A Valid Blockchain-based Data Trading Ecosystem

Taotao Li, Dequan Li

No Institute Given

Abstract. Data, an important asset in digital economy, has fueled the emergence of a new data trading market. Big data market can efficiently promote data trading and further increases the utility of data. However, to realize effective data trading, several challenges needs to be resolved. First, it needs to resolve disputes over data availability in the data trading. Second, atomic exchange and payment fairness between the seller and the buyer are hard to guarantee. Third, data trading platform is the single-point-failure. In this paper, we resolve these challenges by presenting a valid blockchain-based data trading ecosystem. The ecosystem constructs a decentralized arbitration mechanism to address the dispute over data availability in data trading. The ecosystem also designs a sale contract and a deterministic public-key encryption algorithm to guarantee fairness of data trading between the seller and buyer. The features of blockchain is preventing single-point-failure of data trading platform. We prove the desirable security properties that a secure data trading ecosystem should have. Discussion of the presented ecosystem is given. To demonstrate availability, we implement our proposed data trading ecosystem using smart contract in Solidity and program in Java, and evaluate its performance.

Keywords: data trading, smart contract, decentralized arbitration, blockchain

1 Introduction

With a number of new technologies and applications are integrated into our daily life, such as mobile and social networking applications, and smart-world systems based on the Internet of Things (smart city, smart home, smart grid, smart transportation, and others), vast amount of data will be collected [22, 32, 31, 36, 39, 40, 24]. Some companies and public sectors [10, 28, 25] use such data to make market decisions and improve their services [26]. In the era of big data, the generated data have become important asset and fueled the emergence of a new data trading in data market. In fact, several data trading markets have already emerged recently, such as the DataExchange and Datacoup [18, 20].

A simple traditional data trading ecosystem is shown in Fig. 1. There are three entities, namely: seller, buyer, data exchange platform (a trusted middle-party). The seller sends data to be sold to data exchange platform and sets an appropriate selling price. The buyer browses datas of interest and pays the money

to the data exchange platform. On receiving the money, the data exchange platform transfers data purchased to the buyer and pays partial money to the seller (deducting transaction fees and management fees). When the buyer believes that data he/she purchased is invalid, the data exchange platform resolves the dispute relying on a centralized mechanism or a trusted third entity.

To realize effective data exchange in such a centralized exchange model [43, 14, 9, 21], several challenges needs to solved, as explained below:

1) How the platform uses a decentralized mechanism to resolve the dispute over data availability in the data trading. And the dispute means that the buyer believes that data he/she purchased is invalid.
2) Atomic exchange and payment fairness between the seller and the buyer are hard to guarantee, due to the opaque nature of data exchange platform.
3) The centralized data exchange platform is the single-point-failure. When the data exchange platform crash and/or is attacked, it will halt.

Fig. 1. Simplified conventional data trading and centralized data process

Related work. Zhao et al. [41] proposed a fair data trading protocol based on blockchain, which integrates similarity learning, ring signatures and double-authentication-preventing signatures to guarantee the availability of data trading and fairness between the seller and the buyer. Although the protocol introduces a market manager to resolve the dispute over the data availability in the data trading, the market manager in the protocol is centralized. Xiong et al. [37] presented a smart contract based data trading mode by combining the machine learning. The mode uses smart contract to guarantee the fairness of data exchange, and utilizes arbitration institution to deal with the dispute over the data availability in the data trading. However, the arbitration institution in the mode is a trusted entity of trading parties. Dai et al [12] presented a secure data trading ecosystem based on blockchain by combining the Intels Software Guard Extensions (SGX). Although the ecosystem can protect the data processing, the source of data and the analysis results, it depends on trusted SGX-based execution environment. Delgado-Segura et al [13] presented fair protocol for data trading based on Bitcoin transactions. The protocol constructs a new type of Bitcoin transaction by using script language, but the operation code OP_AND they used is invalid in practical Bitcoin network.

Chen et al. [9] presented a blockchain-based ecosystem for big data market. The ecosystem uses blockchain technology to record transaction logs and other important documents. Zhou et al [42] proposed a distributed blockchain-based data vending framework, which integrates the data embedding and similarity learning to tackle the dilemma between the effective of data retrieval and leakage risk from indexing the data. However, they mechanisms are insufficient for solving the dispute and availability for fair data exchange based on blockchain.

Yang et al [38] presented a design of data trading platform based on blockchain by combining cryptography technology, which ensures that the platform does not have the opportunity to peek, copy and store trading data. Wang et al. [34] proposed a distributed data management based on blockchain. The data management mode guarantees that datas in the process of data trading is not retained and copied by the third-party. However, these schemes are not implemented in a real blockchain environment.

Our contributions. In this paper, we present a valid blockchain-based data trading ecosystem to complement the existing data trading ecosystem. A decentralized arbitration mechanism consisting of an arbitration contract, a producing contract and a deterministic digital signature algorithm resolves the dispute over data availability in data trading. The system designs the sale contract, advanced encryption standard algorithm and deterministic public-key encryption algorithm to guarantee fairness of data trading between the seller and buyer. The features of blockchain is preventing single-point-failure of data trading platform. We formally prove the security of our system and discuss the arbitration process. We implement our system using smart contract in Solidity and program in Java, and demonstrates the validity of our system. The full code of smart contract and API interface have been released to the GitHub (https://github.com/01007467319/TTL-ECO-SC.git).

1.1 Organization

The rest of this paper is organized as follows. In Section 2, we introduce the preliminaries of our system. We present mode and security requirements of data trading in Section 3. We describe overview and module design of data trading protocol in Section 4, and the security analysis and discussion are given in Section 5 and 6. In Section 7, we implement a valid data trading system, and evaluate its execution efficiency and cost. We conclude this paper in the last section.

2 Preliminaries

2.1 Smart contract

In the 1990s, cryptographer Nick Szabo [33] proposed the term and defined it as "a set of promises, specified in digital form, including protocols within which the parties perform on the other promises." Since then, The concept of smart contracts has changed, especially after the introduction of a decentralized blockchain

platform, which is a "autonomous agent" stored in the blockchain, encoded as part of the "create" transaction, which introduces a contract blockchain. Once successfully created, smart contracts are identified by contract addresses; each contract has a certain number of virtual coins (such as ether) and private storage associated with its predefined executable code. The contract state consists of two main parts: the private store and the number of virtual coins it holds. Contract code can manipulate variables like traditional imperative programs. For example, the Ethereum contract code is a stack-based, low-level bytecode language called the Ethereum Virtual Machine (EVM) code. The player defines the contract using a high-level programming language, such as Solidity [15] (a JavaScript-like language), and then compiles it into EVM code. In order to invoke the contract, players send the transaction to the contract address. Transactions typically include: payment for the execution of the contract or input data for the invocation.

2.2 Advanced encryption standard

In January 1997, the US National Institute of Standards and Technology (NIST) issued a call for an Advanced Encryption Standard (AES) to replace the current Data Encryption Standard (DES). In October 2000, NIST [1] announced that the winning algorithm was Rijndael [11] (a block cipher designed by the Belgian cryptographers Vincent Rijmen and Joan Daemen based on SPN structure). Today, AES is widely used and no significant security weaknesses have been discovered. We apply the practical AES [11] to implement data trading. There are three algorithms in AES: Key generate (AES.Gen), Encrypt (AES.Enc), and Decrypt (AES.Dec). The details are given below:

- **AES.Gen.** The key-generation algorithm **AES.Gen** takes as input the security parameter 1^n and outputs a key k; we write this as k $\leftarrow$ **AES.Gen**(1^n) (emphasizing that **AES.Gen** is a randomized algorithm). We assume that any key k $\leftarrow$ **AES.Gen**(1^n) satisfies $|k| \geq$ n.
- **AES.Enc.** The encryption algorithm **AES.Enc** takes as input a key k and a plaintext message m $\in \{0,1\}^\star$ and outputs a ciphertext c; we write this as c $\leftarrow AES.Enc_k(m)$.
- **AES.Dec.** The decryption algorithm **AES.Dec** takes as input a key k and a ciphertext c, and outputs a message m or $\perp$. We write this as m $:= AES.Dec_k(c)$ (assuming here that **Dec** does not return an $\perp$).

It is required that for every n, every key k $\leftarrow$ **AES.Gen**(1^n), and every m $\in \{0,1\}^\star$, it holds that $AES.Dec_k(AES.Enc_k(m)) = m$.

2.3 Deterministic Public-Key encryption scheme

The notion of deterministic public-key encryption (DPKE) was introduced by Bellare et al. [2]. These are public-key encryption schemes where the encryption

algorithm is deterministic. When using a deterministic encryption algorithm, however, the full-fledged notion of semantic security [17] is out of reach. Subsequent research results [4, 7, 3, 8, 16, 27, 35] has successfully shown that a natural variant of the notion of semantic security can be guaranteed even when using a deterministic encryption algorithm, as long as plaintexts are independent of the public key used by the scheme. Raghunathan et al [30] extends the previously proposed notions of security, allowing adversaries to adaptively choose plaintext distributions after seeing the public key, in an interactive manner.

In this section, we briefly introduce the algorithms of deterministic public-key encryption that are used in the protocol we proposed below. A deterministic public-key encryption scheme is a tuple of polynomial-time algorithms (**DPKE.Gen, DPKE.Enc, DPKE.Dec**) such that:

- **DPKE.Gen.** The key-generation algorithm **DPKE.Gen** is a randomized algorithm that takes as input the security parameter 1^λ and outputs a pair of keys (sk, pk); we write this as $(sk, pk) \leftarrow$ **DPKE.Gen**(1^λ) with the pk is the public key and sk is the private key.
- **DPKE.Enc.** The encryption algorithm **DPKE.Enc** is a deterministic algorithm that takes as input a public key pk and a message m $\in \{0,1\}^{n(\lambda)}$ with n = ploy(λ), and outputs a ciphertext c. We write this as c $\leftarrow DPKE.Enc_{pk}(m)$.
- **DPKE.Dec.** The decryption algorithm **DPKE.Dec** is a possibly randomized algorithm that takes as input a private key sk and a ciphertext c, and outputs a message m $\in \{0,1\}^{n(\lambda)}$ or a special symbol $\perp$ denoting failure. We write this as m $:= DPKE.Dec_{sk}(c)$.

It is required that, except possibly with negligible probability over $(sk, pk) \leftarrow$ **DPKE.Gen**(1^λ), we have $DPKE.Dec_{sk}(DPKE.Enc_{pk}(m)) = m$ for any (legal) message m.

2.4 Deterministic digital signature algorithm scheme

Deterministic digital signature algorithm (DDSA) was first described by Bernstein et al. [5]. The characteristic of DDSA is that there is no need for generating fresh random numbers per signature. Edwards curve Digital Signature Algorithm (EdDSA) [5] follows a similar approach and uses the hash of the private key and the message M as a nonce. Thus, any change of M results in a new nonce. Some deterministic variant [29, 6, 23] of DSA was provided. We apply the DDSA to support the algorithm **Producing** (see section 4.2.4 for details) in the data trading protocol and adopt the practical instantiation of DDSA scheme [29]. A deterministic digital signature algorithm scheme is a tuple of polynomial-time algorithms (**DDSA.Gen, DDSA.Sign, DDSA.Ver**) such that:

- **DDSA.Gen.** The key-generation algorithm **DDSA.Gen** takes as input the security parameter 1^n and outputs a pair of keys (sk, pk); we write this as $(sk, pk) \leftarrow$ **DDSA.Gen**(1^n) with the pk is the public key and sk is the private key. We assume for convenience that pk and sk each has length at least n, and that n can be determined from pk, sk.

- **DDSA.Sign.** The signature algorithm **DDSA.Sign** takes as input a private key sk and a message m from some message space (that may depend on pk), and outputs a signature σ. We write this as $\sigma \leftarrow DDSA.Sign_{sk}(m)$.
- **DDSA.Ver.** The deterministic verification algorithm **DDSA.Ver** takes as input a public key pk, a message m and a signature σ, and outputs a bit b, with b = 1 denoting valid and b = 0 denoting invalid. We write this as b $:= DDSA.Ver_{pk}(m, \sigma)$.

It is required that, except with negligible probability over $(pk, sk) \leftarrow$ **DDSA.Gen**(1^n), it holds that $DDSA.Ver_{pk}(m, DDSA.Sign_{sk}(m)) = 1$ for every (legal) message m.

3 Blockchain-based data trading model and security requirements

In the section, we introduce the blockchain-based data trading model and relevant security requirements.

3.1 Blockchain-based data trading model

There are three entities, namely: seller, buyer and blockchain data exchange platform - see also Fig 2. The seller is the data provider, and profits from selling the data. The buyer is the consumer of the data and generates payment transaction in the platform to purchase the data. The platform is secure blockchain for contract deployment, contract invoking and contract execution; and consists of nodes and arbitrators (a.k.a, miners). The data exchange platform provides two services for the buyer and the seller. The former is a fair data trading service that supports the buyer sales data and the buyer purchases data. The latter is a decentralized arbitration service that provides arbitration for the buyer and the seller. Especially, when the buyer believes that the purchased data does not consistent with the description of the raw data, the buyer can apply for arbitration and gains a fair result.

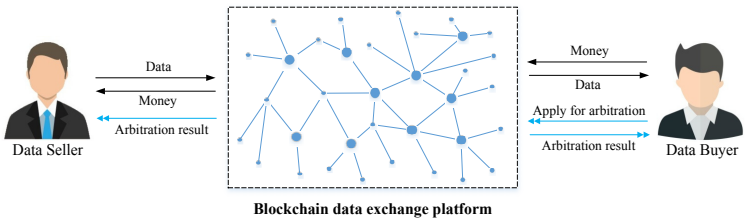

Fig. 2. Simplified fair data trading and decentralized arbitration process

Definition 1 (Blockchain-based data trading protocol). *The data trading protocol is a tuple of polynomial time algorithms (**Segmenting, Sale, Arbitration, Producing**) such that:*

- ***Segmenting.*** *The algorithm **Segmenting** takes as input the data and the key, and outputs ciphertext c; we write this as* $c \leftarrow$ ***Segmenting***$(data, key)$. *Parse c into* $(c_1, c_2, ..., c_n)$.
- ***Sale.*** *The algorithm **Sale** takes as input the seller's deposit* $deposit_{seller}$, *the buyer's deposit* $deposit_{buyer}$, *the public key pk of the buyer, the index i of the data segment and the money* $money_i$ *of the data segment, and outputs a result. We write this as* $result \leftarrow Sale(deposit_{seller}, deposit_{buyer}, pk, i, money_i)$.
- ***Arbitration.*** *The algorithm **Arbitration** takes as input a data ciphertext* c_i, *data key* k_i, *public key pk, a ciphertext* w_i *and the description* d_{des_i} *of the i-th data segment, and outputs a bit* $b \in \{0, 1\}$. *We write this as* $b \leftarrow Arbitration(c_i, k_i, pk, w_i, d_{des_i})$.
- ***Producing.*** *The algorithm **Producing** takes as input a block header* $block_{header}$, *an integer k, and outputs a committee A consisting of k arbitrator. We write this as* $A \leftarrow Producing(block_{header}, k)$. *Parse A into* $(A_1, A_2, ..., A_k)$.

3.2 Security requirements

Following security requirements of Zhao et al. [41] a secure blockchain-based data trading protocol should satisfy the following requirements:

- **Completeness.** The completeness means that, if both the sellers and the buyers are honest at all epochs, then an honest seller can receive the money and an honest buyer can gain the valid data.
- **Confidentiality.** The confidentiality says that a malicious buyer without the money is unable to gain the data.
- **Availability.** If a seller's behavior is honest, then the buyer can gain valid data. If a malicious seller sells fake data, he will be punished in the sense that he loses his deposit.
- **Fairness.** The fairness means that, at the end of the protocol, either the seller obtains the money and the buyer gets valid data or neither the seller and the buyer obtain nothing.

In addition, we define a decentralization requirement for the data trading protocol.

- **Decentralization.** The decentralization says that, when the buyer believes that data he/she purchased is invalid, the protocol can solve the dispute by decentralizing arbitration contract.

4 Data trading protocol

In this section, we first give an overview of our protocol and then describe the module design of our protocol in detail.

4.1 Overview

A typical workflow in our protocol is shown in Fig. 2. For the convenience of description, we especially extract the Sale contract (SC), Arbitration contract (AC) and Producing contract (PC) from the blockchain.

- Step(1)-(4). The seller posts the information of data to be sold to the blockchain and deploys the SC. The buyer browses the data information and submits purchasing data information to the SC.
- Step(5)-(7). The seller sends encrypted data key to the buyer. The buyer then gets the data by decrypting encrypted datas with its private key and data key obtained by decrypting.
- Step(8)(9). The buyer verifies the validity of the data. If the data is valid, the buyer submits application of transaction finish. Otherwise the buyer applies for arbitration.
- Step(10)-(14). The decentralizing AC gets arbitration information from the blockchain such as the SC, and randomly selects k arbitrator by calling the PC.
- Step(15)(16). The decentralizing AC produces an arbitration result and sends it to the SC. The SC then executes arbitration result returned by the AC.

In the protocol, the buyer has his/her DPKE key pair (sk, pk) and the seller has his/her AES key k.

4.2 Module design

In this section, we present an elaborate module design of the data trading protocol. These interacting modules are Data segmenting, Sale contract, Arbitration contract and Producing contract. A summary of notations used in our protocol is shown in Tables 1.

4.2.1 Data segmenting

The process of data segmenting is as shown in Algorithm 1. First, the seller divides the data into multiple segments of data $d_i (i \in [1, n])$; then, the seller encrypts each data segment separately with different data keys $k_i (i \in [1, n])$, and obtains the ciphertexts $c_i = AES.Enc_{k_i}(d_i)(i \in [1, n])$. Finally, the seller stores the ciphertexts c_i, the description d_{des_i} and the price for the data segments in the blockchain.

The segmentation of the above data satisfies the following principles: (i) each data segment is required to allow the arbitrator to identify the authenticity of the data segment, and to ensure that the Arbitration contract can make a favorable result for the seller on the premise that the seller is honest; (ii) in order to reduce the information leakage of the data segments during the arbitration process, the information that needs to be exposed to the arbitrator should be as small as possible.

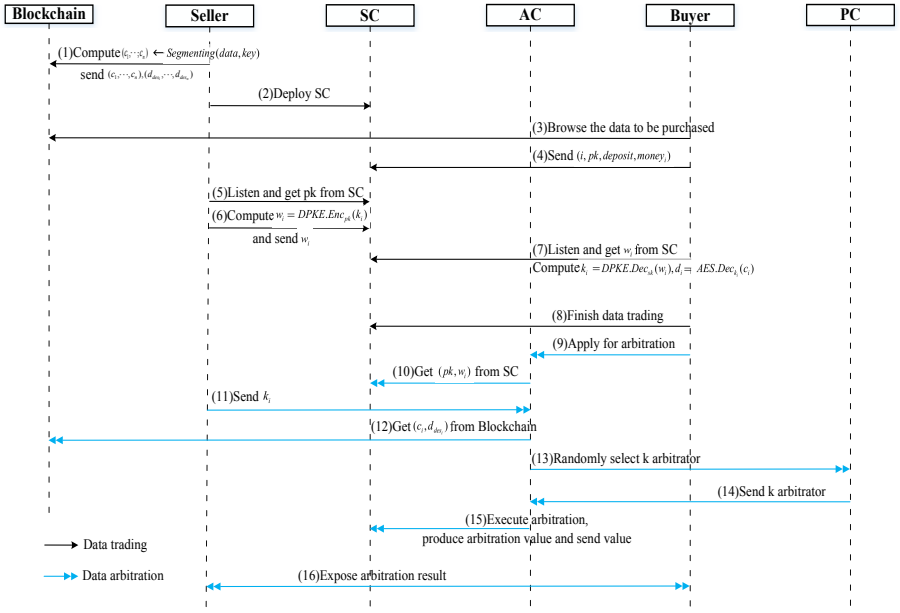

Fig. 3. The overview of the proposed protocol

Table 1. Summary of notations

Notation	Explanation
D_{role}	the address of role
$deposit_{role}$	the deposit of role
τ	deposit threshold
d_{des_i}	the description of the i-th data segment
$money_i$	money for purchasing the i-th data segment
$value_{A_i}$	the voting result of the i-th arbitrator
$block_{header}$	the header of the block
$chain_{height}$	the height of the blockchain

Algorithm 1 Segmenting

```
function SEGMENT(data, key)
    (d_1, d_2, ..., d_n) ← data
    (k_1, k_2, ..., k_n) ← key
    for i ≤ n do
        c_i = AES.Enc_{k_i}(d_i)
    end for
    return (c_1, c_2, ..., c_n)
end function
```

4.2.2 Sale contract

The main task of a sale contract is to execute the data trading. The process of the sales contract is shown in Algorithm 2. In the practical environment, the buyer can browse the information of data segments, determine the data to be purchased, and submit the deposit $deposit_{buyer}$, the money $money_i$, the public key pk and the index i of the data segment to be purchased to the algorithm 2.

The buyer can determine whether the seller deposit $deposit_{seller}$ is sufficient. If the seller does not submit the deposit, or the amount of the deposit is insufficient, Algorithm 2 will automatically refund the buyer's deposit and money, and the transaction will be terminated. If the seller's deposit is sufficient, the seller uses the buyer's public key pk to encrypt the data key k_i corresponding to the data segment purchased by the buyer; the seller obtains ciphertext $w_i = DPKE.Enc_{pk}(k_i)$ and sends it the Sale contract.

Upon obtaining the ciphertext w_i from the Sale contract, the buyer uses the private key sk to decrypt ciphertext w_i and obtains data key $k_i = DPKE.Dec_{sk}(w_i)$. The buyer then decrypts the ciphertext c_i using the data key k_i to obtain the data segment $d_i = AES.Dec_{k_i}(c_i)$. Finally, the buyer determines whether the data segment d_i is valid (meaning whether a valid data key k_i is received by the buyer and whether the data segment is consistent with its the description, etc.). If d_i is valid, Algorithm 2 transfers the $money_i$ to the seller; the deposits is refunded and the transaction ends. Otherwise, the buyer applies for arbitration.

4.2.3 Arbitration contract

The main task of the Arbitration contract is to perform data arbitration. Before executing arbitration, it is necessary to determine the arbitration committee (as described in Section 4.2.4). When the buyer believes that the purchased data d_i is invalid, the buyer submits the arbitration application to the Arbitration contract within the specified time. The process of the Arbitration contract is shown in Algorithm 3.

- Preparing data. The Arbitration contract gets c_i, pk, w_i, d_{des_i} from the blockchain such as the Sale contract. The seller provides the data key k_i to the Arbitration contract.

- Arbitrating. Suppose that a miner accepts an arbitration application and becomes the first arbitrator A_1. The purpose of the A_1 is to arbitrate whether

Algorithm 2 Sale

function SALE($deposit_{seller}, deposit_{buyer}, pk, i, money_i$)
  if $deposit_{seller} \geq \tau$ **then** ▷ if the seller's deposit is enough
    $w_i = DPKE.Enc_{pk}(k_i)$
  else
    $D_{buyer} = money_i$
    $D_{buyer} = deposit_{buyer}$
    exit()
  end if
  $k_i = DPKE.Dec_{sk}(w_i)$
  if $d_i = AES.Dec_{k_i}(c_i)$ **then** ▷ if the k_i received by the buyer is true
    $D_{seller} = money_i$
    $D_{seller} = deposit_{seller}$
    $D_{buyer} = deposit_{buyer}$
  else
    if (invoke ARBITRATION()) = 1 **then**
      $D_{seller} = money_i$
      $D_{seller} = deposit_{seller}$
      $D_{committee} = deposit_{buyer}$
    else
      $D_{buyer} = money_i$
      $D_{buyer} = deposit_{buyer}$
      $D_{committee} = deposit_{seller}$
    end if
  end if
end function

both the seller and the buyer are honest in the data trading process. The arbitration is as follows:

- Is the data segment d_i sold by the seller consistent with the description d_{des_i} of the d_i (a.k.a., $d_{des_i} = AES.Dec_{k_i}(c_i)$)? If they are consistent, the result is "Yes", otherwise the result is "No".
- Did the seller submit a valid data key k_i to the buyer (a.k.a., $w_i = DPKE.Enc_{pk}(k_i)$)? If the k_i is valid, the result is "Yes", otherwise the result is "No".

Only when the results of the above two judgments are "Yes", does the arbitrator vote to support the seller, that is, the arbitrator adds "value" to 1. Otherwise the arbitrator votes to support the buyer and subtracts "value" by 1. If the seller does not provide the information required for arbitration within a certain period of time, the arbitrator subtracts "value" by 1.

- Generating the result of the arbitration. Since the k is an odd number greater than 3, and each arbitrator either adds "value" to 1 or subtracts "value" by 1. Therefore, the "value" must be a positive number (the seller succeeds) or a negative number (the buyer succeeds), and the final value of the "value" serves as the final arbitration result of the arbitration committee. If the seller succeeds in the arbitration, the seller obtains the money and his deposit re-

Algorithm 3 Arbitration

```
1: function ARBITRATION(c_i, pk, w_i, d_des_i, k_i)
2:     value ← 0
3:     while i ≤ k do                          ▷ arbitrated by each arbitrator A_i
4:         A_i ← invoke(PRODUCING())
5:         if w_i = DPKE.Enc_pk(k_i) then
6:             if d_des_i = AES.Dec_k_i(c_i) then
7:                 value = value + 1
8:             else
9:                 value = value − 1
10:            end if
11:        else
12:            value = value − 1
13:        end if
14:        i++
15:    end while
16:    if value > 0 then                       ▷ indicate arbitration result
17:        return 1
18:    else
19:        return 0
20:    end if
21: end function
```

funded by the Sale contract, the buyer's deposit serves as arbitration fees, and vice versa.

4.2.4 Generation of arbitrator

The process of generating an arbitrator is shown in Algorithm 4. The arbitration committee consists of k arbitrators, where k is an odd number greater than 3 (analysis as described in Section 6). In the practical environment, the arbitrator can be volunteered by the miners who generates the valid block. In order to randomly generate each arbitrator, Algorithm 4 takes the newly block header $block_{header}$ (without facing risk of forking) on the current blockchain as a random source [1]. The specific method is as follows: the hash calculation is performed on the block header $block_{header}$, and the hash value h is obtained; the h modulo the height number $chain_{height}$ of the current blockchain (for security, the height number of the stable blockchain is taken), obtaining a certain block; the arbitrators are miners who had mined the block. For the fairness of arbitration, each arbitrator is selected after the previous arbitrator completes the voting. The implementing process is as follows: after each arbitrator finishes the voting, the parameter $block_{header}$ is signed to obtain a signature value σ_i; hashing the σ_i and the $block_{header}$, and producing the $(i+1)$-th arbitrator A_{i+1}.

[1] the hash value of the block header is generally randomly generated. For example, in Ouroboros [19], because the miners are randomly selected, the block headers generated by the miners are also random.

In order to avoid arbitrators cheating, the digital signature algorithm used here is a deterministic digital signature algorithm [29].

It is very important that, since the members of the arbitration committee are randomly selected, this is a decentralized arbitration mechanism consisting of Arbitration contract and deterministic digital signature algorithm. The mechanism facilitates the completion of the transactions and reduces the occurrence of dishonest behavior in the transaction process.

Algorithm 4 Producing

function PRODUCING($block_{header}$, k)
  for i $\leq$ k **do**
    if $i = 1$ **then**
      $h = Hash(block_{header})$
      $A_i \leftarrow h \bmod(chain_{height})$
      $\sigma_i = DDSA.Sign_{sk}^{A_i}(block_{header})$
    else
      $h = Hash(block_{header}||\sigma_{i-1})$
      $A_i \leftarrow h \bmod(chain_{height})$
      $\sigma_i = DDSA.Sign_{sk}^{A_i}(block_{header})$
    end if
  end for
  return $(A_1, A_2, ..., A_k)$
end function

5 Proofs of security

In this section, we prove the security of the blockchain-based data trading protocol.

Lemma 1. *The presented blockchain-based data trading protocol is completeness.*

Proof. If both the buyer and the seller honestly abide by the data trading protocol described in Section 4, then from the completeness of the underlying DPKE and AES scheme, the buyer can always receive the data key k_i and uses it to decrypt the ciphertext c_i. Therefore, the buyer obtains the data segment d_i, and the seller also gets the money $money_i$. □

Lemma 2. *The presented blockchain-based data trading protocol is confidentiality.*

Proof. In our protocol, data segment d_i is encrypted by the seller using data key k_i, the seller will use the public key pk provided by the buyer to encrypt the data key k_i and send $w_i = DPKE.Enc_{pk}(k_i)$ to the buyer only after the

buyer has paid the money for data d_i. If the buyer does not provide the money, the buyer can not obtain the ciphertext w_i. Without the w_i, the buyer can not extract the data key k_i to decrypt the ciphertext c_i. □

Lemma 3. *The presented blockchain-based data trading protocol is availability.*

Proof. If the seller's behavior is honest, the honest buyer can obtain the ciphertext w_i and extract the data key k_i from the w_i to decrypt the ciphertext c_i. The buyer then obtains valid data d_i. If a malicious seller who provides the invalid ciphertext c_i or data key k_i, he will be punished (analysis as described in Section 4.2.3). For the malicious buyer, he can not gain the data key k_i to decrypt the ciphertext c_i. □

Lemma 4. *The presented blockchain-based data trading protocol is fairness.*

Proof. First, we suppose that the seller is malicious, the seller can obtain the money but does not submit the valid data segment (the content of the data segment does not match its description) and data key. In this case, the buyer receives the invalid data and will apply for arbitration. The result of the arbitration is that the seller not only loses the buyer's money, but also loses his/her deposit, as described in Section 4.2.3. This contradicts the assumption. The seller can cheat with only negligible probability.

Then, we consider that the buyer is malicious and the seller is honest, in the sense that the buyer can gain the valid data without providing the money for data segment, or the buyer attempts to withdraw the money by submitting an arbitration application after receiving the valid data. For the former, the seller never sends ciphertext w_i for data key k_i to the buyer. The buyer is unable to extract the data key k_i from the ciphertext w_i to decrypt the c_i. For the latter, the result of the arbitration is that the buyer not only can not get his/her the money, but he/she also loses his/her deposit. Based on the above situation, a malicious buyer gains a contradiction. The probability of success for a malicious buyer is negligible. Therefore, the presented blockchain-based data trading protocol is fairness. □

Lemma 5. *The presented blockchain-based data trading protocol is decentralization.*

Proof. We consider that the protocol can solve the dispute over data availability in data trading through a decentralized Arbitration contract. In our protocol, when receiving the arbitration application submitted by the buyer, the Arbitration contract will generate an arbitration committee formed by randomly selecting a group of arbitrators, as described in Section 4.2.4. Each arbitrator in the committee has only one voting right, and needs to arbitrating and voting for the arbitration application. In this case, the result of the arbitration relies on a group of arbitrators rather than a single arbitrator. It is negligible that a single arbitrator can solve the dispute. Therefore, the presented blockchain-based data trading protocol is decentralization. □

6 Discussion of the arbitration

Let us first note that the number of arbitrators should be an odd number greater than 3. One reason is that if the number of arbitrators is too small, arbitrators are prone to corruption. For example, when the arbitration committee has only three arbitrators, the second arbitrator may directly choose a vote that is consistent with the vote of the first arbitrator in order to easily obtain the arbitration fees. In this case, the process of the arbitration is not very different from the case where there is only one arbitrator to arbitrate. Another reason is that an odd number of arbitrators can always produce an arbitration result. It is worth noting that if an arbitrator does not vote or has a problem in the process of voting, the final value of "value" may be equal to 0. In order to avoid this environment, the Arbitration contract additionally randomly selects an arbitrator to vote.

Arbitration candidates may decide whether to participate in arbitration according to the amount of arbitration fees. The number of arbitrators can be determined by both the buyer and the seller through an Arbitration contract, or by a Sale contract pre-designed by the seller. After k arbitrators execute voting, the vote is terminated and no new arbitrators are created. Note that when the number of votes supporting a party reaches $\frac{k+1}{2}$, even if some arbitrators have not yet arbitrated, the voting result is determined, and the voting can be terminated early. More than half of the voting results will be used as final arbitration results.

In the case of a dispute, the amount of the deposit should be sufficient to pay the arbitration fee. In order to facilitate the seller to submit the transaction deposit each time, the seller can prepay a certain amount of deposit to its "deposit pool[2]". In each transaction, the same amount as the buyer's deposit is taken from the seller's deposit pool and used as a deposit for the transaction. When the deposit is refunded, it will be returned directly to the deposit pool. The advantage is that both the buyer and seller deposits are the same, which reflects fairness; on the other hand, it reduces the seller's operations for each transaction, and also reduces the load of data storage on the blockchain. When the seller decides not to sell the data, the deposit in the deposit pool is returned to the seller after a period of lock-up. The purpose of setting a period of lock-up is to avoid having unfinished transactions in progress.

In the process of arbitration, if the deposit used as the arbitration fee is insufficient, it may result in fewer participants in the arbitration and the inability to complete the arbitration within a long period of time. At this time, both parties need to add a deposit. If the party fails to append the deposit, the Arbitration contract will automatically determine that he/she has failed.

The fail party arbitrated by the arbitration committee served his/her deposit as an arbitration fees to the arbitration committee and distributed it among the arbitrators. The method of distribution may be that all arbitrators divide the arbitration fees equally, or those arbitrators that votes are consistent with the

[2] "deposit pool" is a special account in the blockchain, the function of the account can be completed through a smart contract.

final result of the arbitration divide the arbitration fees equally. The latter can enable arbitrators to conduct fair arbitration, because only honest arbitrators can get more arbitration fees. However, in the case of a small arbitration committee, this method of distribution may result in the arbitrator not making a fair decision. For example, in order to get the arbitration fee early, the arbitrator only supports the party with the most votes currently. Therefore, the number of arbitrators is preferably greater than 3.

7 Performance evaluation

In this section, we execute a prototype of our protocol using smart contract in Solidity and programs in Java, and show its performance evaluation. Since Solidity does not provide the application programme interface (API) of Segmenting, DPKE and DDSA for the time of research, we will separately quantify the gas cost of smart contract and the time cost of cryptographic algorithms. We implement DPKE and DDSA algorithms with the Java cryptography API[3] (java.security and java.crypto) for our simulation. which the DPKE is variant of the RSA without padding and the DDSA is variant of the Elgamal. Our open source code related to this prototype has been posted online at $https://github.com/01007467319/TTL\text{-}ECO\text{-}SC.git$.

The simulation platform is Intel(R) Core(TM) i5-4570 CPU 3.20 GHz 4.00GB RAM and Windows 10 operate system. The time cost of Segmenting algorithms is shown in Table 2. The Segmenting algorithms has different time cost with different security level and different size of data encrypted. The AES.Enc algorithm and the AES.Dec algorithm are extremely fast, and their time cost are not very different on different security level. The time cost of cryptographic algorithms is shown in Table 3 (the modulus n is 1024 bits) and Table 4 with the average of 200 runs. The DPKE algorithm and the DDSA algorithm are cheap. If the DPKE.Gen algorithm only generates one key pair (sk, pk), it average costs 0.004 ms. We analyze the time cost of DDSA on different size of the prime number p in Fig. 4. See Table 4 and Fig. 4 for more details. We can find that the execution time of the DDSA.Ver algorithm is considerably fast, thus it is very easy for verifying an arbitrator.

We deploy the interacting Sale contract, Arbitration contract and Producing contract in our protocol by Solidity. They implement on a local computer based on Remix. The estimated gas cost to deploy, call and execute contract is provided in Table 5. The Sale contract to deploy is expensive since it includes to call Arbitration contract and Producing contract. The function of Sale contract consists of depositing deposit, returning deposit, paying money, submitting data and executing arbitration. Fig. 5 shows the relationship between the size of committee and execution time. This relationship is that, with the expansion of the size of the arbitration committee, the time for selecting arbitrators increases linearly.

[3] https://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html.

Table 2. The time cost of Segmenting algorithms

Security	Data size	Execute time (s)	AES.Enc (µs)	AES.Dec (µs)
128 bits	10 kb	0.45	0.94	0.96
128 bits	20 kb	0.64	0.96	0.95
128 bits	30 kb	0.84	0.97	0.96
256 bits	10 kb	0.48	1.29	1.28
256 bits	20 kb	0.71	1.30	1.29
256 bits	30 kb	0.89	1.26	1.28

Table 3. The time cost of DPKE algorithms

DPKE	DPKE.Gen (ms)	DPKE.Enc (ms)	DPKE.Dec (ms)
Max time	0.025	0.004	0.197
Min time	0.001	0.001	0.097
Average time	0.004	0.002	0.134

Table 4. The time cost of DDSA algorithms

DDSA	Size of p	DDSA.Gen (ms)	DDSA.Sign (ms)	DDSA.Ver (ms)
Max time	1024 bits	0.63	1.30	0.98
Min time	1024 bits	0.32	0.96	0.76
Average time	1024 bits	0.48	1.13	0.87

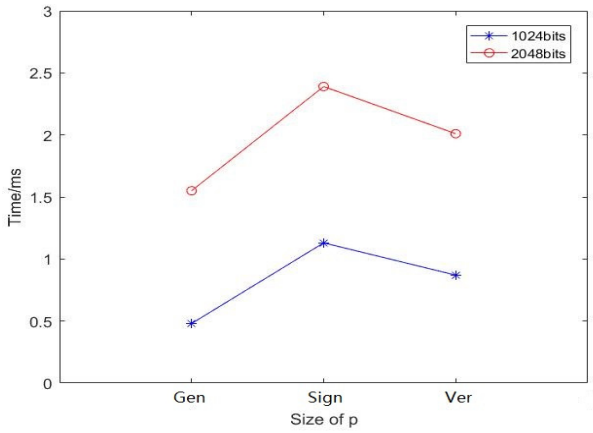

Fig. 4. The time cost of DDSA on different size of p

Table 5. Gas cost of smart contract

Smart contract	Function	Transaction gas	Execute gas	Gas cost (ether)
Sale	deploy contract	2099770	1554189	0.07307918
	deposit_seller	82312	61040	0.00286704
	deposit_buyer	104058	82786	0.00373688
	deposit_buyerdata	64218	40578	0.00209592
	payment_sellerdata	45049	21729	0.00133556
	payment_transaction	31430	41395	0.00145650
	withdraw	24389	27505	0.00103788
	arbitration_invoke	262510	272350	0.01069720
Producing	deploy contract	244230	145386	0.00779232
	producing	168265	144625	0.00625780
Arbitration	deploy contract	308436	194640	0.01006152
	arbitration	119763	101075	0.00441676

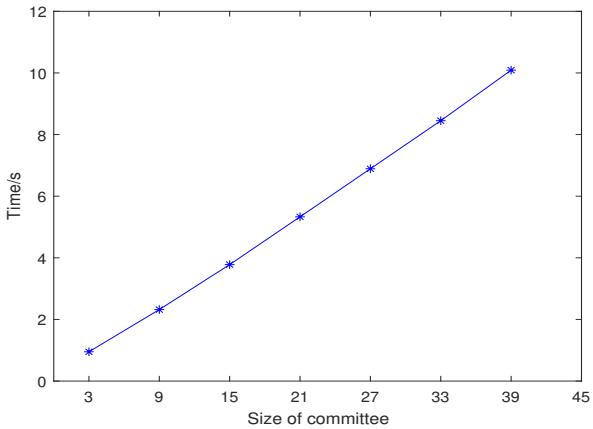

Fig. 5. The average time with the increase of committee size

8 Conclusion

In this paper, we have proposed a valid data trading ecosystem based on blockchain to complement deficiencies in existing data trading market. By introducing a decentralized arbitration mechanism consisting of an arbitration contract, a producing contract and a deterministic digital signature algorithm, we system can resolve the dispute over data availability in data trading. We also design a sale contract, advanced encryption standard algorithm and deterministic public-key encryption algorithm to ensure fairness of data trading between the seller and buyer. We utilize the advantage of blockchain to prevent single-point-failure of data trading platform. We formally prove the security of our system and discuss the arbitration process. The system was implemented based on smart contract and Java, and its performance evaluated.

Regarding future work, we will further research how to prevent the buyer from selling purchased (raw) data. Designing audit contract is used to detect whether the buyer resold the purchased data. However, a malicious buyer may try to transform the purchased data by implementing arbitrary operation, in order to evade the audit contract. Therefore, how to prevent the buyer from selling purchased data is an interesting and challenging research topic in the future.

Bibliography

[1] Federal Information Processing Standards Publication 197. Advanced encryption standard. In *U.S. Department of Commerce, National Institute of Standards and Technology*, 2001.

[2] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007*, pages 535–552, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[3] Mihir Bellare, Zvika Brakerski, Moni Naor, Thomas Ristenpart, Gil Segev, Hovav Shacham, and Scott Yilek. Hedged public-key encryption: How to protect against bad randomness. In *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, pages 232–249, 2009.

[4] Mihir Bellare, Marc Fischlin, Adam O'Neill, and Thomas Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, pages 360–378, 2008.

[5] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. *J. Cryptographic Engineering*, 2(2):77–89, 2012.

[6] Nina Bindel, Sedat Akleylek, Erdem Alkim, Paulo S. L. M. Barreto, Johannes Buchmann, Edward Eaton, Gus Gutoski, Juliane Krämer, Patrick Longa, Harun Polat, Jefferson E. Ricardini, and Gustavo Zanon. qtesla. *Submission to the NIST Post-Quantum Cryptography Standardization [NIS]*, https://qtesla.org, 2017.

[7] Alexandra Boldyreva, Serge Fehr, and Adam O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, pages 335–359, 2008.

[8] Zvika Brakerski and Gil Segev. Better security for deterministic public-key encryption: The auxiliary-input setting. *J. Cryptology*, 27(2):210–247, 2014.

[9] Jinchuan Chen and Yunzhi Xue. Bootstrapping a blockchain based ecosystem for big data exchange. In *2017 IEEE International Congress on Big Data, BigData Congress 2017, Honolulu, HI, USA, June 25-30, 2017*, pages 460–463, 2017.

[10] Xue-wen Chen and Xiaotong Lin. Big data deep learning: Challenges and perspectives. *IEEE Access*, 2:514–525, 2014.

[11] Joan Daemen and Vincent Rijmen. The block cipher rijndael. In *Smart Card Research and Applications, This International Conference, CARDIS*

'98, Louvain-la-Neuve, Belgium, September 14-16, 1998, Proceedings, pages 277–284, 1998.

[12] Weiqi Dai, Chunkai Dai, Kim-Kwang Raymond Choo, Changze Cui, Deiqing Zou, and Hai Jin. SDTE: A secure blockchain-based data trading ecosystem. *IEEE Trans. Information Forensics and Security*, 15:725–737, 2020.

[13] Sergi Delgado-Segura, Cristina Pérez-Solà, Guillermo Navarro-Arribas, and Jordi Herrera-Joancomartí. A fair protocol for data trading based on bitcoin transactions. *IACR Cryptology ePrint Archive*, 2017:1018, 2017.

[14] Massimo Felici, Theofrastos Koulouris, and Siani Pearson. Accountability for data governance in cloud ecosystems. In *IEEE 5th International Conference on Cloud Computing Technology and Science, CloudCom 2013, Bristol, United Kingdom, December 2-5, 2013, Volume 2*, pages 327–332, 2013.

[15] Ethereum Foundation. The serpent contract-oriented programming language.

[16] Benjamin Fuller, Adam O'Neill, and Leonid Reyzin. A unified approach to deterministic encryption: New constructions and a connection to computational entropy. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, pages 582–599, 2012.

[17] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[18] Taeho Jung, Xiang-Yang Li, Wenchao Huang, Jianwei Qian, Linlin Chen, Junze Han, Jiahui Hou, and Cheng Su. Accounttrade: Accountable protocols for big data trading against dishonest consumers. In *2017 IEEE Conference on Computer Communications, INFOCOM 2017, Atlanta, GA, USA, May 1-4, 2017*, pages 1–9, 2017.

[19] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, pages 357–388, 2017.

[20] Fan Liang, Wei Yu, Dou An, Qingyu Yang, Xinwen Fu, and Wei Zhao. A survey on big data market: Pricing, trading and protection. *IEEE Access*, 6:15132–15154, 2018.

[21] Kaitai Liang, Willy Susilo, and Joseph K. Liu. Privacy-preserving ciphertext multi-sharing control for big data storage. *IEEE Transactions on Information Forensics and Security*, 10(8):1578–1589.

[22] Jie Lin, Wei Yu, Nan Zhang, Xinyu Yang, Hanlin Zhang, and Wei Zhao. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet of Things Journal*, 4(5):1125–1142, 2017.

[23] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium. *Submission to the NIST Post-Quantum Cryptography Standardization [NIS]*, https://pq-crystals.org/dilithium, 2017.

[24] Sriharsha Mallapuram, Nnatubemugo Ngwum, Fang Yuan, Chao Lu, and Wei Yu. Smart city: The state of the art, datasets, and evaluation platforms. In *16th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2017, Wuhan, China, May 24-26, 2017*, pages 447–452, 2017.

[25] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers. Big data: The next frontier for innovation, competition, and productivity. pages https://www.mckinsey.com/business–functions/digital–mckin sey/our–insights/big–data–the–next–frontier–for–innovation, 2011.

[26] Andrew Mcafee and Erik Brynjolfsson. Big data: The management revolution. *Harv Bus Rev*, 90(10):60–6, 68, 128, 2012.

[27] Ilya Mironov, Omkant Pandey, Omer Reingold, and Gil Segev. Incremental deterministic public-key encryption. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 628–644, 2012.

[28] Mehdi Mohammadi, Ala I. Al-Fuqaha, Sameh Sorour, and Mohsen Guizani. Deep learning for iot big data and streaming analytics: A survey. *IEEE Communications Surveys and Tutorials*, 20(4):2923–2960, 2018.

[29] Thomas Pornin. Deterministic usage of the digital signature algorithm (DSA) and elliptic curve digital signature algorithm (ECDSA). *RFC*, 6979:1–79, 2013.

[30] Ananth Raghunathan, Gil Segev, and Salil P. Vadhan. Deterministic public-key encryption for adaptively-chosen plaintext distributions. *J. Cryptology*, 31(4):1012–1063, 2018.

[31] John A. Stankovic. Research directions for the internet of things. *IEEE Internet of Things Journal*, 1(1):3–9, 2014.

[32] Yunchuan Sun, Houbing Song, Antonio J. Jara, and Rongfang Bie. Internet of things and big data analytics for smart and connected communities. *IEEE Access*, 4:766–773, 2016.

[33] Nick Szabo. Formalizing and securing relationships on public networks. *First Monday*, 2(9), 1997.

[34] Y. Z. Wang, J. Hou, , and Y. Zhang. Data management based on block chain technology. *Electron. Des. Eng.*, 27:87–90 and 95, 2019.

[35] Hoeteck Wee. Dual projective hashing and its applications - lossy trapdoor functions and more. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 246–262, 2012.

[36] Jianjia Wu and Wei Zhao. Design and realization of winternet: From net of things to internet of things. *TCPS*, 1(1):2:1–2:12, 2016.

[37] Wei Xiong and Li Xiong. Smart contract based data trading mode using blockchain and machine learning. *IEEE Access*, 7:102331–102344, 2019.

[38] Maojiang Yang and Pingan Technology. A design of data trading platform based on cryptology and blockchain technology. *Information and Communications Technologies*, 2016.

[39] Xinyu Yang, Xuebin Ren, Jie Lin, and Wei Yu. On binary decomposition based privacy-preserving aggregation schemes in real-time monitoring systems. *IEEE Trans. Parallel Distrib. Syst.*, 27(10):2967–2983, 2016.

[40] Andrea Zanella, Nicola Bui, Angelo Paolo Castellani, Lorenzo Vangelista, and Michele Zorzi. Internet of things for smart cities. *IEEE Internet of Things Journal*, 1(1):22–32, 2014.

[41] Yanqi Zhao, Yong Yu, Yannan Li, Gang Han, and Xiaojiang Du. Machine learning based privacy-preserving fair data trading in big data market. *Inf. Sci.*, 478:449–460, 2019.

[42] Jiayu Zhou, Fengyi Tang, He Zhu, Ning Nan, and Ziheng Zhou. Distributed data vending on blockchain. In *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), iThings/GreenCom/CPSCom/SmartData 2018, Halifax, NS, Canada, July 30 - August 3, 2018*, pages 1100–1107, 2018.

[43] Cong Zuo, Jun Shao, Joseph K. Liu, Guiyi Wei, and Yun Ling. Fine-grained two-factor protection mechanism for data sharing in cloud storage. *IEEE Trans. Information Forensics and Security*, 13(1):186–196, 2018.