# CPchain: A Copyright-Preserving Crowdsourcing Data Trading Framework Based on Blockchain

Dingjie Sheng[1], Mingjun Xiao[1*], An Liu[2], Xiang Zou[3], Baoyi An[1], and Sheng Zhang[4]

[1]School of Computer Science and Technology & Suzhou Institute for Advanced Study, University of Science and Technology of China, Hefei, China
[2]School of Computer Science and Technology, Soochow University, Suzhou, China
[3]The Third Research Institute of the Ministry of Public Security, Shanghai, China
[4]State Key Lab. for Novel Software Technology, Nanjing University, P.R., Nanjing, China
[*]Correspondence to: xiaomj@ustc.edu.cn

***Abstract*—Crowdsourcing data trading is a novel paradigm in which the crowdsourcing technology is adopted to collect big data for trading. At present, existing crowdsourcing data trading systems usually depend on a trusted broker and haven't considered the truthfulness and the quality of data (QoD) simultaneously. Besides, copyright protection is the another issue that has not been properly addressed. To tackle these problems, we propose a Copyright-Preserving crowdsourcing data trading framework based on Blockchain, named CPchain, which mainly includes a smart contract. We design an auction algorithm based on semantic similarity to guarantee the truthfulness and individual rationality while ensuring QoD. Moreover, we combine digital fingerprint technology with blockchain to protect data copyright without a third-party certification authority. Furthermore, we develop a simple prototype of our proposed trading framework on the Ethereum test network. We have carried out a lot of experiments to demonstrate the significant performances of our framework.**

***Index Terms*—Blockchain,Crowdsourcing,Copyright protection,Data trading,Digital fingerprint**

## I. Introduction

In recent years, big data has gradually become the sought-after resource due to its huge potential economic value [1]. However, most data are held by only a few institutions and companies and are regarded as private resources rather than shared with people in need. Consequently, many data trading systems have proliferated, such as Datatrading, Terbine, CitizenMe, and DataExchange. To facilitate data sharing, a novel data trading paradigm, named Crowdsourcing Data Trading (CDT), is proposed, in which the crowdsourcing [2], [3] technology is utilized to trade data [4], [5].

Generally speaking, a typical CDT (e.g., Factual, Infochimps) consists of three parties: a data trading broker, some data sellers and data consumers. Consumers submit their requirements to the broker. Then, the broker selects some sellers according to certain strategies. These sellers will sell the data to consumer to get a certain reward. There have been some works to design the CDT systems. For example, [6] proposes a general dataset purchasing framework, named CROWDBUY and CROWDBUY++, based on crowdsourcing, with which a consumer can efficiently buy desired data from available sellers. An incentive-based crowdsourced entity collection framework, called CrowdEC which encourages sellers to provide more distinct items, is proposed in [7]. However, most of them depend on a trusted third-party (i.e., a trusted broker). It makes users doubt the reliability of the trading,which will reduce users' enthusiasm for participating in trading.

At present, blockchain as an emerging technology has attracted a lot attention due to its reliability and transparency [8]. It is almost impossible for users to tamper with the data on the blockchain. Through the blockchain, mutually distrusted users can complete the data exchange and currency transfer securely without a trusted third-party, avoiding high legal and transaction costs. Actually, blockchain can deploy complex program codes to be executed automatically, called smart contracts [9]. These smart contracts can force users to fulfill their respective obligations. Hence, they can replace trusted third-parties to play the role of brokers in data trading system. There have emerged some novel data trading systems based on blockchain. For example, [10] proposes the RADToken which combines auction mechanism and blockchain to construct a truthful, rational and decentralized trading system. A blockchain-based data trading ecosystem which reduces the challenge of securing the dataset to the challenge to secure the data processing, is proposed in [11].

There are two grave challenges for designing a blockchain-based CDT system. The first challenge is to guarantee the truthfulness and data quality of trading simultaneously. Although blockchain-based CDT can ensure the credibility of trading, it cannot guarantee the truthfulness. This is because sellers might report a fake bid price to achieve more profits during the trading process. And the consumer may pay as little as possible to increase their profits, which will damage the benefits of sellers. Besides, we also need to match the data requirements (e.g., volume, type, format, etc) between sellers and the consumer which enables consumer to purchase data that meets their needs, i.e., ensuring the quality of data (QoD) in trading. Furthermore, We must ensure that the matching process is open, transparent and credible. The second

978-1-7281-6607-0/20/$31.00 ©2020 IEEE

challenge is copyright protection issue. As digital products can be easily counterfeited or duplicated. Specifically, if the purchased data is pirated by buyers, the value of data from the original data owners as sellers will be significantly affected, leading to the unwillingness of data owners to participate in the market. Thus, data copyright protection schemes must be designed to ensure the owners' legal rights.

To tackle the above-mentioned challenges, we propose a Copyright-Preserving crowdsourcing data trading framework based on Blockchain (CPchain). In CPchain, we leverage the blockchain to conduct credible trading between sellers and the consumer. Then, we introduce the auction mechanism to encourage sellers to participate in trading honestly, which ensures the truthfulness and individual rationality. In addition, we utilize semantic similarity to measure the requirement matching degree between sellers and the consumer to ensure QoD. Furthermore, we propose a novel digital fingerprint protocol based on blockchain, which addresses the issue of copyright protection without a third-party certification authority (CA). More specifically, the contributions of the paper are summarized as follows:

1) We propose a Copyright-Preserving crowdsourcing data trading framework based on Blockchain, named CPchain. In the absence of a truthful broker, it can enable mutually untrusted participants to conduct truthful and credible data trading, while ensuring the quality and copyright of data.
2) We design a semantic-similarity-based auction mechanism to select the winners and determine the payments through smart contract. It settles the problem of requirement matching by semantic similarity, which ensures the QoD of trading. Meanwhile, the proposed auction mechanism ensures the truthfulness and individual rationality.
3) We propose a copyright protection scheme that combines blockchain and digital fingerprinting technology to ensure the data owners' legal rights. It not only achieves copyright protection but also eliminates the problem of depending on third-party CA in traditional digital fingerprint protocols.
4) We implement a prototype of the CPchain and deploy it on the Ethereum test network. Comprehensive experiments show the feasibility of CPchain.

The remain of the paper is organized as follows. In Sect. II, we illustrate the design of framework we propose. After that, we elaborate on the details of our proposed semantic-similarity-based auction mechanism and blockchain-based digital fingerprint protocol in Sect. III and Sect. IV, respectively. In Sect. V, we make a performance analysis on our proposed framework. Then, we present simulations and evaluations in Sect. VI. After reviewing the related work in Sect. VII, we finally conclude in Sect. VIII.

## II. Framework Design

We choose the Ethereum as the underlying blockchain platform to develop and deploy smart contract. Each user in Ethereum has an Externally Owned Accounts (EOA) through which user can transfer *Ether* (i.e., the currency in the Ethereum) and invoke the function in smart contract. In addition, when a user invoke a function in smart contract, it will consume a certain of fee which called *gas* in Ethereum. A *transaction* in Ethereum is a message sent from one account to another. The transaction can contain binary data and ether. There will be a gas limit for each transaction. If the limit is exceeded, the transaction will fail. Besides, there are some built-in keywords in smart contracts (e.g., *require, modifier, msg.sender,* etc) by which we can constrain the execution of each function in the smart contract, i.e., only the specified user can invoke within the designated time slot, otherwise, the function call request will be rejected [9].

First of all, each user invokes the registration function to register in the smart contract in order to participate in the trading. Before starting a trading, the consumer and sellers will make a text description $des_c$ and $des_i$ of the data they want buy/sell. They can get the respective semantic feature vectors of $des_c$ and $des_i$ (i.e., $F_c$ and $f_i$) through the same pre-trained feature vector generator (e.g., Doc2Vec [12] and Glove [13]). Besides, the consumer and sellers will also generate their digital fingerprints in advance. Note that, for each trading, the consumer must generate a unique digital fingerprint for it. Each trading is started by only one consumer, and each participating user needs to submit a deposit to the smart contract to prevent its misconduct.

### A. Framework Overview

As shown in Fig. 1, the CPchain which mainly contains the following five entities:

*Consumer*. The consumer is the buyer of the CDT system. At the beginning, it will submit a purchase request $R_c$ to the smart contract to start a trading, denoted as $R_c$ = $\{PK_c$, $F_c$, $M_{thre}$, $encFP_c\}$. Here, $PK_c$ is the consumer's public key. $F_c$ is the feature vector of the $des_c$. $encFP_c$ is the consumer's digital fingerprint which encrypted by $PK_c$. And the total quality of data purchased by consumer should not be less than $M_{thre}$. The consumer is responsible for invoking the auction function to select the winners and calculate the corresponding payments. Besides, the consumer eventually need to pay sellers.

*Seller*. We denote the set of sellers as $S$ = $\{s_1$, $s_2$, $s_3$,..., $s_n\}$. For each seller $s_i$, he will submit a participation request, denoted as a tuple $B_i$ = $\{f_i$, $Encb_i$, $FP_{copyright}\}$. $f_i$ is the feature vector of $des_i$. $s_i$ encrypts his bid by $PK_c$ to get $Encb_i$ before submitting $B_i$. $FP_{copyright}$ is the digital fingerprint used by the seller to identify the copyright of data. In addition, each seller needs to calculate the semantic similarity $m_i$ between $f_i$ and $F_c$. Then, the seller who selected by the auction algorithm needs to securely deliver data to the consumer in order to get reward.

*IPFS*. It is a peer-to-peer distributed file system. When a seller uploads the data to IPFS, it will return a unique corresponding hash address, through which the consumer can query and download data.

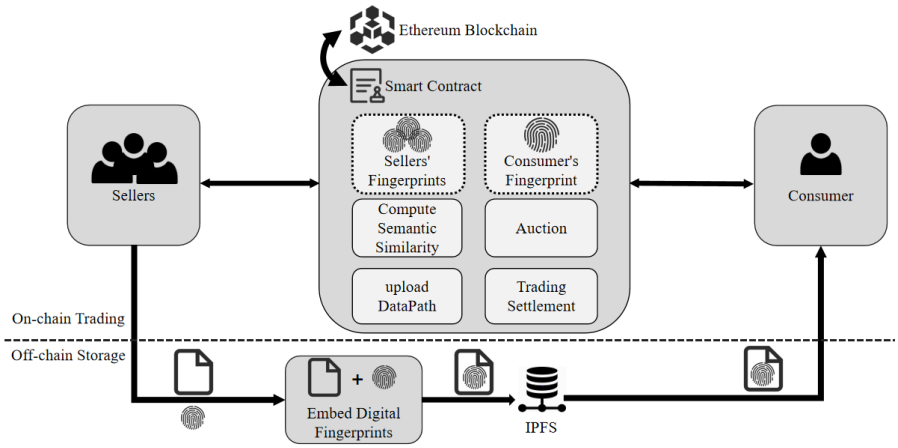


Fig. 1. The CPchain framework

*Smart Contract*. The smart contract replaces trusted third-party to play the role of broker. The consumer and sellers interact through the smart contract. Moreover, user information (e.g., list of registered users, $R_C, B_i$, etc) and trading records are permanently stored in the smart contract.

*Digital Fingerprint Embedding Algorithm*. We assume that there have been some robust fingerprint embedding algorithms for different types of data. Each seller utilizes this algorithm to embed digital fingerprints into their data to get watermarked data (i.e., data with fingerprint).

### B. The Workflow of the CPchain

The workflow of CPchain mainly includes the following phases:

**Phase 1: Establishing Trading.** The consumer submits the purchase request $R_C$ with deposit to the smart contract. Meanwhile, he will set the executable time slots (i.e., start time and end time) for each function. Then, the smart contract notifies all sellers to participate in the trading. After that, each seller will submit their participation request $B_i$ to the smart contract.

**Phase 2: Calculating Semantic Similarity.** Each seller calculates the semantic similarity between himself and the consumer. The task of semantic similarity calculation is to return the similarity value based on the feature vectors of two texts. Since the parameters (i.e., $F_c$ and $f_i$) and program codes have been uploaded and stored in the smart contract in advance, sellers cannot tamper with them. Thus, sellers can only invoke the program by rule, which ensures that sellers cannot cheat. The initial semantic similarity of each seller is 0. If a seller does not perform the function as specified, his semantic similarity will be 0 and he will be eliminated in the subsequent auction process. Procedure 1 shows the pseudo code of Semantic Similarity Calculation. There are two constraints to invoke this function, i.e., the caller must be a registered seller (i.e., $S$) and invoke the function within the executable time slots. $msg.sender$ is the built-in parameter of smart contract, representing the EOA that invoked the function (i.e., the seller). $now$ is also a built-in parameter, which can get the current timestamp to judge whether it is in the legal execution time slot. Line 3 is the equation for calculating the semantic similarity. $MS$ is a *Mapping* for storing the semantic similarity of each winner.

1: computeSimilarity($F_c, f_i$):
2: require $msg.sender \in S$ and $now \in [startTime, endTime]$;
3: compute semantic similarity as follows:

$$m_i = 1/\sum_{j=1}^{k} (x_j - y_j)^2$$

4: set $MS[msg.sender] = m_i$;

**Procedure 1:** Compute Semantic Similarity

**Phase 3: Auction.** The consumer downloads sellers' encrypted bids $encBids = \{Encb_i \mid s_i \in S\}$ from the smart contract, and decrypts them with his private key. Then, the consumer uses the plaintexts of bids $bids = \{b_i \mid s_i \in S\}$ and $MS$ as parameters to invoke the auction function, so as to select the winners $W \subseteq S$ and calculate the payments $P = \{p_i \mid w_i \subseteq W\}$. The consumer prepays the calculated total payment $P_{total}$ to the smart contract. After that, smart contract notifies the auction results to all winners. The auction mechanism will be explained in detail in Sect. III.

**Phase 4: Delivery of Copyrighted Data.** Each winner gets the consumer's fingerprint from the smart contract. Through the fingerprint embedding algorithm, each winner embeds the $FP_C$ and $FP_{copyright}$ into his data $data_i$ to get watermarked data $data_i^*$. After that, each winner uploads $data_i^*$ to IPFS to obtain the hash storage path $Addr_i$. Then, each winner uploads $Addr_i$ and $Encdes_i$ ($des_i$ encrypted by $PK_c$) to the smart contract. The consumer gets the $Addrs = \{Addr_i \mid w_i \in W\}$ and $encDes = \{Encdes_i \mid w_i \in W\}$from the smart contract, and then, downloads the data $Data = \{data_i^* \mid w_i \in W\}$ from IPFS. The digital fingerprint protocol we designed will be detailed in Sect. IV.

**Phase 5: Trading Settlement.** Finally, it is the trading settlement phase, as shown in Procedure 2. Each seller who

calculateRefund():
  require $msg.sender = consumer$;
  for each $s_i$ not in $W$ do:
    transfer $deposit$ to $s_i$;
  for each $w_i$ in $W$ do:
    \\ *If the condition satisfied, the i-th winner has fulfilled duties and vice versa.*
    if $Addr_i$ isIn $Addrs$ do:
      transfer $T_i = deposit + p_i$ to $W_i$;
    else:
      $refund = refund + deposit + p_i$;
  transfer $T_c = deposit + refund$ to consumer;

**Procedure 2:** Trading settlement

loses the auction will receive a refund of deposit $deposit$. If a winner fulfills duties as agreed, the smart contract will transfer a certain amount i.e., $T_i = deposit + p_i$ to him. Otherwise, the winner will not get the expected reward, and his deposit will also be fined to the consumer. Finally, the consumer receives $T_c = deposit + refund$.

## III. Semantic-Similarity-based Auction

### A. Problem Formulation

We regard the similarity of each seller as their QoD. Then, the goal of the Semantic-Similarity-based Auction (SSA) is to find a winner set so that the Sum of Semantic similarities of All Winners (SSAW) is not less than the threshold $M_{thre}$ while minimizing costs simultaneously. To formalize this problem, we let the $\delta^W$ denote the sum of the semantic similarities of all winners, where

$$\delta^W = \sum\nolimits_{w_i \in W} m_i \tag{1}$$

Besides, due to the truthfulness of sellers (will be proven in Section 5), we regard the seller's bid as his cost (i.e., $c_i = b_i$) and use $C^W$ to indicate the total costs. Then, this problem can be formalized as follows:

$$min \quad C^W = \sum\nolimits_{w_i \in W} c_i = \sum\nolimits_{w_i \in W} b_i \tag{2}$$

$$s.t. \quad W \subseteq S \tag{3}$$

$$\delta^W \geq M_{thre} \tag{4}$$

### B. Winner Selection

The winner selection can be transformed into a 0-1 knapsack problem, which is NP-hard. Thus, we propose a greedy algorithm to solve the problem. We first define a utility function $E(W)$ which indicates the current total semantic similarity of all winners:

$$E(W) = min\{\delta^W, M_{thre}\} \tag{5}$$

Then, the marginal utility of $w_i \in S \setminus W$ can be defined as:

$$E_i(W) = E(W \cup \{w_i\}) - E(W) \tag{6}$$

The design of the winner selection is illustrated in Procedure 3. The greedy strategy of the SSA is to select the seller with the highest ratio of marginal benefit to bid (i.e., $argmax\frac{E_j(W)}{b_j}$) as the winner in each iteration.

winnerSelection($MS, bids$):
  require $msg.sender = consumer$ and $now \in [startTime, endTime]$;
  while $E(W) < M_{thre}$ do:
    set $w_{opt} \leftarrow argmax\frac{E_j(W)}{b_j}\ from\ S$;
    $W \leftarrow W \cup w_{opt}$;
    $E(W) \leftarrow E(W) + E_{opt}(W)$;
    $C^W \leftarrow C^W + b_{opt}$;
    $S \leftarrow S \setminus w_{opt}$;
  return $W, C^W$;

**Procedure 3:** Winner Selection

paymentDetermination($MS, bids$):
  require $msg.sender = consumer$ and $now \in [startTime, endTime]$;
  for each $w_i$ in $W$ do:
    while $E(W^*) < M_{thre}$ do:
      set $w_h \leftarrow argmax\frac{E_j(W^*)}{b_j}\ from\ S_{-i}$;
      $W^* \leftarrow W^* \cup w_h$;
      $E(W^*) \leftarrow E(W^*) + E_i(W^*)$;
      $S_{-i} \leftarrow S_{-i} \setminus w_h$;
      $p_i = max\{\frac{E_i(W^*).b_h}{E_h(W^*)}, p_i\}$
  return $P$;

**Procedure 4:** Payment Determination

### C. Payment Determination

As Procedure 4 shows, we let $S_{-i}$ denote all sellers except $w_i$ and $W^*$ denotes the winner set over $S_{-i}$ of the $h_{th}$ iteration. We assume the $w_h$ is the winner in the $h_{th}$ iteration. Then, according to the auction rules, if $w_i$ wants to be selected as a winner in the $h_{th}$ iteration, $\frac{E_i(W^*)}{b_i} \geq \frac{E_h(W^*)}{b_h}$ must be satisfied. Then, we can get the critical value of $w_i$ in current iteration $\frac{E_i(W^*).b_h}{E_h(W^*)}$. We assume that after the $H$ rounds of iteration, the algorithm terminates. Then, the critical payment $p_i$ of $w_i$ is the maximum critical value:

$$p_i = max\{\frac{E_i(W^*).b_h}{E_h(W^*)} \mid h = 1, 2, 3, ..., H\} \tag{7}$$

## IV. Blockchain-based Digital Fingerprint Protocol

### A. The principle of BDFP

Our proposed Blockchain-based Digital Fingerprint Protocol (BDFP) is built on top of homomorphic encryption. Homomorphic encryption scheme is a public-key cryptosystem with homomorphic properties. It can perform operations on encrypted data without decrypting them. Let $\otimes$ and $\oplus$ be the multiplication and addition operations in the prime field $\psi_q$. For $\forall x, y \in \psi_q$, $x \otimes y \stackrel{def}{=} xy \bmod q$ and $x \oplus y \stackrel{def}{=} x + y \bmod q$. Besides, let $m_1, m_2 \in \psi_q$ are two plaintexts, and $E[.]$ is the homomorphic encryption operation. Then, the homomorphic encryption scheme satisfies $E[m_1] \otimes E[m_2] = E[m_1 \oplus m_2]$.

The details of the BDFP are presented as follows. Firstly, the consumer will encrypt his fingerprint with his public key $PK_c$ to get $encFP_C$ which will be sent to sellers. Then, sellers

embed their own fingerprints $FP_{copyright}$ into their data. After that, sellers encrypt their data by $PK_c$. At last, sellers embed $encFP_C$ into their encrypted data.

**Input:**
Raw data and fingerprints, $data_i, encFP_C, F_{copyright}$;
**Output:**
The watermarked data, $data_i^*$;
**phase 1:** consumer sends his fingerprint to smart contract with the purchase request.
1: the consumer submits the purchase request $R_c = \{PK_c, F_c, M_{thre}, encFP_c\}$.
**phase 2:** each winner conducts the watermarked data after auction process.
2: for each $w_i$ in $W$ do:
3: $w_i$ gets $encFP_c$ from smart contract and conducts $data_i^*$ as Eq. (8) and Eq. (9);
**phase 3:** track pirate.
4: when a data owner finds illegal pirated data $data_{pira}$ after the trading do:
5: gets the fingerprint $FP^* = extract(data_{pira})$.
\\ *Find the illegal consumer from Ethereum trading records.*
6: for each $R_c$ in $Reqs$ do:
7: if $R_c.encFP_c = enc(FP^*, R_c.PK_c)$
8: the consumer is the pirate.
9: when a seller uploads the data hash path to smart contract:
10: each data owner downloads the $data_i^*$ from IPFS
11: if data owner can detect a encrypted fingerprint $FP^{'}$ that:
12: $FP^{'} = enc(FP_{copyright}, PK_c)$
13: this seller is a pirate because he want to resell the data from others
**phase 4:** the seller submits the evidences to the arbiter for accountability.
**phase 5:** arbiter verifies the evidences through the Ethereum to make judgment.

**Protocol 1:** the BDFP Protocol

$$data_i^{'} = data_i \odot FP_{copyright} \tag{8}$$

$$\begin{aligned} data_i^* &= enc(data_i^{'} \odot FP_c, PK_c) \\ &= enc(data_i^{'}, PK_c) \odot encFP_c \end{aligned} \tag{9}$$

Eq. (8) and Eq. (9) indicates the homomorphic encryption-based digital fingerprint embedding process. The digital fingerprint insertion operation is denoted as $\odot$ and $enc()$ is a privacy homomorphism with respect to the insertion operator. For example, the well known RSA public key cryptosystem is a privacy homomorphism with respect to multiplication.

### B. The Design of BDFP

Protocol 1 describes the process of BDFP. **Phases 1-2** are the fingerprint embedding process. The consumer uploads his digital fingerprint to the smart contract when he starts the trading. Each winner will embed the $encFP_c$ and $FP_{copyright}$ into his data after he is selected. **Phases 3-5** are the accountability process. As shown in Fig. 2 and lines 4-8, if data owner finds a pirated data $data_{pira}$ (it's a plaintext of $data_i^*$) from anywhere (e.g., Internet), he can utilize the $extract()$ algorithm to extract the fingerprint $FP^*$ from the pirated data. Then, the data owner queries the owner of the fingerprint (i.e., the illegal pirate) from the smart contract. Note that, some sellers may want to resell the data from others for illegal income. Therefore, data owners have the motivation to verify whether the data uploaded by sellers is legal. Besides, as shown in Fig. 3 and lines 9-13, it can indicate the seller is a pirate if data owner can extract his copyright fingerprint from the data uploaded by this seller. Obviously, the data owner can submit the $FP^*$ or $FP^{'}$, $data_{pira}$ and trading records which stored on blockchain as evidences to the arbiter. If the verification is passed, the accountability is successful.

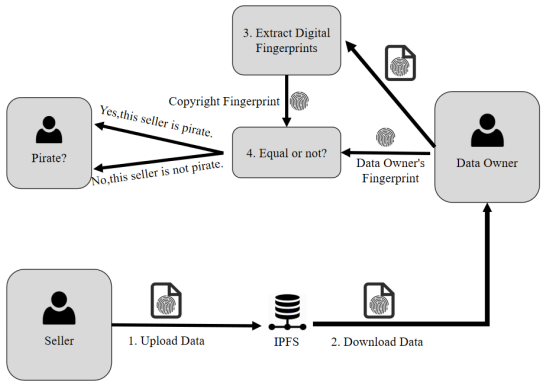


Fig. 2. Case one

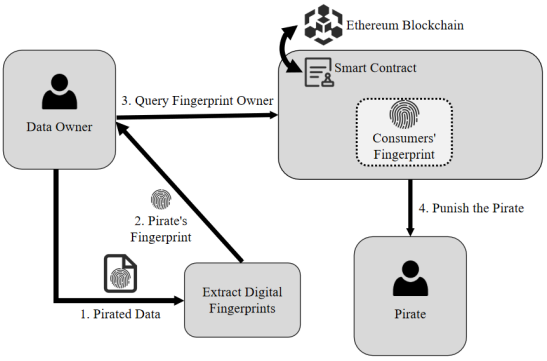


Fig. 3. Case two

## V. Performance Analysis

### A. Security of the CPchain

We design the following mechanisms to ensure the security of the CPchain:

**Deposit.** Each user who participates in the trading is required to pay a deposit, which can urge users to complete the entire trading process as required. Once a user violates the rules, he will get nothing and his deposit will be fined.

**Function Execution Constraint.** Through the built-in *require* and *modifier* keywords of solidity, we can constrain each function in the smart contract that can be invoked only by the specified user within the designated time slot, otherwise, the function call request will be rejected.

**Data Storage.** Sellers will upload their own data which is encrypted by the consumer's public key to IPFS, and submit the returned hash storage paths to the consumer through the smart contract. IPFS can guarantee non-repudiation and tamper-proofing of data. Meanwhile, the encryption technology makes it impossible for anyone except the consumer to decrypt the data, and thus, the confidentiality of the data is guaranteed. Moreover, IPFS is a free P2P storage system, which solves the problem of high storage costs on the blockchain.

**Verification of Data.** Each winner sends the $Encdes_i$ to consumer along with the $Addr_i$ as a commitment to the data. If consumer finds that the purchased data is obviously not up to the requirements, the $Encdes_i$ can be decrypted to get $des_i$. Then, consumer generates corresponding feature vector of $des_i$ and compares it with the $f_i$ which is stored on the smart contract. If it's consistent, the $Encdes_i$ can be used as an evidence to prove that the winner has not submit data that matches his description; otherwise, it directly indicates that the winner violates the rules. In addition, each seller uses consumer's public key to encrypt his description, bid and data, etc. So no one can get the sensitive information except the consumer and sellers.

### B. Security of the BDFP

In the light of [14], [15], the BDFP achieves copyright protection and solves the following difficult problems in the traditional digital fingerprint protocols:

*Definition 1:* **Unbinding Problem**. Driven by the temptation of benefit, a dishonest data owner may, after discovering an illegal digital copy, port the watermark to another digital work of higher value, thus creating counterfeit piracy.

*Theorem 1:* The BDFP has tackled the Unbinding problem.

*Proof 5.1:* In each trading process, the consumer will upload a unique digital fingerprint which is bound to this trading. Besides, the storage path of the watermarked data for each trading is stored persistently on the blockchain. Thence, realizing the binding of the data and the fingerprint, the theorem holds.

*Definition 2:* **Conspiracy problem**. On the one hand, an third-party CA may collude with a dishonest seller to counterfeit piracy to frame honest users; on the other hand, it may also collude with the illegal consumer to remove fingerprints from fingerprinted data, frustrating the tracking of piracy.

*Theorem 2:* The BDFP has tackled the Conspiracy problem.

*Proof 5.2:* We replace the traditional third-party CA with a blockchain platform. The decentralized trusted platform makes the conspiracy problem no longer exist.

*Definition 3:* **Anonymity**. In electronic trading, each user usually does not want their identity to be revealed.

*Theorem 3:* Anonymity is an inherent property of BDFP.

*Proof 5.3:* The anonymity is essential attribute of blockchain technology. The EOA cannot be linked to entities in the real world.

### C. The Truthfulness and Individual Rationality of SSA

*Definition 4:* **Truthfulness**. We denote the truthful bid and fake bid as $b_t$ and $b_f$ respectively. The corresponding payment is denoted as $p(b_t)$ and $p(b_f)$. We can say that SSA is truthful if the following inequality is satisfied:

$$p_i(b_t) - c_i \geq p_i(b_f) - c_i$$

*Definition 5:* **Individual Rationality**. We say that SSA satisfies the individual rationality if the user's profit is not negative.

$$p_i(b_t) - c_i \geq 0$$

*Lemma 1:* The winner selection is bid-monotonic.

*Proof 5.4:* Assume that $w_i$ wins in the $j_{th}$ iteration of winner selection with his bid $b_i$, i.e., $\frac{E_i(W)}{b_i}$ is the maximal in this iteration. If $w_i$ reports a new bid $b_i'$ that $b_i' < b_i$, he will still win in or before the $j_{th}$ iteration because $\frac{E_i(W)}{b_i'} > \frac{E_i(W)}{b_i}$. Thereby, Lemma 1 holds.

*Lemma 2:* The payments for all winners are critical.

*Proof 5.5:* Assume that $w_i$ wins in the $h_{th}$ iteration, so the winner selection sets $W$ and $W^*$ are the same from the $0_{th}$ to the $(h-1)_{th}$ iteration. We assume that $w_i$ reports a bid $b_i^{'}$ instead of $b_i$. To prove the criticality of the payment $p_i$, we need to prove that $w_i$ will win if $b_i^{'} \leq p_i$ and will be eliminated once $b_i^{'} > p_i$. Then, we consider these two cases in the $h_{th}$ iteration:

Case 1:$b_i^{'} > p_i$. According to the Procedure 3, we can derive that: $\frac{E_i(W_{h-1})}{b_i^{'}} = \frac{E_i(W_{h-1}^*)}{b_i^{'}} < \frac{E_i(W_{h-1}^*)}{p_i} \leq \frac{E_h(W_{h-1}^*)}{b_h}$, where the $w_h$ is a winner, so that $E(W_h^*) = E(W_{h-1}^*) \cup \{w_h\}$. The first equation holds because $W_{h-1} = W_{h-1}^*$. And the last inequation makes sense due to $p_i \geq \frac{b_h E_i(W_{h-1}^*)}{E_h(W_{h-1}^*)}$ according to Eq. (7). Hence, $w_h$ is selected as the winner instead of $w_i$ in the $h_{th}$ iteration. Moreover, $E(W_h) = E(W_{h-1}) \cup \{w_h\} = E(W_h^*)$. Based on the above analysis, we can see that $w_i$ will fail in all iterations of winner selection.

Case 2:$b_i^{'} \leq p_i$. Assume that the winner selection runs over $S_{\_i}$ which is the process for determining payment of $w_i$. According to Eq. (7), we assume that $p_i = \frac{b_h E_i(W_{h^{'}-1})}{E_h(W_{h^{'}-1})}$, where $w_h$ is the winner in the $h_{th}^{'}$ iteration. Now, we run the winner selection again with the input set $S$. We discuss two subcases of this new process: 1) $w_i$ wins before the $h_{th}^{'}$ iteration; 2) $w_i$ does not win before the $h_{th}^{'}$ iteration. In the $h_{th}^{'}$ iteration: $\frac{E_i(W_{h^{'}-1})}{b_i} \geq \frac{E_i(W_{h^{'}-1})}{p_i} \geq \frac{E_h(W_{h^{'}-1})}{b_h}$. Therefore, $w_i$ wins in this iteration.

In conclusion, the payments for all winning bids are critical.

*Theorem 4:* The SSA is truthful.

*Proof 5.6:* According to Myerson's theorem [16], SSA is truthful since the winner selection rule is monotone (i.e., Lemma 1) and each winner is paid with a critical value (i.e., Lemma 2).

TABLE I
SEMANTIC SIMILARITY TEST RESULTS

| Test text | Semantic similarity |
|---|---|
| A ten-minute surveillance video | 24.73 |
| An annotated emotional corpus | 35.72 |
| An English emotional corpus | 35.83 |
| Ten Chinese emotional corpus | 46.90 |

*Theorem 5:* SSA has the property of individual rationality.

*Proof 5.7:* There are two cases here. Case 1: $w_i$ is not chosen as a winner. Then, his cost and payment are all zero (i.e., $c_i = p_i = 0$). Case 2: If $w_i$ wins, he will be paid a critical value $p_i$ which $p_i \geq b_i$ holds (i.e., Lemma 2). In addition, we can get $b_i = c_i$, so $p_i \geq c_i$. Hence, $p_i - c_i \geq 0$, indicating that the payoff of $w_i$ is always non-negative. The theorem holds.

## VI. EXPERIMENT

### A. Training and Testing of Feature Vector Generator

We adopt the Doc2vec as feature vector generator. We use Gensim, an open source library in Python which contains Doc2Vec, to build and train a feature vector generator. The key training parameters of the Doc2Vec model we set are: *vector_size=100,window=5*. The *vector_size* represents the dimension of the feature vector (i.e., $F_c$ and $f_i$). The *window* is the maximum distance between the predicted word and context words used for prediction within a document. Moreover, we use the AG's corpus of news articles as the train set. It is an effort by ComeToMyHead for more than one year, collecting more than 1 million news articles from more than 2,000 different news sources. The total training sample has 120,000 articles, including four classes: world, sports, business, science and technology. Moreover, we test the pre-trained feature vector generator. For example, the target text is: *Ten Chinese emotional texts*. The test results are shown in the Table I. It can be seen that the experimental results are in line with human cognition. In the subsequent auction experiments, we will use the pre-trained feature vector generator to conduct feature vectors for each user.

### B. Evaluation of Smart Contract

First we use Python and the aforementioned pre-trained feature vector generator to simulate users (i.e., the consumer and sellers). Besides, we implement the smart contract by utilizing the Solidity programming language and write a JavaScript test file to evaluate the cost of each function (i.e., Time cost, Gas cost). We choose the most popular development framework Truffle to simulate the Ethereum network to test smart contract. We mainly implement the following 11 functions in smart contract: *publishRequest, participationRequest, computeSimilarity, setBids, winnerSelection, paymentDetermination, prepay, uploadDataPath, downloadDataPath, calculateRefund, payment*, i.e., functions 1-11. In the experiment, we fix the number of sellers participating in the auction to 10. Each user's feature vector is set to 100 dimensions and seller's bid ranges from 1 to 10. The QoD threshold $M_{thre}$ is fixed at 80.

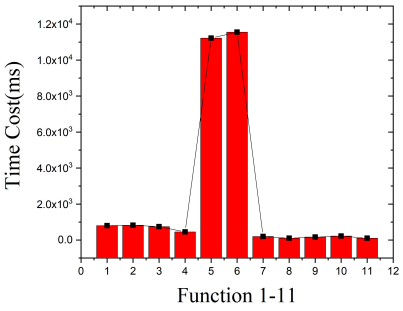


Fig. 4. Time cost of functions

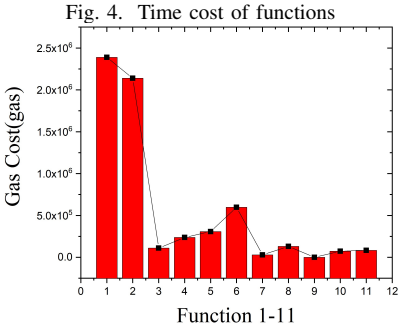


Fig. 5. Gas cost of functions

In Fig. 4, the main time overhead is spent on the two functions of *winnerSelection* and *paymentDetermination*. This is reasonable because these two functions involve multiple rounds of iterations, and each round of iteration involves a lot of complex operations. Other functions are mostly data read/write and simple calculation, with little time overhead. But another factor in the cost of gas is the type of operation. The gas cost of storing data in Ethereum is very expensive. Since the *publishRequest* and *participationRequest* will store a lot of data (especially the 100-dimensional feature vector) to Ethereum, the results of the first two functions in Fig. 4 and Fig. 5 are opposite. We also evaluate the impact of the number of sellers that participate in the auction on the function overhead. In Fig. 6 and Fig. 7, the time costs and gas costs are roughly linear with the number of sellers in the auction. In other words, when the number of users participating in the auction increases, the costs will not increase sharply.

### C. The Performance of SSA

In order to make the experimental results more obvious and reliable, we fixed the number of sellers here to 50. The other experimental parameters are as follows: $M_{thre}$ = 100, the dimension of the feature vector is 100, and the bid range is (0,10]. It can be seen from Fig. 8 that the users selected as winners are concentrated in the upper left of the graph, which means the greater semantic similarity and smaller bid. This is in line with the expected effect because we always choose seller with the largest $\frac{E_i(W)}{b_i}$ as the winner. Fig. 9 shows a comparison of the payments received by the winners with their real costs. Each point is above the line $y = x$ which means

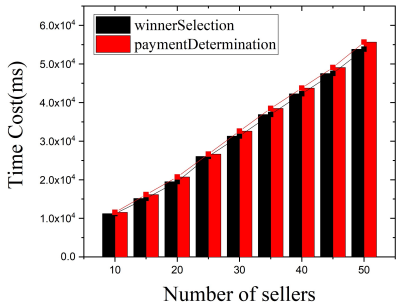


Fig. 6. Time cost of different number of bidders

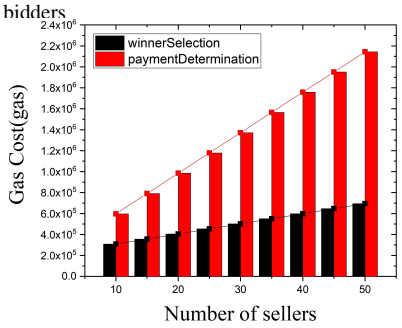


Fig. 7. Gas cost of different number of bidders


Fig. 8. Winner selection

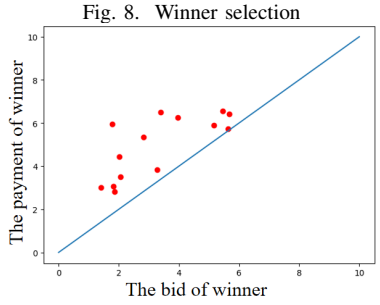


Fig. 9. Payment determination

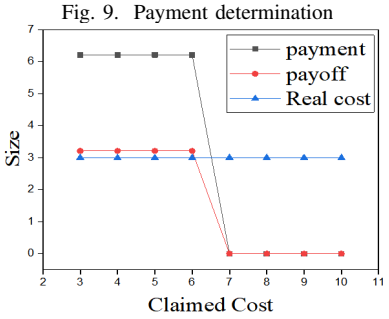


Fig. 10. Payoff of a bid

that each winner will receive payment no less than their real cost, so that individual rationality is guaranteed.

To verify the truthfulness, we randomly pick a winning bid, change his claimed cost, and recalculate the related payments as well as the payoffs. The results are illustrated in Fig. 10. The payment is 6.21, and the real cost is 3. Then, the payoff is 3.21. We find that the payoff remains unchanged if the claimed cost is no more than the payment. Moreover, when the claimed cost is larger than the critical value 6.21, the payoff becomes zero. Furthermore, we select the First-Price Sealed Bid Auction (FPSB) as the baseline algorithm and compare the Sum of the Similarities of All Winners (SSAW, i.e., the QoD.) that are achieved by the two auction algorithms under different numbers of winners and different budget limits. The results in Fig.11 and Fig. 12 show that the SSA algorithm always performs better than the FPSB algorithm for a given number of winners or budget limit.

## VII. Related Work

**Blockchain-based Trading:** Recently, owning to the transparency, fairness, and security of blockchain, a few works have proposed decentralized trading system based on blockchain. For example, [17] implements a decentralized energy trading system using blockchain to address the problem of providing trading security. In [18], they propose a fault-tolerant incentivisation mechanism (FTIM) which is proven to accommodate the requirements of two important application scenarios by achieving mechanism properties. Besides, they also design a MPCSToken smart contract to facilitate its service auction, task execution and payment settlement process. To avoid the high storage cost on blockchain, [19] proposes a framework that allows sellers and consumers to interact through smart contracts, complete transactions, and transfer data storage off the chain, alleviating the problem of high storage costs.

**Auction Mechanism:** As well known, incentive mechanism is often utilized to motivate users to participate in trading honestly. For instance, to measure the long-term quality of workers, a long-term dynamic quality-aware incentive mechanism for crowdsourcing, named MELODY, which models the interaction between requesters and workers as a continuously running reverse auction is proposed in [20]. To solve a non-trivial set cover problem in vehicle-based, nondeterministic crowdsensing system, [21] proposes a truthful, reverse auction-based incentive mechanism for selecting the winning bidder with almost minimal social costs and determining the payment of all participants.

**Copyright Protection:** As for copyright protection issues, in [22], a uniqueness index is defined and proposed, which is a new rigorous measurement of the data uniqueness. However, as long as the data has been modified, it can escape their detection. As well known that digital watermarking is suitable to solve this problem. [14] proposes a novel watermarking protocol which integrates a buyer's watermark, trading iden-

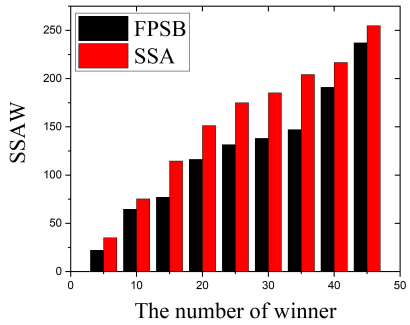


Fig. 11. SSAW under different number of winners

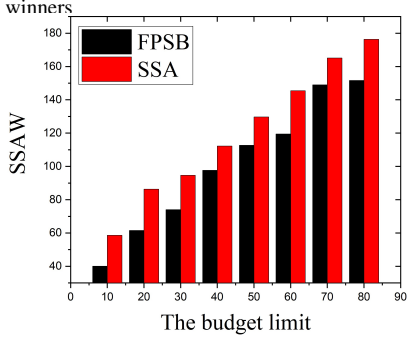


Fig. 12. SSAW under different Budget limit

tification number, and a buyer's one-time public/private key to protect the ownership of digital content. To protect privacy, [23] designs a protocol which ensures that the buyer pays the correct price without revealing the item purchased, and that the seller can identify the buyer who issued the pirated copy.

## VIII. Conclusion

In this paper, we propose a Copyright-Preserving crowdsourcing data trading framework based on blockchain. We implement a smart contract that enables sellers and the consumer to conduct credible and truthful trading while ensuring the quality and copyright of data. We introduce the concept of semantic similarity to measure the degree of requirement matching which assures the quality of data. Taking the truthfulness and individual rationality into account, we propose a semantic-similarity-based suction algorithm to impel more users to participate in data trading honestly. Besides, we design a blockchain-based digital fingerprint protocol to address copyright protection issue without third-party certification authority. Finally, we implement a prototype on the Ethereum test network and the evaluations demonstrate its feasibility.

## IX. Acknowledgments

This research was supported by the National Key Research and Development Program of China (Grant No. 2017YFB0802302), the National Natural Science Foundation of China (NSFC) (Grant No. 61872330, 61572457, 61572336, 61379132, 61936015, U1709217), the NSF of Jiangsu Province in China (Grant No. BK20191194, BK20131174, BK2009150), Natural Science Research Project of Jiangsu Higher Education Institution (No.18KJA520010), and Anhui Initiative in Quantum Information Technologies (Grant No. AHY150300).

## References

[1] F. Liang, W. Yu, D. An, Q. Yang, X. Fu, and W. Zhao. A survey on big data market: Pricing, trading and protection. *IEEE Access*, 6:15132–15154, 2018.

[2] Jeff Howe. The rise of crowdsourcing. *Wired magazine*, 14(6):1–4, 2006.

[3] M. Xiao, J. Wu, L. Huang, R. Cheng, and Y. Wang. Online task assignment for crowdsensing in predictable mobile social networks. *IEEE Transactions on Mobile Computing*, 16(8):2306–2320, 2017.

[4] C. J. Zhang, L. Chen, H. V. Jagadish, M. Zhang, and Y. Tong. Reducing uncertainty of schema matching via crowdsourcing with accuracy rates. *IEEE Transactions on Knowledge and Data Engineering*, 32(1):135–151, 2020.

[5] Q. Liu, Y. Tian, J. Wu, T. Peng, and G. Wang. Enabling verifiable and dynamic ranked search over outsourced data. *IEEE Transactions on Services Computing*, pages 1–1, 2019.

[6] L. Zhang, Y. Li, X. Xiao, X. Li, J. Wang, A. Zhou, and Q. Li. Crowdbuy: Privacy-friendly image dataset purchasing via crowdsourcing. In *IEEE INFOCOM 2018*, pages 2735–2743, 2018.

[7] C. Chai, J. Fan, and G. Li. Incentive-based entity collection using crowdsourcing. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 341–352, 2018.

[8] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. [online] Available: https://bitcoin.org/bitcoin.pdf, 2008.

[9] Buterin V. *A Next-Generation Smart Contract and Decentralized Application Platform.White paper*. 2014.

[10] Baoyi An, Mingjun Xiao, An Liu, Guoju Gao, and Hui Zhao. Truthful crowdsensed data trading based on reverse auction and blockchain. In *Database Systems for Advanced Applications*, pages 292–309, 2019.

[11] W. Dai, C. Dai, K. R. Choo, C. Cui, D. Zou, and H. Jin. Sdte: A secure blockchain-based data trading ecosystem. *IEEE Transactions on Information Forensics and Security*, 15:725–737, 2020.

[12] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *ArXiv*, abs/1405.4053, 2014.

[13] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[14] Chin-Ling Chen, Chin-Chang Chen, De-Kui Li, and Po-Yueh Chen. A verifiable and secret buyer–seller watermarking protocol. *IETE Technical Review*, 32(2):104–113, 2015.

[15] Abid Khan, Farhana Jabeen, Farah Naz, Sabah Suhail, Mansoor Ahmed, and Sarfraz Nawaz. Buyer seller watermarking protocols issues and challenges – a survey. *Journal of Network and Computer Applications*, 75:317 – 334, 2016.

[16] Roger B. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6(1):58–73, 1981.

[17] N. Z. Aitzhan and D. Svetinovic. Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams. *IEEE Transactions on Dependable and Secure Computing*, 15(5):840–852, 2018.

[18] F. Shi, Z. Qin, D. Wu, and J. McCann. Mpcstoken:smart contract enabled fault-tolerant incentivisation for mobile p2p crowd services. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 961–971, 2018.

[19] M. Li, J. Weng, A. Yang, W. Lu, Y. Zhang, L. Hou, J. Liu, Y. Xiang, and R. Deng. Crowdbc: A blockchain-based decentralized framework for crowdsourcing. *IEEE Transactions on Parallel and Distributed Systems*, pages 1–1, 2018.

[20] H. Wang, S. Guo, J. Cao, and M. Guo. Melody: A long-term dynamic quality-aware incentive mechanism for crowdsourcing. *IEEE Transactions on Parallel and Distributed Systems*, 29(4):901–914, 2018.

[21] G. Gao, M. Xiao, J. Wu, L. Huang, and C. Hu. Truthful incentive mechanism for nondeterministic crowdsensing with vehicles. *IEEE Transactions on Mobile Computing*, 17(12):2982–2997, 2018.

[22] T. Jung, X. Li, W. Huang, Z. Qiao, J. Qian, L. Chen, J. Han, and J. Hou. Accounttrade: Accountability against dishonest big data buyers and sellers. *IEEE Transactions on Information Forensics and Security*, 14(1):223–234, 2019.

[23] A. Rial, J. Balasch, and B. Preneel. A privacy-preserving buyer–seller watermarking protocol based on priced oblivious transfer. *IEEE Transactions on Information Forensics and Security*, 6(1):202–212, 2011.