Information Sciences 544 (2021) 469–484

Contents lists available at ScienceDirect

Information Sciences

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

FAPS: A fair, autonomous and privacy-preserving scheme for big data exchange based on oblivious transfer, Ether cheque and smart contracts

Tiantian Li [a], Wei Ren [a,b,c,*], Yuexin Xiang [a], Xianghan Zheng [d,e], Tianqing Zhu [a], Kim-Kwang Raymond Choo [f], Gautam Srivastava [g,h]

[a] School of Computer Science, China University of Geosciences, Wuhan, PR China
[b] Guangxi Key Laboratory of Cryptography and Information Security, Guilin 541004, PR China
[c] Key Laboratory of Network Assessment Technology, CAS, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, PR China
[d] Fuzhou University, Fuzhou, PR China
[e] Mingbyte Technology, QingDao, PR China
[f] Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX 78249-0631, USA
[g] Department of Mathematics and Computer Science, Brandon University, 270 18th Street, Brandon, Canada
[h] Research Center for Interneural Computing, China Medical University, Taichung 40402, Taiwan

ARTICLE INFO

Article history:
Received 16 March 2020
Received in revised form 27 August 2020
Accepted 29 August 2020
Available online 14 September 2020

Keywords:
Data exchange
Fairness
Privacy preservation
Smart contract
Oblivious transfer
Ether cheque

ABSTRACT

Decision-making plays an increasing important role in our digitalized society, and hence the need for trustworthy data. For example, in a traditional trading process, a trusted middle person generally helps to arbitrate any dispute between two or more parties involved in the trading. However, having such a trusted middle person can be labor-intensive, increases transaction costs, and lacks fairness during arbitration due to the subjectivity of the middle person. While we can avoid the need for a middle person using smart contract-based approaches, there are other limitations (e.g., fairness and privacy preservation) that need to be addressed. In this paper, we present a fair scheme for big data exchanging that allows buyers and sellers to autonomously and fairly complete transactions, without involving any third-party middle person. Specifically, our scheme uses smart contracts and oblivious transfer protocol, in combination with our proposed Ether cheque system. Smart contracts are used to achieve transaction fairness, autonomy and trading timing control. The oblivious transfer protocol helps us to preserve the privacy of transactions. We use Ether cheque to improve the fairness of the transaction and make the transaction more convenient. Also, our proposed approach can facilitate the identification of a cheating party.

1. Introduction

Data is now the new currency in our smart and interconnected society, where data underpins many of our decision-making and strategy formulations. For example, commercial organizations can analyze data from different sources to profile their (prospective) customers, to deliver customized products or services, or adjust the delivery of their products or services

* Corresponding author at: School of Computer Science, China University of Geosciences, Wuhan, PR China.
E-mail addresses: weirencs@cug.edu.cn (W. Ren), raymond.choo@fulbrightmail.org (K.-K.R Choo).

https://doi.org/10.1016/j.ins.2020.08.116
0020-0255/

based on personalized demands, etc. With the significant growth in big data transactions, there is also a corresponding need to formulate rules and systems to protect the fairness and privacy of such transactions. However, there are challenges in dealing with the significant volume, veracity and volume of data. For example, conventional servers are generally incapable of storing or processing such large data in a timely or cost-efficient manner.

Besides, there are trust issues. It is not realistic to expect that users or other entities in the ecosystems to trust each other completely. The trust issues give rise to the need for some trusted third-party entity to help the completion of online transactions. However, the introduction of a trusted third-party has high cost and resource implications. Moreover, how trustworthy are these third parties?

Existing schemes do not generally achieve fairness, privacy and autonomy simultaneously. For example, Dziembowski et al. [5] used a smart contract to design a digital good exchange protocol, which facilitates the data exchange between the buyers and the sellers without involving a trusted third-party. However, privacy preservation has not been considered in the protocol. For example, Aiello et al. [6] proposed a protocol based on the oblivious transfer protocol, which does not considered fairness in data exchanging.

Even though the smart contract is widely applied, a challenge is that all data on the smart contract is publicly accessible by everyone, which complicates the designing of the privacy-preserving schemes. Therefore, we posit the potential of combining smart contracts with the oblivious transfer protocol in order to preserve the privacy of buyers and sellers, without involving any third-party. Specifically, we define the *privacy* information in the following sense:

- The seller does not know which parts of the data the buyer has received.
- The buyer only knows the parts of the data (s)he has received, but (s)he knows nothing about other parts of the data.

Our proposed scheme is designed to achieve *fairness* as explained below:

- In each transaction, data will be recorded and used as evidence in the dispute.
- Even if any party has cheated in the transaction, the other party's interests are not affected.
- Any user can report cheating to the system, and the system will impose a penalty (e.g., deposit forfeiting or blacklisting) on the cheating party.

A summary of our proposed scheme is presented below:

1. Our smart contract-based data trading scheme, buyers and sellers can fairly and autonomously complete transaction(s) without involving any third-party entity. In addition, we use time limits to realize trading timing control.
2. We apply m-out-of-n oblivious transfer protocol on the transactions to achieve enhanced privacy and security.
3. We propose a novel Ether cheque system to achieve fairness and autonomy, which is also more user-friendly.

The layout for the remaining is as follows. In the next two sections, we will briefly review the related literature and present the problem formulation, respectively. Our proposed scheme is presented in Section 4. In Section 5, we explain how to deal with exception handling. We also evaluate the performance of our proposed approach in the same section. In Section 6, we conclude this paper.

2. Related work

The oblivious transfer has been studied since the 1990s, for example by Beaver et al. [7]. In 2001, Aiello et al. [6] proposed the first one-round (two-pass) protocol for oblivious transfer, which does not rely on the random oracle model. Since then, there have been other extensions or improved protocols, such as those of Mu et al. [8] and Rabin [9]. For example, Rabin [9] proposed using the oblivious transfer protocol to allow a sender to transmit a message to a receiver, where the receiver has a probability between 0 and 1 of learning the message. However, the sender is unaware of whether the receiver can do so. Chu et al. [10] presented an efficient two-round k-out-of-n oblivious transfer scheme. Later in 2015, an oblivious transfer protocol with verification was presented by Kak [11], designed to minimize the risk of cheating in generating probability events using random sequences.

Liu et al. [12] designed a key exchange strategy for big data applications, in order to realize more efficient authenticated key exchange without sacrificing data security. Cheng et al. [13] proposed a scheme to protect the mapping of different data elements of each provider using a trapdoor function. Other approaches designed to manage and secure big data, such as medical big data, include those of Azaria et al. [14] and Xia et al. [4]. Also, Oh et al. [15] and Jiang et al. [16] proposed data trading schemes for the Internet of things (IoT) that focus on trading data efficiently and ensuring the security of data exchange respectively.

A number of approaches to facilitate secure big data trading have also been proposed in the literature – e.g. see the survey of Liang et al. [3]. Dong et al. [1], for example, proposed a big data sharing framework, designed to secure and share user data effectively. Jung et al. [2] proposed using an index to measure data uniqueness, and when misbehavior is detected, the data brokers can identify the dishonest consumer.

There have also been attempts to utilize blockchain in big data applications [17–20]. For example, Yang et al. [21] proposed a blockchain-based approach to minimize the risk of transactional record leakage.

Also, the rapid development of artificial intelligence technologies such as machine learning also brings new possibilities for big data exchange. Xiong et al. [22] designed a scheme using machine learning to solve the dispute during data trading. Zhao et al. [23] proposed a machine learning and blockchain based data exchange scheme that achieves privacy preservation and fairness.

3. Problem formulation

Prior to presenting our proposed approach in the next section, we will briefly introduce our system and adversary models as well as the design goals here.

3.1. System model

In this model, we assume there are two participants, namely: a seller (say, Alice) and a buyer (say, Bob). Alice has the data Bob is interested in.

To simplify the transfer of data in the transaction, Alice needs to convert the data into keys. In this case, the data transmitted in the transaction is a short string rather than some large-sized data. Our scheme uses a distributed file storage system, say InterPlanetary File System (IPFS). Since IPFS is content-addressable, the data address changes as the content changes. Alice needs to upload the data for sale to IPFS, where she will get the address corresponding to the data and the key associated with the address.

In addition, in order to meet the requirements of the m-out-of-n oblivious transfer protocol, Alice needs to divide the data into n blocks and generate n corresponding keys, which are then encrypted and uploaded to the smart contract. When Bob gets the address key of the purchased data, he is supposed to upload the Ether cheque for payment to the smart contract. At the same time, Bob will use the keys to check the data on IPFS, and any problem is detected, he can report to the contract.

After the transaction, Alice will get the cheque and Bob will get the purchased data. Any alleged cheating behavior will be investigated.

The system model is shown in Fig. 1.

3.2. Adversary model

There are a number of potential threats in any transaction, and we categorize such threats into the following:

Category 1. Threats from a dishonest seller.

- *The product sold by the seller is inconsistent with the advertised product, for example, the seller replaces the advertised product with another product.* Since the IPFS is used, any changes in the data will result in the corresponding address changing. Hence, this is evidence that can be used subsequently.
- *The seller provides non-existent data keys or addresses.* Clearly, when the buyer discovers that the data keys and/or addresses are non-existent, (s)he can lodge a claim within the system specified time.

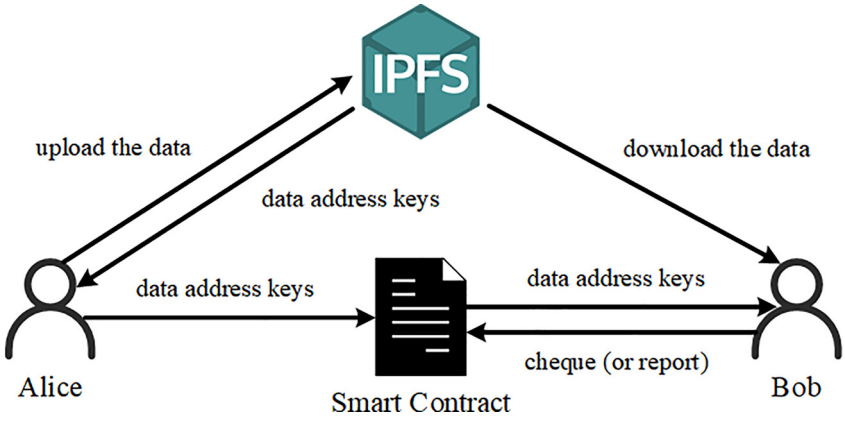

Fig. 1. System model.

Category 2. Threats from a dishonest buyer.

- *The buyer refuses to pay after receiving the data.* This will result in the seller not getting the payment, resulting in unfair trading. Such non-payment behavior will be penalized by forfeiting the buyer's deposit, and blacklisting the buyer (i.e., recording this dishonest behavior on the blockchain).
- *The buyer terminates the transaction abnormally, when (s)he has received the purchased data, or part of the purchased data which has a value higher than his/her deposit.* The deposit amount is equal to or higher than that of the transaction, and hence the incentive to cheat by terminating the transaction half-way is removed.
- *The buyer initiates the transaction multiple times with completing any of these transactions.* This takes up resources and leads to denial of service. There are a number of ways that such an activity can be mitigated. For example, we can record the number of transactions initiated by a buyer and limit the number of daily transactions per user. In other words, if the number of duplicate transactions exceeds the limit and none of these transactions are complete, then the system will prevent further trading service requests from the same user for a predetermined amount of time (e.g., 24 h).

Category 3. Threats from others apart from the buyer and the seller.

- *The attacker impersonates the seller or the buyer, in an attempt to steal the data.* Smart contracts are used to mitigate impersonation attacks to a certain extent based on the identity authentication of addresses. That means only the entity corresponding to the address that creates the transaction and deploys the smart contract, and the entity corresponding to the address that initiates the transaction can take part in the process of the transaction.
- *The smart contract is attacked.* In the case of security bugs existing in the smart contract, the attacker can steal the tokens in the smart contract or disrupt the normal trading order, causing economic losses for the buyer and the seller. In our scheme, we assume that the smart contract is secure.

3.3. Design goals

The design goals of our data exchange scheme is as follows:

1. *Fairness.* After the transaction terminates normally, Bob receives the data he had paid for, and Alice receives the proceeds of the sale. Any alleged misbehavior will be investigated and penalized (e.g., deposit forfeit, or blacklisting).
2. *Autonomy.* Alice and Bob autonomously complete the transaction through a smart contract without the help and arbitration of any third-party entity.
3. *Privacy Preservation.* During the transaction, Bob only knows the blocks of the data he bought, and he knows nothing about Alice's other data. At the same time, Alice does not know which blocks of the data Bob received from her.
4. *Timing Control.* The transaction must be completed within the specified time, rather than being delayed indefinitely.

4. Proposed scheme

Our scheme mainly applies smart contracts and oblivious transfer protocol. To achieve fairness and autonomy and avoid involving any third-party entity, we use smart contracts to complete our scheme. The content of the smart contract can be executed autonomously, and the content of the smart contract is public and agreed by buyers and sellers before the transaction that cannot be modified by anyone during execution. Otherwise, the application of oblivious transfer protocol in our scheme can enhance fairness and privacy preservation. Because of the characteristics of the oblivious transfer protocol, the process of oblivious transfer can help us judge if the participants of the transaction have violated the transaction rules and oblivious transfer protocol also guarantees that the private information will not be leaked to a certain extent.

A summary of the notations used is listed in Table 1. Additionally, if there is no specific explanation, smart contracts appearing in this section and the following sections are referred to the smart contract for trading.

Next, we will introduce the first building block of our proposed scheme.

4.1. Ether cheque

We propose using the Ether cheque, instead of tokens, to pay for the data. The characteristic of the Ether cheque is mainly reflected in the following two aspects: First, the buyer does not need to pay tokens for every initiation of a transaction, the buyer only needs to transfer the tokens from the storing smart contract to the trading smart contract by the Ether cheque; Second, when the transaction is terminated due to abnormal operations, the tokens will directly return to the storing smart contract, and the buyer can use the refunded tokens in the storing smart contract for the next transaction, instead of being

Table 1
Notations.

Notations	Description
AK_i	Symmetric key of Alice for encrypting HK_i
$Block_{price}$	Price of each data block Alice sells
$Block_{num}$	The number of data blocks Bob decides to purchase
$Cheque_{BA}$	The cheque Bob has signed for Alice
D_i	Each data block
DA_i	The address of each data block
$Deposit_A$	The amount of deposit Alice sends to the smart contract
$Deposit_B$	The amount of deposit Bob sends to the smart contract
EHK_i	HK_i encrypted by AK_i
HK_i	The key corresponding to DA_i generated by IPFS
K_i	Symmetric key of Bob
PK_{Alice}	Public key of Alice
PK_{Bob}	Public key of Bob
(PK_i, SK_i)	A pair of public and private keys
R	A random number used in the Ether cheque
SK_{Alice}	Private key of Alice
SK_{Bob}	Private key of Bob
$TimeLimit_i$	The time limit during the transaction process

refunded to the buyer's address. In addition, some important trading information of the buyer and the seller are recorded in the Ether cheque that is helpful for the judgment of cheating behaviors. The below example explains how it works.

The transaction requires two deployed smart contracts, one for storing tokens and one for trading. If Bob wishes to trade, he needs to send sufficient tokens to the smart contract that stores the tokens prior to the transaction. As part of the data sale, if Bob wants to pay Alice 10 *ether*, he should first randomly generate a number R using a random number generator (RNG) and calculate the hash value of $Hash_R$. Bob also needs to encrypt R with Alice's public key, denoted as $E_{PK_{Alice}}(R)$.

Then, Bob generates a complete cheque and sends it to the trading smart contract. The specific description of the cheque is as follows:

$$Cheque_O = PK_{Alice} \| Amount \| Date \| E_{PK_{Alice}}(R) \| Hash(R), \tag{1}$$

where PK_{Alice} is Alice's public key, $Amount$ is the number of tokens Bob should pay Alice, and $Date$ is the time when Bob generated the cheque. And Bob needs to sign the cheque as:

$$Cheque_{BA} = Sig(Cheque_O, SK_{Bob}), \tag{2}$$

where $Sig(\cdot)$ is any digital signature algorithm, SK_{Bob} is Bob's private key. Then, Alice can obtain the cheque from the contract to commence trading. This process is illustrated in Fig. 2.

When Alice wants to use $Cheque_{BA}$ to pay or withdraw the tokens, she should use Bob's public key to verify the signature of the cheque. After that, Alice will get the details of the cheque, and she can check the amount and the date. In order to prove her identity, Alice needs to decrypt $E_{PK_{Alice}}(R)$ with her private key SK_{Alice} and get the random number R'.

Then, Alice submits the number R' to the trading smart contract. That contract will calculate the hash value of R' and compare $Hash_{R'}$ with the previous value of $Hash_R$ to verify whether the user of the cheque is indeed Alice. If the verification is

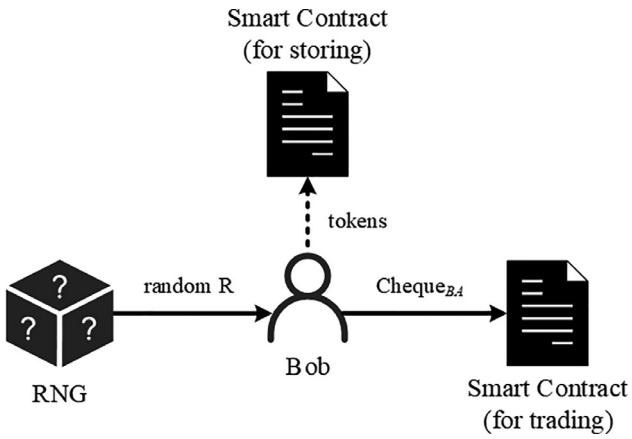

Fig. 2. Process of Bob generating a cheque.

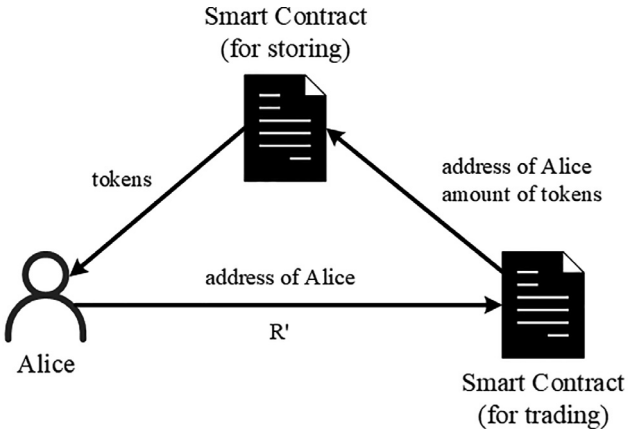

Fig. 3. Process of Alice using the cheque.

successful, the contract for trading will send Alice's address and the number of tokens to the contract for storing tokens. After that, the contract for storing tokens will send the tokens to Alice's address and deduct the equivalent tokens from Bob's balance. The flow chart describing the above process is shown in Fig. 3.

4.2. Basic setting

Before the transaction starts, the following three operations need to be completed first:

1. *Alice decides how many blocks the data should be divided into.* For example, Alice sets the number of data blocks to n. It should be noted that if the value of n is small, it may benefit Alice but not the buyer. Consequently, it will deter the prospective buyer. On the other hand, if the value of n is too large, this will be more favorable to the buyer. Buyers and sellers can negotiate the value of n on the smart contract. Only when both parties agree on the same number, will the transaction continue.
2. *Bob decides the number of data blocks he wants to buy.* Assuming Bob wants to buy m blocks of data, the requirement to be met by m is $1 \leqslant m \leqslant n$. Usually, the value of m is small. Bob checks the m blocks of data to determine if he wants to make the next transaction with Alice.
3. *Alice sets the size of $Deposit_B$, which is equal to $Deposit_A$.* To guarantee the fairness of a transaction, both parties should pay the deposit to the smart contract first. During the transaction, if one party violates the rules, the offending party's deposit will be forfeited.

4.3. Transaction processes

In this section, we will introduce our scheme and the algorithm, and the specific transaction process for both buyers and sellers.

4.3.1. Initialization of smart contracts

To initiate the transaction, Alice should upload the data to IPFS and send $Deposit_A$ to the smart contract. In order to trade fairly, Bob needs to send $Deposit_B$ to the smart contract, which is equal to $Deposit_A$. The detailed description of the above process is shown as follows (the range of all i in this section is $1 \leqslant i \leqslant n$).

1. Alice gets data blocks D_is by dividing the data into n blocks.
2. Alice uploads D_is to IPFS, so as to get data address DA_is and the corresponding key HK_is.
3. Alice sends $Deposit_A$ and $Block_{price}$ to the smart contract. If the amount of $Deposit_A$ is incorrect (i.e. lower than agreed), the smart contract will not accept the request until Alice sends the right amount of $Deposit_A$.
4. Alice sends her public key PK_{Alice} to the smart contract.
5. Before Bob initiates the trade, he should store sufficient tokens (more than $Block_{num} \times Block_{price}$) on the smart contract for storing, and prepare the payment cheque $Cheque_{BA}$.
6. Bob sends $Deposit_B$ to the smart contract.
7. Bob sends $Block_{num}$ to the smart contract which is the the number of blocks Bob wants to purchase.
8. Bob sends his public key PK_{Bob} to the smart contract.
9. Alice uses Bob's public key PK_{Bob} to encrypt each HK_i, that is $E_{PK_{Bob}}(HK_i)$.

10. Alice generates n symmetric keys AK_is, so as to encrypt each $E_{PK_{Bob}}(HK_i)$. Then, get $E_{AK_i}(E_{PK_{Bob}}(HK_i))$ ie EHK_i, and send EHK_i to the smart contract.

If Alice or Bob sends an incorrect deposit amount, the smart contract will refuse the request until the contract receives the right deposit amount. Alice and Bob must complete the above operation within a specified time duration, *TimeLimit*; otherwise, the transaction will be terminated by the system.

The initialization processes are shown in Fig. 4.

We implement some of the main processes of the transaction using the smart contract, and the specific algorithm of the smart contract for dealing with the transaction is shown in Algorithm 1.

Algorithm 1. Smart Contract for Trading

Input: $Deposit_A$, $Deposit_B$, PK_{Alice}, PK_{Bob}, $Block_{price}$, $Block_{num}$, $Cheque_{BA}$, EHK_i, $TimeLimit$

Output: the result of the transaction $Result\_Trans$, the time when the steps of Alice starts $TimeAlice$, the time when the steps of Bob starts $TimeBob$

```
Alice sets TimeLimit, and TimeAlice = now;
Alice sets the amount of Deposit_A, Deposit_B and Block_price;
Alice sends Deposit_A to the smart contract;
Alice uploads PK_Alice to the contract;
if The amount of Deposit_A is right and now ≤ TimeAlice + TimeLimit then
    TimeBob = now;
    Bob sends Block_num and Deposit_B to the contract;
    Bob uploads PK_Alice to the contract;
    if The amount of Deposit_B is right and now ≤ TimeBob + TimeLimit then
        TimeAlice = now;
        Alice sends EHK_is to the smart contract;
        if Bob confirms receipt of the EHK_is and now ≤ TimeAlice + TimeLimit
         then
            TimeBob = now;
            Bob sends Cheque_BA to the smart contract;
            if Cheque_BA is right and now ≤ TimeBob + TimeLimit then
                Result_Trans = true;
            else
                Result_Trans = false;
                The smart contract refunds Deposit_A and Deposit_B;
        else
            Result_Trans = false;
            The smart contract refunds Deposit_A and Deposit_B;
    else
        Result_Trans = false;
        The smart contract refunds Deposit_A;
else
    Result_Trans = false;
    The smart contract refunds Deposit_A;
```

4.3.2. Initiation of oblivious transfer protocol

For exchanging data fairly, the parameters of oblivious transfer protocol should be set by Alice and Bob.

Firstly, Alice generates n pairs of public keys and private keys noted as:

$$(PK_i, SK_i), \quad i \in (1 \leqslant i \leqslant n). \tag{3}$$

Then, Bob decides how many blocks of the data he will buy, assuming that $Block_{num}$ is m. In order to satisfy the requirement of oblivious transfer protocol, Bob should generate m symmetric keys as follows:

$$K_1, K_2, \ldots, K_m. \tag{4}$$

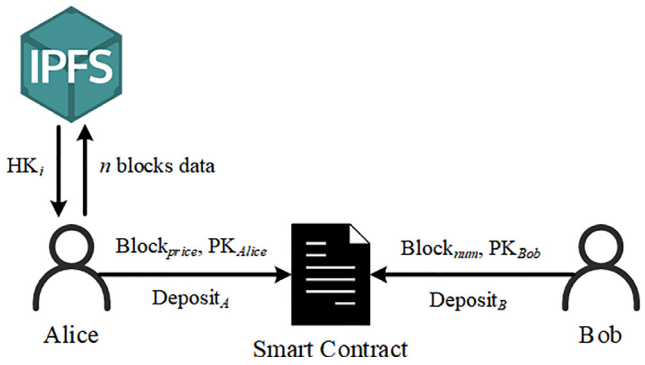

Fig. 4. Initialization processes.

4.3.3. Execution of the transaction

After the initialization of smart contracts and oblivious transfer protocol, the transaction begins. In this stage of the transaction, Bob can freely choose m blocks of the data to purchase and Alice will not know which blocks of data Bob chose to buy. The specific transaction steps of this stage are shown as follows:

1. In order to transfer the data, Alice should send encrypted data keys (Eq. (5)) and generated n public keys (Eq. (6)) to the smart contract:

$$\{EHK_1,\ EHK_2, \ldots,\ EHK_n\}, \tag{5}$$

$$\{PK_1,\ PK_2, \ldots,\ PK_n\}. \tag{6}$$

2. Then, Bob randomly selects $m (1 \leqslant m \leqslant n)$ PK_is from Alice's n PK_is. The selected keys are as follow:

$$\{PK_{i_1},\ PK_{i_2}, \ldots,\ PK_{i_m}\},\ i \in \{1, 2, \ldots, n\}. \tag{7}$$

Next, Bob uses selected m PK_is to encrypt m symmetric keys $K_1, K_2, \ldots, K_m$ (Eq. (8)):

$$\{EK_1,\ EK_2, \ldots,\ EK_m\}. \tag{8}$$

Additionally, the hash value of each K_i needs to be calculated (Eq. (9)). Then Bob sends both encrypted m symmetric keys (Eq. (8)) and their hash values (Eq. (9)) to the smart contract.

$$\{Hash(K_1),\ Hash(K_2), \ldots,\ Hash(K_m)\}. \tag{9}$$

3. Alice uses each private key SK_i of public key PK_i to decrypt each $\{EK_1, EK_2, \ldots, EK_m\}$ from the smart contract. The result is expressed using Eq. (10), and will be uploaded to the smart contract:

$$\begin{pmatrix} DK_{11} & DK_{21} & \ldots & DK_{n1} \\ DK_{12} & DK_{22} & \ldots & DK_{n2} \\ \ldots & \ldots & & \ldots \\ DK_{1m} & DK_{2m} & \ldots & DK_{nm} \end{pmatrix}. \tag{10}$$

Note: There are $n \times m$ results above in total. The line i represents all the possibilities of the symmetric K_i, $i \in (1 \leqslant i \leqslant m)$. Additionally, each line has only one right key K_i and the others are incorrect.

4. After above steps, Alice uses $n \times m$ decrypted keys (Eq. (10)) to encrypt her symmetric keys $\{AK_1, AK_2, \ldots, AK_n\}$, and the results are shown below as (Eq. (11)). In addition to this, Alice needs to send the result (Eq. (11)) to the smart contract.

$$\begin{pmatrix} E_{DK_{11}}(AK_1) & E_{DK_{21}}(AK_2) & \ldots & E_{DK_{n1}}(AK_n) \\ E_{DK_{12}}(AK_1) & E_{DK_{22}}(AK_2) & \ldots & E_{DK_{n2}}(AK_n) \\ \ldots & \ldots & & \ldots \\ E_{DK_{1m}}(AK_1) & E_{DK_{2m}}(AK_2) & \ldots & E_{DK_{nm}}(AK_n) \end{pmatrix}. \tag{11}$$

5. In this step, Bob uses his symmetric keys $\{K_1, K_2, \ldots, K_m\}$ to decrypt the result (Eq. (11)), described as follow:

$$\begin{pmatrix} D_{K_1}(E_{DK_{i1}}(AK_i)) \\ D_{K_2}(E_{DK_{i2}}(AK_i)) \\ \ldots \\ D_{K_m}(E_{DK_{im}}(AK_i)) \end{pmatrix},\ i \in [1, n]. \tag{12}$$

Note: Bob can obtain only one right AK_i in each line, and he will get m different AK_i. Bob also does not have to calculate all results, because he knows which PK_i he has chosen. It means that Bob knows which key to decrypt in each line to get the correct result. Finally, Bob will get m blocks of the data by using AK_i to decrypt EHK_i.

6. When Bob receives m blocks of data he purchased, he should send $Cheque_{BA}$ to the smart contract. Then, Bob has some time to check the keys of data addresses he has obtained on IPFS. If any problem is detected, he can report this to the contract. Otherwise, the transaction ends successfully. After that, the smart contract will refund $Deposit_A$ to Alice and $Deposit_B$ to Bob.

According to the above description, the process of Bob purchasing m block data is shown in Fig. 5.

4.3.4. Optional transaction

If Bob wants to purchase some parts of the remaining data or buy data from other datasets belonging to Alice, Bob should initiate a request for the new transaction. And the new transaction will execute as the steps in the previous section.

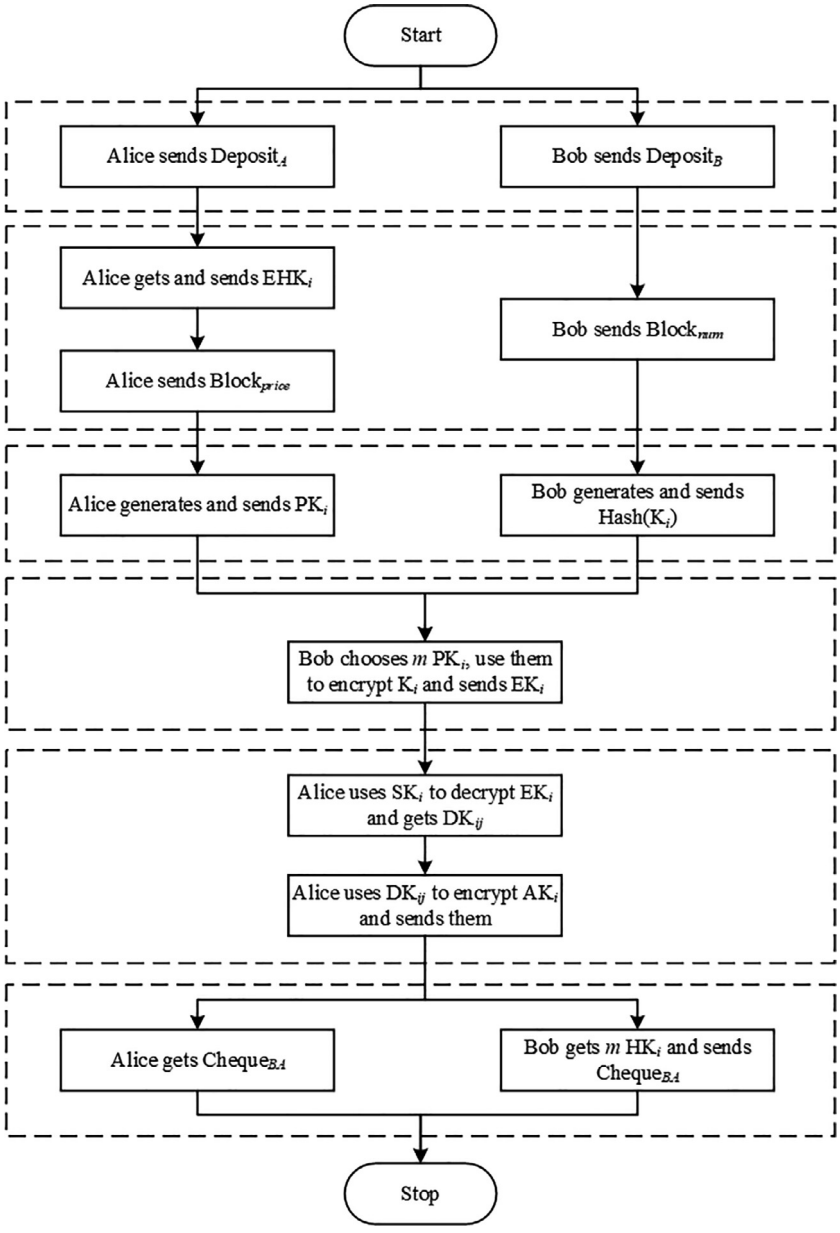

Fig. 5. The process of exchanging m blocks.

But if Bob would like to carry on with obtaining all the rest of the data, he can continue the former transaction with Alice. And the steps of continuing the transaction is shown below.

1. First, Alice and Bob should respectively send $Deposit_A$ and $Deposit_B$ to the smart contract. If the amount of $Deposit_A$ or $Deposit_B$ is incorrect, the smart contract will not respond to the transaction until both of their deposits are right.
2. Alice uses Bob's public key to encrypt all the private keys, expressed mathematically below:

$$\{E_{PK_{Bob}}(SK_1 \| SK_2 \| \ldots \| SK_n)\}, \tag{13}$$

then Alice sends the results in Eq. 13 to the smart contract.
3. Bob uses his private key SK_{Bob} to decrypt Eq. (13) to obtain $\{SK_1, SK_2, \ldots, SK_n\}$. Then, Bob chooses one public key from those he has selected

$$\{PK_{i_1}, PK_{i_2}, \ldots, PK_{i_m}\}, i \in \{1, 2, \ldots, n\}. \tag{14}$$

We assume he chooses $PK_{y_x}(1 \leqslant y \leqslant n, 1 \leqslant x \leqslant m)$.
Now, Bob needs to decrypt $EK_x(1 \leqslant x \leqslant m)$ using $\{SK_1, SK_2, \ldots, SK_n\}$. Finally, Bob will get $\{DK_{1x}, DK_{2x}, \ldots, DK_{nx}\}$. It is vital to note that DK_{yx} is the right K_x, and the other $(m-1)$ results are wrong.
4. According to the result of the first stage, Bob decrypts the line x of the matrix (Eq. (11)) by using $\{DK_{1x}, DK_{2x}, \ldots, DK_{nx}\}$. Then, Bob will obtain $\{AK_1, AK_2, \ldots, AK_n\}$, which can be used to decrypt the corresponding encrypted data keys $\{EHK_1, EHK_2, \ldots, EHK_n\}$. Finally, Bob gets all the keys of data addresses $\{HK_1, HK_2, \ldots, HK_n\}$, and he can obtain all the n blocks of the data from IPFS.
5. When Bob has received the keys, he needs to send a payment cheque to the smart contract. He also has some time to check whether the keys of data addresses are valid. If there is no dispute in the transaction, the smart contract will refund $Deposit_A$ to Alice and $Deopsit_B$ to Bob. Then, the transaction ends successfully and both parties get what they want.

4.4. Timing control

To enhance fairness, we set several time limits. Within each limit, Alice and Bob need to complete some work. In the Algorithm 1, we only describe some important time control steps, here we describe in detail all the essential time control. The specific definition is as follows.

T_0: Alice and Bob need to pay $Deposit_A$ and $Deposit_B$ respectively to the smart contract within T_0. Otherwise, the transaction terminates and any previously paid deposit will be refunded.

T_1: Within T_1, Alice should upload EHK_i and the price of each block of data. In the meantime, Bob should submit the quantity of data he wants to purchase. Otherwise, the transaction terminates.

T_2: Within T_2, Alice generates n pairs of $PK - SK$ and submits n public keys to the smart contract. At the same time, Bob generates and uploads his symmetric keys $\{K_1, K_2, \ldots, K_n\}$ to the smart contract. Otherwise, the transaction terminates.

T_3: Within T_3, Bob chooses m public keys from Alice and submits $\{EK_1, EK_2, \ldots, EK_m\}$ to the smart contract. Otherwise the transaction terminates.

T_4: Within T_4, Alice decrypts m EK_i with n private keys and uploads encrypted n AK_i. Otherwise the transaction terminates.

T_5: Within T_5, Bob is supposed to pay the blockchain cheque to the smart contract. Meanwhile, Alice should verify the received cheque and convey the message to the smart contract. Otherwise, the transaction terminates.

T_b: We set a buffer time T_b to ensure that Bob has sufficient time to check the received keys and data, in order to allow Bob to raise any concern with the transaction within T_b. Meanwhile, only after buffer time T_b can Alice withdraw the fund in the Ether cheque. And after time T_b, $Deposit_A$ and $Deposit_B$ will be sent back to Alice and Bob by the smart contract.

4.5. Enhancement

4.5.1. Credit mechanism

We design a credit mechanism based on smart contracts. All entities who take part in the data exchange will have 0 score at the beginning. For example, if the buyer (e.g., Bob) is the cause of an unsuccessful data exchange transaction due to the buyer's fault, the smart contract will record the address of Bob and decrease his scores, say by one. If the termination is due to timeout, then no penalty will be imposed. If the transaction is successful, then both parties' scores will be increased, say by one.

Therefore, by checking scores, the buyer or the seller can easily obtain the credit information of all the transactions of the other party in this contract. Also, it is possible to call other contracts for their credit information and determine the trustworthiness of the buyer or seller. This, however, can be costly.

4.5.2. Simplify the interaction

We observe that in the scheme both Alice and Bob need to interact with the smart contract five times during the exchange of m blocks of data. In the process of interaction, the buyer and the seller upload the significant volume of data to the con-

tract. The data uploaded by Alice or Bob in the transaction can be used as evidence to resolve any dispute and hence, achieve fairness. However, this consumes bandwidth and increases the cost. Therefore, we consider reducing resource consumption by decreasing the number of interactions and simplifying interaction data.

5. Analysis and evaluation

5.1. Analysis of fairness

If both parts of the transaction are honest, the protocol will be successfully completed. That is the optimal situation. However, some unexpected states exist. Thus, we need to use the data on the smart contract as evidence to find the abnormal step and the participant who violates the rules. In addition, we also apply the deposit rule. The deposit of the offending party will be forfeited and paid to the other party.

1. We set the $Deposit_B$ higher than the price of m blocks of data. This will disincentivize Bob from terminating the transaction abnormally after he has obtained the data.
2. The Ether cheque has been uploaded to the smart contract so that the identity of payer and payee and the amount of token can be verified by others. If any item in the Ether cheque differs from what Bob has claimed, Bob will have a dishonest record on the blockchain.
3. At the end of the first transaction stage, Bob can raise any objection about the transaction within the specified timeframe. So if Bob claims that Alice is cheating and he does not obtain all m blocks of valid data addresses with keys provided by Alice, we will verify that problem by the following process:
   (1) Bob needs to provide one pair of keys and address through which he cannot get the valid data, say AK_x and HK_x. Bob uploads AK_x and HK_x to the smart contract as evidence and the smart contract will calculate and check if $Hash(AK_x)$ is equal to one of the previous values Alice stored.
      a. If this is found to be true, AK_x provided by Bob is truly from the transaction, then we skip to step (2).
      b. If this is found to be false, AK_x must be fabricated by Bob or Alice uploaded the wrong data. Then, we skip to step (3).

   (2) Calculate if $E_{AK_x}(E_{PK_{Bob}}(HK_x))$ is the previous EHK_x that Alice stored at the beginning of transaction.
      a. If this is found to be true, HK_x provided by Bob must come from their exchange process. Other parties can check the data on address HK_x. If the address and the data are available, Bob is determined to be misbehaving. If the address is invalid, Alice is determined to be a dishonest seller. Then, the adjudication process ends.
      b. If this is found to be false, the contract needs Alice to provide the right HK_x. If Alice has the right HK_x, the contract will determine that Bob is misbehaving because HK_x is fabricated. On the contrary, if Alice cannot give the right HK_x, she is determined to have cheated and given the wrong EHK_x at the beginning. The process then concludes.

   (3) Bob provides K_y which he used to decrypt AK_x. Then, the contract will calculate and check if $Hash(K_y)$ is one of the previous $Hash(K_i)$s that Bob submitted.
      a. If this is found to be true, the smart contract continues to execute step (4).
      b. Otherwise, Bob will be determine to have misbehaved and the adjudication concludes.

   (4) Check if K_y is one of DK_{ij}s that Alice uploaded previously.
      a. If this is found to be true, DK_{ij}s uploaded by Alice are correct and we assume the same one as DK_{py}, then jump to step (5).
      b. Otherwise, the smart contract will jump to step (6) to examine DK_{ij}s.

   (5) Check if $E_{K_y}(AK_x)$ or $E_{DK_{py}}(AK_x)$ is consistent with the previous value that Alice submitted.
      a. If this is found to be true, Alice is determined to be cheating because the AK_x given by Bob is indeed from this transaction, and the only reason for the error is that Alice used an AK_x in the exchanging process which differs from the one she claimed at the beginning. Hence, the process concludes.
      b. Otherwise, Bob is determined to be misbehaving because K_y and DK_{py} are valid and AK_x is fabricated by Bob. Hence, the process concludes.

   Note: We do not consider the situation that Alice uploaded the wrong DK_{ij}, but used the right $E_{DK_{ij}}(AK_k)$. We also do not consider if Alice uploads the right DK_{ij} but an incorrect $E_{DK_{ij}}(AK_k)$. We assume that Alice must use DK_{ij} which she has uploaded to encrypt AK_k.
   (6) Alice provides SK_p. Calculate and check if PK_p and SK_p submitted previously by Alice are a couple.
      a. If this is found to be true, the smart contract continues to execute step (7).
      b. Otherwise, we record that Alice is dishonest and the process ends.

   (7) Check if the value of $E_{PK_p}(K_y)$ is previous EK_y.

a. If is found to be true, the data exchanging process is determined to be normal, and neither Alice nor Bob has cheated. However, this is impossible and the process concludes.
b. Otherwise, we record Bob as misbehaving and the process concludes.

5.2. Analysis of privacy preservation

5.2.1. Roles of oblivious transfer protocol in privacy preservation

The oblivious transfer protocol is a kind of secure multi-party computation, and we applied it to our scheme according to its privacy preservation characteristic. Specifically, in our data trading scheme, the oblivious transfer protocol, as a key part of the data exchange system, protects the privacy of the buyer's and the seller's transaction information. In particular, the application of the oblivious transfer protocol prevents Alice from knowing which data blocks Bob computes, and Bob does not know any information about other Alice's data blocks that he does not compute. Simply, Alice does not know which data blocks Bob has chosen to purchase, and Bob does not know information about the remaining data he has not selected during the transaction.

5.2.2. A comparative summary for oblivious transfer protocol

We apply the m-out-of-n oblivious transfer protocol to achieve transactional fairness and privacy preservation. Compared with the classic 1-out-of-2 oblivious transfer protocol, the m-out-of-n oblivious transfer protocol has more complex calculations and slower speed, but it is more secure, and can increase fairness and purchase flexibility. Specifically, choosing one from two obviously has a high probability of correct guesses, and the privacy preservation strength is not sufficiently strong. However, guessing m from n $(n > 2, m > 1, n > m)$ is somewhat difficult. This will enhance the level of privacy preservation and security. Even when the buyer did not get the desired data after payment, the loss associated with the m-out-of-n oblivious transfer protocol is less than the classical oblivious transfer protocol, because using m-out-of-n oblivious transfer protocol one loses m/n money, and when using classical oblivious transfer protocol one loses half of the money.

5.3. Analysis of advantages of applying Ether cheque

Using Ether cheques to complete the payment of transactions increases the fairness of big data exchange, as well as making the trading more user-friendly.

1. The buyer completes the payment by uploading the cheque to the smart contract. The payee of the cheque, the amount, the time, and the signature of the payer can be verified by others. If any of the items are incorrect, Bob will have a dishonest record on the blockchain. This guarantees seller's interest is safeguarded.
2. The cheque system provides a buffer time for the transaction. That means Ether cheques will not take effect immediately at the end of transactions. During the buffer time, the seller must wait for the buyer to check the data commodity. If the buyer alleges any misbehavior, then the smart contract will review the trading process. Once the seller is confirmed to have cheated, the cheque will be revoked. This protects the consumer's interests.
3. The buyer does not need to send tokens to the smart contract every time (s)he trades.

5.4. Performance evaluation

5.4.1. Performance of the smart contract

Data trading requires a time limit. If too much time is used, the transaction efficiency of both parties will be reduced, and public resources will also have been excessively occupied.

We applied the smart contract to complete most of the transaction steps, therefore it is important to test the time consumption of key contract-based transaction steps of our scheme. And we achieved our trading smart contract in Solidity on *http://remix.ethereum.org/*, and the version of the compiler of our work is 0.4.25.

As online testing is greatly affected by network fluctuations, we will test the time consumption of several key steps for ten times and present all test results. And because some of the steps are similar, we only need to select a typical test object. According to the above analysis, we choose the following steps for testing:

- Alice sets $Deposit_A$ consumption (called step A).
- Alice sets $Block_{price}$ (called step B).
- Alice pays the amount of $Deposit_A$ to the smart contract (called step C).
- Bob sends PK_B to the contract (called step D).
- Bob sends $Cheque_{BA}$ to the contract (called step E).
- Alice sends the decrypted check to the contract (called step F).

Our test results for time consumption of key transaction steps on trading smart contracts are shown in the Table 2.

Table 2
Time consumption of key steps of the smart contract.

Time (s)	Test										
Step	1	2	3	4	5	6	7	8	9	10	avg.
A	0.74	0.72	0.63	0.71	0.65	0.63	0.68	0.73	0.60	0.70	0.679
B	0.62	0.55	0.53	0.50	0.56	0.51	0.53	0.51	0.60	0.48	0.539
C	0.68	0.66	0.71	0.70	0.68	0.72	0.69	0.75	0.78	0.76	0.713
D	0.86	0.75	0.85	0.80	0.84	0.83	0.91	0.83	0.78	0.80	0.825
E	0.81	0.84	0.83	0.86	0.78	0.85	0.76	0.80	0.80	0.76	0.809
F	0.98	0.90	0.91	0.97	0.89	0.92	0.95	0.91	0.93	0.95	0.931

From the data analysis provided in the table, we can see that using smart contracts for trading in our scheme is an efficient way because the average time consumption of each vital step is between 0.5 s and 1.0 s, which means our scheme works well in this situation of data transaction.

5.4.2. Performance of oblivious transfer

All keys used in the process (such as symmetric key, the private key of Alice, and the public key of Alice) are generated before the transaction. Therefore, when analyzing the performance of oblivious transfer protocol, we mainly test the time consumption related to key encryption is disregard time spent for key generation.

We use RSA algorithm to generate 1024-bit key pairs that serve as public keys and private keys for Alice and Bob. And we apply the random number generator to generate the 64-bit random character as the symmetric key. Furthermore, we utilize RSA algorithm to perform the asymmetric algorithm for encryption and decryption operations and use AES algorithm for the encryption and decryption operations of the symmetric algorithm.

During the execution of the transaction, there are four encryption or decryption operations related to oblivious transfer (see Section 4.3.3 for details). We take the sum of the time of these four encryption or decryption operations as the total time of a test, based on which we carry out the experiment of analysis.

Our m-out-of-n oblivious transfer protocol is divided into two parts, and we will introduce the experimental results of both parts as follows:

1. The change trend of total time when m value is fixed and n value increases. The result is shown in Fig. 6.
   (1) The fixed value of m is 10 ($n \geqslant 10$).
   (2) The fixed value of m is 30 ($n \geqslant 30$).
   (3) The fixed value of m is 50 ($n \geqslant 50$).

As can be seen from Fig. 6, when the value of m is fixed, the time consumption rises with the increase of n value, and the increase rate will gradually slow down.

The above analysis shows that when Bob chooses to buy m blocks of data in a transaction, as the value n of total data blocks increases, the time consumption of our scheme will increase. Therefore, when determine the total number of data blocks, Alice and Bob need to take time consumption into serious consideration.

2. The change trend of total time when n value is fixed and m value increases. The result is shown below in Fig. 7.
   (1) The fixed value of n is 20 ($m \leq 20$).
   (2) The fixed value of n is 60 ($m \leq 60$).
   (3) The fixed value of n is 100 ($m \leq 100$).

As shown in Fig. 7, we found that when the value of n is fixed, the time consumption increases tremendously as m increases, and the growth rate gradually decreases.

That is to say, in our transaction scheme, when n blocks of the data decided by Alice and Bob are fixed, and m blocks of data chosen by Bob increases, the consumption time of the scheme will increase. Thus, Bob needs to consider the trading time when choosing the suitable number m of data blocks to ensure transaction efficiency.

To sum up, the value of m and n will greatly affect the transaction time of our scheme. Therefore, Alice and Bob should set their transaction parameters – m and n carefully. Under the condition of using the proper value of n and m, our scheme can achieve the data exchange process well while protecting privacy. And we need to emphasize that if the participants would like to choose a large m or n value to protect their privacy, according to the experimental results, they can consider making multiple transactions with smaller value of m in each trading process, so as to improve the transaction efficiency.

5.4.3. Performance of IPFS

To demonstrate that IPFS can be applied to big data transactions, we evaluated the speed of uploading data to IPFS. We tested the files ranging in size from 100 MB to 5 GB, and recorded the time required in uploading of the data. In order to make the experiment more realistic, we used the control variates analysis method. Since the variate is the file size, all the files we

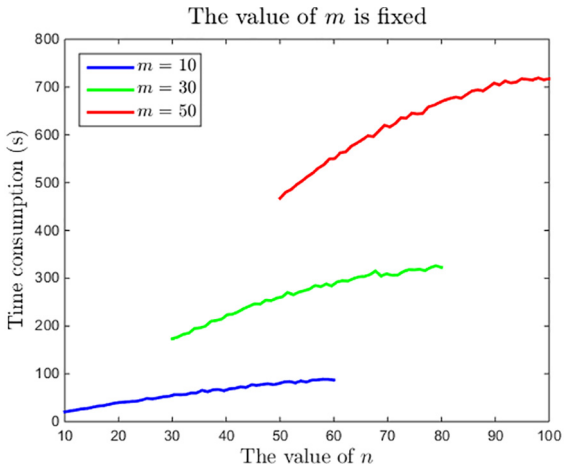

Fig. 6. The change trend of total time when m value is fixed.

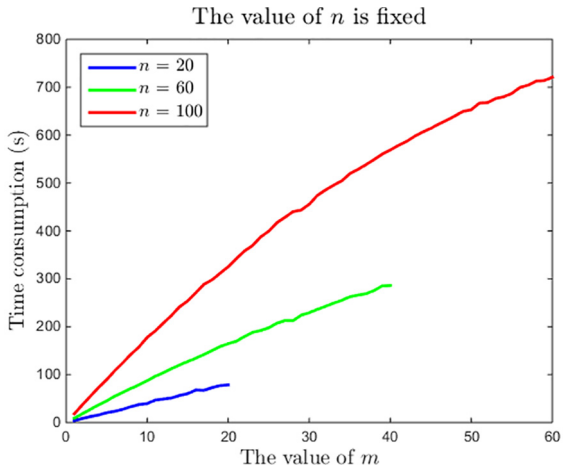

Fig. 7. The change trend of total time when n value is fixed.

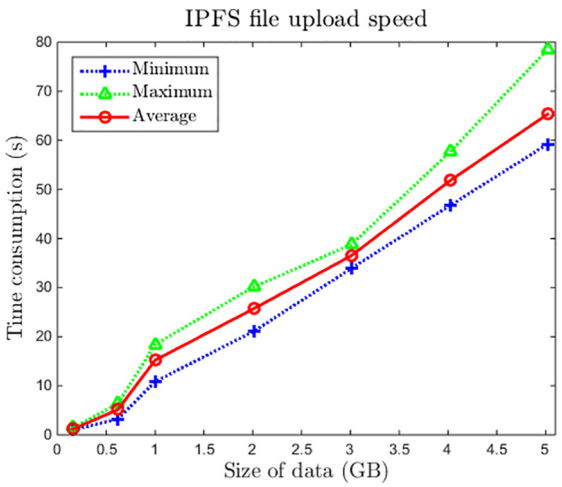

Fig. 8. Data Upload Speed of IPFS.

Table 3
Comparison with other related work.

Item \ Scheme		*FAPS*	*AAP*[2]	*SDM*[29]	*MPF*[30]
Fairness		✓	✗	✓	✗
Privacy preservation		✓	✗	✗	✓
Security	*fake products*	✓	✗	✓	✓
	default of payment	✓	✓	✓	✗
	abnormal termination	✓	✓	✗	✗
	buyer or seller is impersonated	✓	✗	✓	✗
Autonomy		✓	✗	✓	✗
Timing control		✓	✗	✗	✓
Efficiency		✓	✓	✓	✓

used are .rar compressed files. Moreover, after we completed the test of a file, we delete all files stored in the IPFS system. This excludes any potential impact of IPFS storage space on performance. However, we did not consider the impact of the number of folders in compressed files, which may affect the upload speed.

The times associated with uploading the different file sizes to IPFS are shown in Fig. 8.

We observe from the line chart that the upload time varies with the data size, and the average upload time of 5 GB data is 70 s. In general, uploading 1 GB data takes 16 s. The findings demonstrated that uploading data to IPFS is practical.

5.5. Comparison with other recent schemes

We mainly consider comparing our scheme *FAPS* with other recent related work by fairness, privacy preservation, security (see adversary model), autonomy, timing control.

Among the above six items, the former five are defined and analyzed in Section 3, and efficiency means the scheme can complete the transaction within the appropriate period.

We select three recent works [2,22,23], which are respectively denoted as *AAP*, *SDM*, and *MPF*, and the comparison is shown in Table 3.

6. Conclusion

In this paper, we proposed an Ether cheque system in which the buyers and the sellers use cheques rather than tokens in their transactions. We demonstrated how our proposed approach guarantees the private of buyers and sellers and the privacy in data transactions. In addition, we use smart contracts to avoid the need for a third-party entity. The approach includes both process control and time control for transactions, improving the automation and autonomy of data transactions. Moreover, based on the smart contract between two parties, the uploaded data can be used as evidence for handling any subsequent dispute or allegations. The experimental results showed that our scheme is feasible, practical, and has high efficiency in data transaction. Also, comparing to other recent works, our schemeachieves fairness, security, autonomy, timing control, and efficiency.

Future research includes implementing a prototype of the proposed system in collaboration with a real-world service provider. This will allow us to have a more effectively evaluate the real-world utility of our proposed system.

CRediT authorship contribution statement

Tiantian Li: Conceptualization, Methodology, Software, Validation, Investigation, Writing - original draft. **Wei Ren:** Conceptualization, Methodology, Formal analysis, Writing - original draft, Supervision, Project administration, Funding acquisition. **Yuexin Xiang:** Writing - original draft, Resources, Software. **Xianghan Zheng:** Resources, Funding acquisition. **Tianqing Zhu:** Conceptualization, Writing - review & editing. **Kim-Kwang Raymond Choo:** Conceptualization, Writing - review & editing. **Gautam Srivastava:** Writing - review & editing.

Declaration of Competing Interest

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

Acknowledgment

The research was financially supported by National Natural Science Foundation of China (No. 61972366), the Foundation of Key Laboratory of Network Assessment Technology, Chinese Academy of Sciences (No. KFKT2019-003), the Foundation of Guangxi Key Laboratory of Cryptography and Information Security (No. GCIS201913), and the Foundation of Guizhou Provincial Key Laboratory of Public Big Data (No. 2018BDKFJJ009, No. 2019BDKFJJ003, No. 2019BDKFJJ011). K.-K. R. Choo was supported in part by the Cloud Technology Endowed Professorship.

References

[1] X. Dong, R. Li, H. He, W. Zhou, Z. Xue, H. Wu, Secure sensitive data sharing on a big data platform, Tsinghua Science and Technology 20 (1) (2015) 72–80.
[2] T. Jung, X.-Y. Li, W. Huang, J. Qian, L. Chen, J. Han, J. Hou, C. Su, Accounttrade: Accountable protocols for big data trading against dishonest consumers, in: IEEE INFOCOM 2017-IEEE Conference on Computer Communications, IEEE, 2017, pp. 1–9.
[3] F. Liang, W. Yu, D. An, Q. Yang, X. Fu, W. Zhao, A survey on big data market: pricing, trading and protection, IEEE Access 6 (2018) 15132–15154.
[4] Q. Xia, E.B. Sifah, K.O. Asamoah, J. Gao, X. Du, M. Guizani, Medshare: Trust-less medical data sharing among cloud service providers via blockchain, IEEE Access 5 (2017) 14757–14767.
[5] S. Dziembowski, L. Eckey, S. Faust, Fairswap: How to fairly exchange digital goods, in: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2018, pp. 967–984.
[6] B. Aiello, Y. Ishai, O. Reingold, Priced oblivious transfer: How to sell digital goods, in: International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2001, pp. 119–135.
[7] D. Beaver, Precomputing oblivious transfer, in: Annual International Cryptology Conference, Springer, 1995, pp. 97–109.
[8] Y. Mu, J. Zhang, V. Varadharajan, m out of n oblivious transfer, in: Australasian Conference on Information Security and Privacy, Springer, 2002, pp. 395–405..
[9] M.O. Rabin, How to exchange secrets with oblivious transfer, IACR Cryptology ePrint Archive 2005 (2005) 187..
[10] C.-K. Chu, W.-G. Tzeng, Efficient k-out-of-n oblivious transfer schemes with adaptive and non-adaptive queries, in: International Workshop on Public Key Cryptography, Springer, 2005, pp. 172–183.
[11] S. Kak, Oblivious transfer protocol with verification, arXiv preprint arXiv:1504.00601..
[12] C. Liu, X. Zhang, C. Liu, Y. Yang, R. Ranjan, D. Georgakopoulos, J. Chen, An iterative hierarchical key exchange scheme for secure scheduling of big data applications in cloud computing, in: 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, IEEE, 2013, pp. 9–16.
[13] H. Cheng, C. Rong, K. Hwang, W. Wang, Y. Li, Secure big data storage and sharing scheme for cloud tenants, China Communications 12 (6) (2015) 106–115.
[14] A. Azaria, A. Ekblaw, T. Vieira, A. Lippman, Medrec: Using blockchain for medical data access and permission management, in: 2016 2nd International Conference on Open and Big Data (OBD), IEEE, 2016, pp. 25–30.
[15] H. Oh, S. Park, G.M. Lee, H. Heo, J.K. Choi, Personal data trading scheme for data brokers in iot data marketplaces, IEEE Access 7 (2019) 40120–40132.
[16] Y. Jiang, Y. Zhong, X. Ge, Smart contract-based data commodity transactions for industrial internet of things, IEEE Access 7 (2019) 180856–180866.
[17] G. Ishmaev, Blockchain technology as an institution of property, Metaphilosophy 48 (5) (2017) 666–686.
[18] E. Karafiloski, A. Mishev, Blockchain solutions for big data challenges: A literature review, in, in: IEEE EUROCON 2017–17th International Conference on Smart Technologies, IEEE, 2017, pp. 763–768.
[19] B. Liu, X.L. Yu, S. Chen, X. Xu, L. Zhu, Blockchain based data integrity service framework for iot data, in: 2017 IEEE International Conference on Web Services (ICWS), IEEE, 2017, pp. 468–475.
[20] Z. Zheng, S. Xie, H. Dai, X. Chen, H. Wang, An overview of blockchain technology: Architecture, consensus, and future trends, in: 2017 IEEE International Congress on Big Data (BigData Congress), IEEE, 2017, pp. 557–564..
[21] M. Yang, T. Zhu, K. Liang, W. Zhou, R.H. Deng, A blockchain-based location privacy-preserving crowdsensing system, Future Generation Computer Systems 94 (2019) 408–418.
[22] W. Xiong, L. Xiong, Smart contract based data trading mode using blockchain and machine learning, IEEE Access 7 (2019) 102331–102344.
[23] Y. Zhao, Y. Yu, Y. Li, G. Han, X. Du, Machine learning based privacy-preserving fair data trading in big data market, Information Sciences 478 (2019) 449–460.