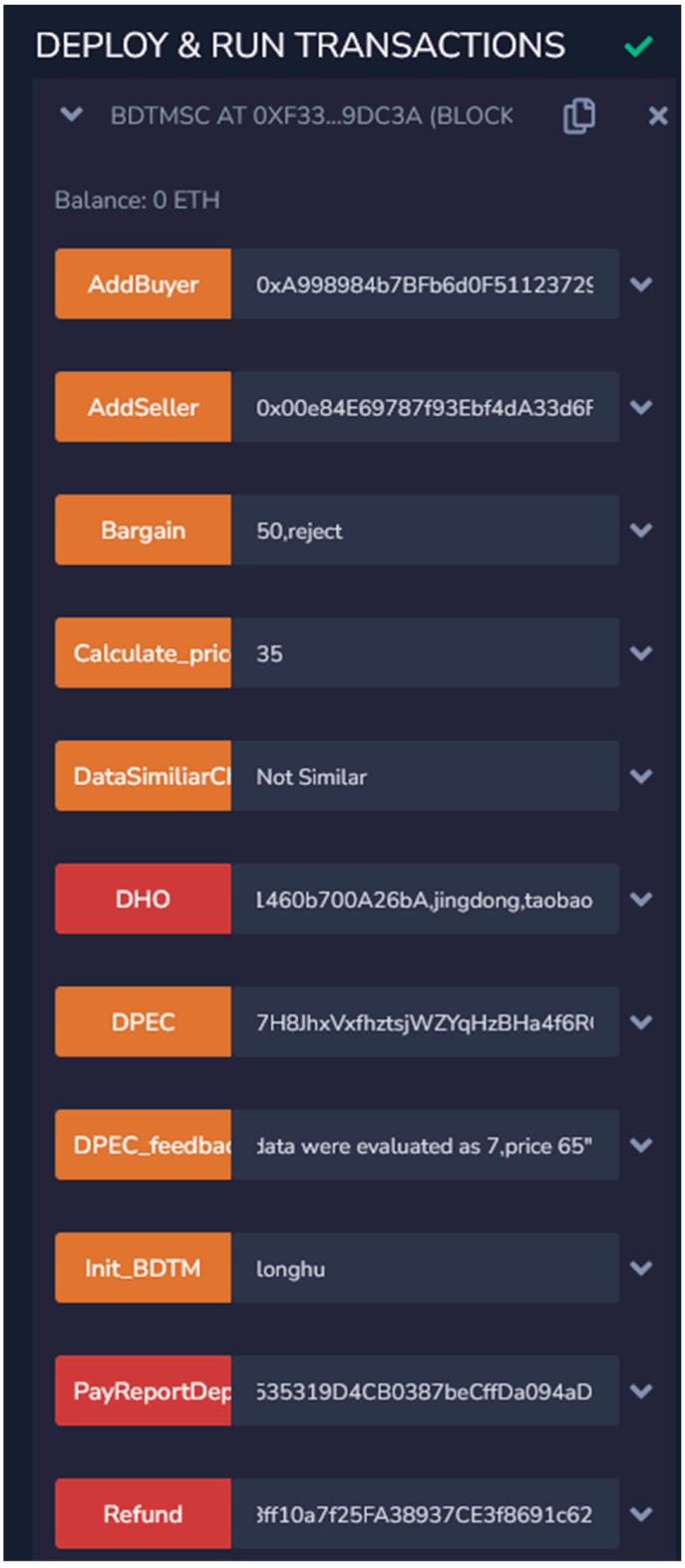



**Figure 6.** Some functions of the SC.


**Figure 7.** (**a**) Block generated by the test contract function. (**b**) TX_HASH of the generated transaction block.

**Table 6.** Gas consumption statistics for invoking BDTM functions.

| Functions | Total Cost (Gas) | Total Cost (USD) |
|---|---|---|
| Init_BDTM | 64,939 | 0.001182474251 |
| DataSimiliarChecking | 26,240 | 0.0047780416 |
| AddBuyer | 185,007 | 0.03368792463 |
| AddSeller | 184,892 | 0.03366698428 |
| Bargain | 373,692 | 0.06804557628 |
| DHO | 268,497 | 0.04889061873 |
| DPEC | 205,649 | 0.03744662641 |
| Refund | 36,245 | 0.00659985205 |
| Reward | 42,684 | 0.00777232956 |
| SetReview | 298,641 | 0.05437953969 |
| PayReportDeposit | 35,495 | 0.00646328455 |

### *6.4. System Model Comparison*

Compared to existing blockchain-based big data trading platforms, our proposed BDTM offers several security and functional integrity benefits. Table 7 presents a comparison, highlighting the advantages of our approach. We utilize the Ethereum platform to register anonymous identities, ensuring the protection of users' privacy. To safeguard users' data ownership, we establish the DHO and employ off-chain machine learning techniques. Data integrity is maintained through the storage of data on IPFS, leveraging its content-addressing feature.

**Table 7.** Comparison of security and functional integrity of different big data trading platforms.

| | Anonymity | Data Ownership Protection | Data Integrity | Price Bargaining Based on SC | Data Evaluation and Pricing | Comment Function |
|---|---|---|---|---|---|---|
| Chen et al. [11] | √ | × | √ | × | × | × |
| Guan et al. [15] | √ | × | √ | × | × | × |
| Dai et al. [17] | × | × | × | √ | × | × |
| Hu et al. [12] | √ | × | √ | √ | √ | × |
| Wei et al. [13] | √ | √ | √ | × | × | × |
| Our proposed scheme | √ | √ | √ | √ | √ | √ |

In contrast, Chen et al. [11] did not take user and data privacy into account in their work. Dai et al. [17] did not provide a guarantee for data integrity. Hu et al. [12] focused on utilizing smart contracts for automated evaluation and pricing, overlooking the impact of market and industry factors on pricing. Wei et al. [13] primarily emphasized data ownership protection while neglecting aspects such as pricing and negotiation. Similarly, Guan et al. [15] proposed smart contract-enabled big data transactions without addressing the issue of allowing users to freely comment on purchased data goods. Our BDTM, however, incorporates this functionality.

## 7. Conclusions

In this paper, we proposed a decentralized data trading model called BDTM, which is based on the Ethereum platform. We analyzed the limitations of centralized data trading platforms, such as the risk of privacy breaches and malicious tampering of user data, the potential for malicious behavior of users to resell data, and the possibility of disagreement over the price of data transactions. To address these issues, we designed an architecture for the data trading platform that involves multiple trusted entities and smart contracts. This architecture is deployed on Ethereum, leveraging the anonymity of users on the Ethereum network. Establishing DPEC and DHO provides users with data quality evaluation services and data ownership protection, aiming to enhance the security, fairness, and efficiency of the system. Additionally, it supports dynamic bargaining, data reward distribution, and the reporting of malicious behavior. We implement the proposed big data trading system and analyze its security and functional integrity. According to a large number of experimental verification and result analysis in Section 6, the proposed BDTM achieves all functions in full accordance with the above design details. The system security and system functional integrity of BDTM are better than the current blockchain-based big data transaction model, and BDTM covers all the processes of big data transactions, reflecting the advantages of BDTM in security and functional integrity.

In future work, we will explore combining BDTM and WEB3 technologies to develop BDTM into a Decentralized Application (DApp) and port it to mobile devices. This is because there is a huge number of mobile device users currently, and developing it into a DApp allows users to conveniently access it anytime, anywhere. Furthermore, we plan to adapt the mature BDTM to a consortium blockchain because the use of consortium chains is growing in various scenarios, incorporating the consensus algorithm of the consortium blockchain to enhance scalability, security, and privacy protection within the system. In trading platforms, there are also other options besides Ethereum. DEX (Decentralized Exchange) can also be used for large data trading. Compared to Ethereum, DEX supports the trading of multiple decentralized tokens, allowing more digital assets to trade freely in the market and promoting liquidity of assets, while Ethereum only supports Ether trading. However, Ethereum has a large, rich ecosystem, including developer communities, tools, and applications, which makes it possible to obtain more support and tools when conducting large data trading through Ethereum. Their downsides are also quite evident. DEX transactions have higher fees, and trading is peer-to-peer, so liquidity is relatively

low, meaning some assets may have higher buy and sell price gaps when traded on DEX. For Ethereum, some security vulnerabilities in smart contract programming could lead to smart contracts being attacked, and in the past, some smart contract bugs have resulted in significant fund losses. The best choice between these options needs to comprehensively consider the task and user requirements.

**Author Contributions:** Conceptualization, Z.Y.; writing—original draft preparation, Z.Y.; writing—review and editing, supervision, C.Z.; format review and proofreading, S.Y.; data curation, B.X.; data curation, X.W. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the National Science Foundation of China, grant number 61703426, and by the Innovation Capability Support Plan of Shaanxi, China, grant number 2020KJXX-065.

**Data Availability Statement:** We uploaded the code used in the manuscript to https://github.com/yangzhou-code/LMNN (accessed on 7 October 2023).

**Acknowledgments:** We greatly appreciate the members of our laboratory for their help in writing the manuscript.

**Conflicts of Interest:** The authors declare that they have no known competitive financial interest or personal relationship that may affect the work reported in this paper.

## References

1. Ibrahim, R.F.; Al-Haija, Q.A.; Ahmad, A. DDoS Attack Prevention for Internet of Thing Devices Using Ethereum Blockchain Technology. *Sensors* **2022**, *22*, 6806. [CrossRef] [PubMed]
2. Sagiroglu, S.; Sinanc, D. Sinanc. Big data: A review. In Proceedings of the 2013 International Conference on Collaboration Technologies and Systems (CTS), San Diego, CA, USA, 20–24 May 2013.
3. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. Available online: http://bitcoin.org/bitcoin.pdf (accessed on 4 October 2023).
4. Al-Haija, Q.A.; Alnabhan, M.; Saleh, E.; Al-Omari, M. Applications of blockchain technology for improving security in the internet of things (IoT). In *Blockchain Technology Solutions for the Security of IoT-Based Healthcare Systems*, 1st ed.; Academic Press: Amsterdam, NL, USA, 2023; pp. 199–221.
5. Qaqish, E.; Aranki, A.; Al-Haija, Q.A.; Qusef, A. Security Comparison of Blockchain and Cloud-based Identity Management: Considering the Scalability Problem. In Proceedings of the 2023 International Conference on Inventive Computation Technologies (ICICT), Lalitpur, Nepal, 26–28 April 2023.
6. Odeh, A.; Keshta, I.; Al-Haija, Q.A. Analysis of Blockchain in the Healthcare Sector: Application and Issues. *Symmetry* **2022**, *14*, 1760. [CrossRef]
7. Zyskind, G.; Nathan, O. Decentralizing Privacy: Using Blockchain to Protect Personal Data. In Proceedings of the 2015 IEEE Security and Privacy Workshops, San Jose, CA, USA, 18–20 May 2015.
8. Chen, W.; Chen, Y.; Chen, X.; Zheng, Z. Toward Secure Data Sharing for the IoV: A Quality-Driven Incentive Mechanism with On-Chain and Off-Chain Guarantees. *IEEE Internet Things* **2020**, *7*, 1625–1640. [CrossRef]
9. Jung, T.; Li, X.Y.; Huang, W.C.; Qiao, Z.Y.; Qian, J.W. AccountTrade: Accountability Against Dishonest Big Data Buyers and Sellers. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 223–234. [CrossRef]
10. Jung, T.; Li, X.Y.; Huang, W.C.; Qian, J.W.; Chen, L.L. AccountTrade: Accountable protocols for big data trading against dishonest consumers. In Proceedings of the IEEE INFOCOM 2017—IEEE Conference on Computer Communications, Atlanta, GA, USA, 1–4 May 2017.
11. Chen, J.; Lv, Z.; Song, H. Design of personnel big data management system based on blockchain. *Future Gener. Comput. Syst.* **2019**, *101*, 1122–1129. [CrossRef]
12. Hu, D.H.; Li, Y.F.; Pan, L.X.; Li, M.; Zheng, S.L. A blockchain-based trading system for big data. *Comput. Netw.* **2021**, *191*, 107994. [CrossRef]
13. Wei, X.; Li, X. Data resource protection based on smart contract. *Comput. Secur.* **2020**, *98*, 102004.
14. Park, J.S.; Youn, T.Y.; Kim, H.B.; Rhee, K.H.; Shin, S.U. Smart Contract-Based Review System for an IoT Data Marketplace. *Sensors* **2018**, *18*, 3577. [CrossRef] [PubMed]
15. Guan, Z.; Shao, X.; Wan, Z. Secure Fair and Efficient Data Trading Without Third Party Using Blockchain. In Proceedings of the 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), Halifax, NS, Canada, 2–7 July 2018.
16. Gao, G.; Xiao, M.; Wu, J.; Zhang, S.; Huang, L. DPDT: A Differentially Private Crowd-Sensed Data Trading Mechanism. *IEEE Internet Things* **2020**, *7*, 751–762. [CrossRef]

17. Dai, W.; Dai, C.; Choo, K.-K.R.; Cui, C.; Zou, D. SDTE: A Secure Blockchain-Based Data Trading Ecosystem. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 725–737. [CrossRef]
18. Li, X.X.; Peng, J.H.; Gao, S.Q.; Shi, Z.K.; Li, C.P. Achieving fair and accountable data trading for educational multimedia data based on blockchain. *Wirel. Netw.* **2022**, *28*, 1–13. [CrossRef]
19. Zhao, Y.Q.; Yu, Y.; Li, Y.N.; Han, G.; Du, X.J. Machine learning based privacy-preserving fair data trading in big data market. *Inform. Sci.* **2019**, *478*, 449–460. [CrossRef]
20. Guan, Z.; Zhao, Y.; Li, D.; Liu, J. TBDCT: A Framework of Trusted Big Data Collection and Trade System Based on Blockchain and TSM. In Proceedings of the 2018 IEEE International Conference on Smart Cloud (SmartCloud), New York, NY, USA, 21–23 September 2018.
21. Sober, M.; Scaffino, G.; Schulte, S.; Kanhere, S.S. A blockchain-based IoT data marketplace. *Clust. Comput.* **2022**, *25*, 3523–3545. [CrossRef]
22. An, B.; Xiao, M.; Liu, A.; Xu, Y.; Zhang, X. Secure Crowdsensed Data Trading Based on Blockchain. *IEEE Trans. Mob. Comput.* **2023**, *22*, 1763–1778. [CrossRef]
23. Buterin, V. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. Available online: https://ethereum.org/en/whitepaper/ (accessed on 11 October 2023).
24. Al-Haija, Q.A.; Alsulami, A.A. High Performance Classification Model to Identify Ransomware Payments for Heterogeneous Bitcoin Networks. *Electronics* **2021**, *10*, 2113. [CrossRef]
25. Larimer, B.D. Transactions as Proof-of-Stake. Available online: https://cryptochainuni.com/wp-content/uploads/Invictus-Innovations-Transactions-As-Proof-Of-Stake.pdf (accessed on 8 October 2023).
26. Szabo, N. Smart Contracts: Building Blocks for Digital Markets. *Extropy* **1996**, *18*, 28.
27. IPFS Is the Distributed Web. Available online: https://ipfs.io (accessed on 11 October 2023).
28. Rubinstein, A. Perfect Equilibrium in a Bargaining Model. *Econometrica* **1982**, *50*, 97–100. [CrossRef]
29. Mirchev, M.; Mircheva, I.; Kerelovska, A. The Academic Viewpoint on Patient Data Ownership in the Context of Big Data: Scoping Review. *J. Med. Internet Res.* **2020**, *22*, 22214. [CrossRef] [PubMed]
30. Weinberger, K.Q.; Saul, L.K. Distance Metric Learning for Large Margin Nearest Neighbor Classification. *J. Med. Internet Res.* **2009**, *10*, 207–244.