Journal of Network and Computer Applications 176 (2021) 102953

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

A multi-type and decentralized data transaction scheme based on smart contracts and digital watermarks

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

[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

ARTICLE INFO

Keywords:
Data sharing
Data transaction
Smart contract
Digital watermarks
Blockchain
Smart city

ABSTRACT

While data sharing cross management is increasingly popular, there are challenges in such a decentralized data sharing mode, especially in terms of transaction fairness and copyright protection. However, few schemes consider both fairness and copyright protection in the current literature. In this paper, we propose a flexible and Ethereum-based scheme to facilitate copyright protection in data sharing blockchain, by using watermark-based policies. This allows us to achieve properties such as preventing the re-selling of data (e.g., partial data, combined data, and leaked data). We also design five algorithms to automatically guarantee data sharing fairness and proactively protect data copyright, by using smart contracts. In other words, several new security features are introduced to existing transaction schemes, while still achieving feasible performance. The response times for each key step of smart contract in the proposed scheme is between 0.41 s and 0.91 s.

1. Introduction

Edge computing is increasingly popular in a broad range of applications such as smart cities, partly due to the benefits over a pure cloud computing deployment, and advances in communication technologies that enable communications for significant data volume pervasively and ubiquitously. However, in existing systems, most data collected within a system are generally not shared with other systems/entities in the ecosystem. For example, data sensed and collected by the Internet of Things (IoT) devices within a smart building may not be shared with other systems in the smart city. In other words, most systems remain siloed, and consequently data are not shared due to separation of management domains.

There has been a trend of moving towards data sharing, particularly between systems and platforms, in order to maximize data value and facilitate informed decision making, for example using data sharing as a service platform (Dai et al., 2019). Existing data sharing markets can be either centralized or decentralized. In the former model, a centralized entity charges some admission fee or imposes commission on the transactions. While decentralized data transaction models may not have such limitations, there are also challenges in the design and implementation of decentralized data sharing platforms. For example, how do we ensure the fairness of data transactions, without requiring a centralized governance model? Also, how do we achieve copyright protection of the exchanged data such as defending against illegal/unauthorized re-sharing of sold data? In addition, the fairness in data transactions means that the participant who conducts a malicious transaction to affect the profit of the other participant should be penalized, and the participant who is victimized should potentially be compensated.

Hence, this necessitates the design of a platform that is self-governable. In recent years, many different approaches to ensure copyright protection on data transactions have been designed, such as those based on digital watermark, and blockchain. However, existing approaches generally do not allow one to handle disputes that occur during the transactions. In a recent work, Dai et al. (2019) introduced a blockchain-based data trading ecosystem, where neither the data broker nor the buyer can obtain access to the seller's raw data. Specifically, both data broker and buyer can only get access to the analysis findings that they require. The authors also presented a set of trading protocols for the data trading market.

* Corresponding author at: School of Computer Science, China University of Geosciences, Wuhan, PR China.
E-mail address: weirencs@cug.edu.cn (W. Ren).

https://doi.org/10.1016/j.jnca.2020.102953
Received 10 August 2019; Received in revised form 10 December 2020; Accepted 15 December 2020
Available online 19 December 2020

Along a similar line, in this paper we propose a decentralized scheme for flexible and self-governable data sharing. The scheme allows one to enforce proactive data copyright protection and automatically guarantee fair sharing, using watermarks and smart contracts respectively. In our scheme, the blockchain will also be used as a transaction account book, value exchange channel, and virtual database on watermarks. A summary of this paper's contributions is as follows:

1. We propose a flexible scheme for data copyright protection in data sharing by using watermark-based policies, e.g., minimizing the risk of re-selling of partial, combined, and/or leaked data.
2. We propose dedicated algorithms and protocols for automatically guaranteeing data sharing fairness and pro-actively protecting data copyright in data transactions by using smart contracts.

In the next section, we will review relevant prior work. The trade model and transaction process are presented in Section 3. Section 4 describes our scheme, and its evaluation and analysis are respectively presented in Sections 5 and 6. Finally, Section 7 concludes the paper.

2. Related work

We will now discuss recent literature related to secure data sharing and copyright protection.

2.1. Data sharing based on blockchain

There have been attempts to design blockchain-based secure data sharing, as previously discussed. For example, Liu (2016) explored the application of blockchain for data sharing in healthcare to record operations on the medical data and protect the integrity of sharing data. Karafiloski and Mishev (2017) surveyed existing big data solutions that could be empowered by the blockchain technology for personal data protection, digital property, Internet of things, and healthcare fields. Vo et al. (2018) presented a blockchain-based business application for micro-insurance and artificial intelligent (AI) marketplace. They also designed a blockchain-powered big data analysis platform. In a separate work, Guan et al. (2018) proposed a trusted big data collection and trading system using blockchain, with the aim of providing a fair and trusted platform for each participant. Specifically, the blockchain is applied to provide decentralized accounting and trading platform. In their scheme, sharing of private data transactions can be realized to a certain extent.

Nasonov et al. (2018) proposed a distributed big data platform, where a blockchain-based distributed digital data marketplace is used to ensure data transaction integrity. Bandara et al. (2018) proposed a blockchain based big data storage, built over Apache Cassandra distributed database. Their scheme can make big data structured and meaningful, and more secure at the same time. On this basis, it is easier for users to more effectively analyze big data. Zhang et al. (2019) explored the application of blockchain for the Internet of Vehicles (IoV), and more specifically in facilitating the vehicle to broadcast information and realizing trusted storage of information. In addition, the information broadcast and stored on the blockchain is authenticated and undeniable. Yang et al. (2019) presented a blockchain-based method to protect transaction records, by protecting sensitive personal information during data sharing, and also ensuring the efficiency of data sharing.

2.2. Smart contracts in data sharing

A number of smart contract-based schemes to facilitate big data sharing have been proposed, some of which have been implemented and used in practice. Bartoletti and Pompianu (2017), for example, studied how smart contract was interpreted in some of these platforms and analyzed the most common programming patterns in Ethereum. Their research work on the design patterns of these smart contracts helps other researchers make targeted improvements to these technologies related to smart contracts and blockchain. Mao et al. (2018) provided a blockchain-based credit evaluation system to strengthen the effectiveness of supervision and management in the food supply chain, which gathers credit evaluation text from traders by smart contracts on the blockchain. Zhuang et al. (2018) employed coded smart contract regulations to simulate several scenarios in healthcare processes. Various levels of data access privileges have also been designed to utilize a suite of customized smart contract settings. Wright and Serguieva (2017) proposed an efficient and secure automated management mechanism for smart contracts that is a method for the transfer of smart contracts underlying a hundred entities among disparate smart contracts or subcontracts. The scheme they proposed contributes to service delivery and enable resources to be accessed and allocated on a smaller scale. Gilcrest and Carvalho (2018) analyzed the value of smart contracts and blockchain as alternatives to traditional contractual obligations, as well as the advantages and disadvantages of replacing traditional contracts with these novel technologies. They also explored possible innovations in smart contracts and blockchain in the regulatory field, as well as how U.S. lawmakers are starting to deal with smart contracts and blockchain. Ali et al. (2018) proposed a secure data provenance framework for cloud-centric IoT networks by utilizing the immutable, deterministic and public nature of the blockchain smart contracts. They designed the framework in which the cryptographic hash of the device metadata is stored in the blockchain, while the actual data is stored off-chain. This approach makes the framework quite scalable in a network built with IoT devices. At the same time, in their design, they deploy multiple smart contracts in the blockchain to ensure the correctness of the source of the data stored in the cloud.

2.3. Copyright protection for big data

Copyright protection for big data is another topic of ongoing interest. For example, Agyekum et al. (2019) utilized a blockchain network, whose processing node acts as the proxy server that performs re-encryption on the data. To guarantee data confidentiality, the scheme stores some data on the blockchain and others on the cloud. The application of blockchain retains interaction records between entities and their solutions do not involve any trusted third party, while it also allows fine-grained access control. Liang et al. (2018) reviewed existing digital copyright protection mechanisms for big data markets, including digital copyright identification, digital rights management, digital encryption, watermarking, and others, as well as outlining challenges in data protection for the data trading lifecycle. In our big data era, the dissemination and transactions involving big data have also become more extensive, and data may involve secondary transactions without authorization from data sources. For example, Martin-Sanchez et al. (2017) identified common methodological challenges and reviewed relevant initiatives related to the re-use of patient data collected in routine clinical care, as well as analyzing the economic benefits derived from the secondary use of this data.

2.4. Digital watermarking for copyright protection

It is generally acknowledged in the literature that digital watermarking is an efficient solution to achieve copyright protection of big data. For example, Agrawal et al. (2003) demonstrated the potential to use watermarking database relations to deter data piracy. The advantage of their scheme is that there is no need to access the original data as well as the watermarking during watermarking detection. Moreover, the watermarking can be effectively maintained when inserting, updating, and deleting. And Lafaye et al. (2012) proposed a watermarking method for vectorial geographical databases especially on buildings. To protect the copyright of geographic data, their method can resist partial watermarking removal attacks, such as noise addition, cropping,

and over-watermarking. Also, Bhargava et al. (2012) presented a digital image authentication system (DIAS), designed to perform visible and invisible watermarking on images. DIAS contains two main parts, one for hiding information inside images, and the other for image information detection, that can be applied for both color images and gray images. Recently, a watermark based access control scheme for image big data was proposed by Guo et al. (2018) Instead of storing the access control strategies on the server, their scheme adds access control strategies to the image being accessed as the watermarking. The accessors in this model can view images without accessing the server, which also reduces the burden on the server and reduces access delay. The above works can be considered as possibilities or achievements of the applications of digital watermarking in protecting image big data copyright.

Differing from these existing literature, we focus on achieving more flexible copyright protection, which relies on various protection policies embedded in watermarks. Specifically, we propose proactive protection based on smart contracts, and our scheme is decentralized without relying on any trusted third party.

In the next section, we will describe the problem formulation.

3. Preliminary

3.1. System model

Data sharing can be conducted by any individual participator who is part of the data sharing community, for example as paying customers. In other words, any entity that installs the client software will join the community as a member. We thus do not assume the existence of a centralized trusted party and do not rely on any intermediate data markets. We also assume all data trading can be conducted online, and all payments are conducted in a pre-defined blockchain system, which could be a virtual autonomous community or organization for big data sharing.

Current classical blockchain systems, such as Bitcoin (BTC), Ethereum (ETH), and enterprise operation system (EOS), can be applied as the underlying blockchain infrastructure, in which portfolios of smart contract functions can be enabled. As the copyright of data must be protected, client software of the blockchain system will be extended by plug-in tools, which implement our proposed extra functions, such as watermark embedding, watermark extracting, and block retrieving.

We also assume that concrete watermarking methods are available and independent of the design of our scheme; thus, it is out of the scope of the paper. Specifically, in this paper we focus on the design of blockchain frameworks and protocols based on smart contracts using available watermarks.

Blockchain can be broadly considered a virtual list-based database with decentralized consensus. Policies regulating the further transaction privileges can be stored in the blockchain, and can be accessed by all participants. Watermarks will convey policies and be embedded in trading data. Those policies usually can be negotiated off-line in the community as system parameters, specified by various watermarks, such as re-selling enabled and re-selling disabled.

3.2. Adversary model

In this section, according to the real process of data trading, we identify six typical potential adversaries in decentralized data trading scenarios which are the most common attacks on data trading participants and the most threatening attacks on data trading systems. The specific definitions of these attacks are shown as follows:

Definition 1. Repeat Transaction Attack. Adversary intends to re-sell prior purchased data. *Specifically, the adversary sells the data purchased from a seller to another participant without authorization.*

Table 1
Notations.

Notations	Description
W	Watermark
M_A	The token amount of traded data configured by Alice
SD_A	The token amount of security deposit configured by Alice
$PriKey_B$	Private key of Bob
$PubKey_B$	Public key of Bob
K	Symmetric key of Alice
K'	Symmetric key of Alice encrypted by $Pubkey_B$
D	Big data of Alice
D'	Big data of Alice encrypted by K
MAC	Message digest for D' from Alice
MAC'	Message digest for D' from Bob
MAC_K	Message digest for K from Alice
MAC'_K	Message digest for K from Bob
$AddrPay$	Payment address configured by Alice
$AddrData$	Address of traded data configured by Alice
Rep_B	Report of Bob
TL_{trans}	Time limit of every steps during the transaction
TL_{rep}	Time limit of sending report after Bob received the key

Definition 2. Partial Transaction Attack. Adversary intends to re-sell partial prior purchased data. *Specifically, the adversary sells the partial data bought from a seller to another participant without the seller's authorization.*

Definition 3. Combination Transaction Attack. Adversary intends to combine multiple prior purchased data, partially or entirely. *Specifically, the adversary sells the combined data bought from different sellers to another participant without authorization.*

Definition 4. Leakage Transaction Attack. One intends to or makes mistakes to leak prior purchased data to others. *Specifically, the adversary leaks the purchased data to other people.*

In order to resist the above four attacks, our scheme uses robust watermarks that are embedded within data as identification, which are difficult to be tampered by attackers. In addition, transaction information and watermarking information are uploaded to the blockchain as a result of a transaction, and cannot be modified and can be easily traceable.

Definition 5. Transaction Fraud. One peer in the transaction is cheated by another peer, resulting in the loss of data or token. *Specifically, on the one hand, the buyer has paid tokens to the adversary, but the adversary does not send the data to the buyer or sends the incorrect data to the buyer, which means the loss of the token for the buyer; On the other hand, if the adversary has received the data from the seller, however, the adversary does not pay for the data, which means the loss of the data for the seller.*

Definition 6. Faked Transaction. The transaction information is faked. *Specifically, the adversary only intends to interfere with the normal transaction and actually the adversary does not want to trade with the seller. The adversary can affect the performance and availability of the entire system by initiating a large number of faked transactions.*

In order to resist the above two attacks, we regulate the transaction launching conditions by smart contracts pro-actively. In addition, transaction information and data watermarking information of both peers are stored in the smart contracts and executed automatically.

4. Proposed scheme

We will now present the key notations used in the remainder of the paper in Table 1.

To explain our proposed scheme, we assume there exist two entities called Alice and Bob where Alice has the data that Bob is interested in buying.

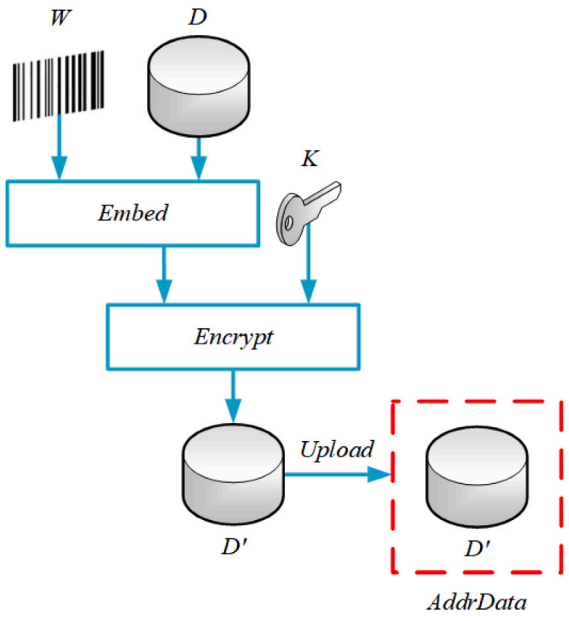


Fig. 1. Data encryption and uploading.

4.1. Design of smart contract templates

We leverage existing smart contracts and blockchain to perform two main functions: automatically responding to big data transactions and automatically responding to report information.

Also, we define the reward and punishment mechanism as follows:

Definition 7. Buyer's reward mechanism. Before a transaction, Alice will pay a token amount of security deposit SD_A to the smart contract. If Bob detects dishonest behaviors in big data D, he can send Rep_B to the smart contract with Alice's symmetric key K as evidence. After that, other miners can access $AddrData$ to verify Rep_B by K, D or W in D. If Rep_B is true, then the smart contract will refund and compensate Bob with SD_A that Alice paid.

Definition 8. Seller's punishment mechanism. Similar to the above situation, if other miners verify Rep_B as true, then the smart contract will send SD_A to Bob as compensation and refund Bob's prior payment (i.e., M_A).

Definition 9. Buyer's punishment mechanism. If other miners verify that Rep_B as false, then the smart contract will send M_A to Alice and refund SD_A to Alice.

To realize the data transaction, our smart contract will have the following three modules:

1. *Trading Module*. This module receives the parameters that Alice or Bob sends, and the module can provide different responses according to the parameters. And this module in the smart contract includes the algorithm for the trading process, which is shown in Algorithm 1.
2. *Verification Module*. This module shows the address of the public tool to generate hash values. The module provides MAC and MAC_K that Alice claimed, and Bob can verify K and D' by them. The specific process is shown in Algorithm 2.
3. *Report Module*. This module receives Rep_B, and Bob should upload K received from Alice as evidence. In this way, other miners can access $AddrData$, then verify whether Alice cheats Bob by K, D or W in D. Algorithm 3 describes this module.

Algorithm 1: Design of Trading Module

Input: M_A, SD_A, K, K', $PubKey_B$, TL_{trans}, Rep_B
Output: the Time Alice Completes process TAC, the Time Bob Completes process TBC, Trading Processing Result TPR

```
Alice sends SD_A to the smart contract, TAC = now;
if Bob sends M_A to the smart contract in (TAC + TL_trans) then
    Bob uploads PubKey_B to the smart contract, TBC = now;
    if Alice responds to Bob's requirement in (TBC + TL_trans) then
        if Alice confirms M_A = True and PubKey_B = True then
            Alice generates K' and uploads K' to the smart contract;
            Bob gets K by K' from the smart contract to check the data;
            if The data is correct then
                Bob ends the transaction;
                The smart contract sends M_A and SD_A to Alice;
                TPR = True;
            else
                Bob sends Rep_B to the smart contract;
                TPR = False;
        else
            Alice or Bob terminates the transaction;
            The smart contract refunds M_A to Bob and SD_A to Alice;
            TPR = False;
    else
        Alice or Bob terminates the transaction;
        The smart contract refunds M_A to Bob and SD_A to Alice;
        TPR = False;
else
    Alice ends the transaction;
    The smart contract refunds SD_A to Alice;
    TPR = False;
```

After Alice sets all parameters that need to be initialized in the smart contract, then Alice should deploy the smart contract designed above on the blockchain. Otherwise, the trading will not be accepted by others.

4.2. Initialization of smart contracts

All transactions need to adopt the above smart contract template, and Alice sets the parameters of the smart contract for trading.

First, Alice should generate a W and embed it into D. The method W generated is

$$W = Hash(D) \parallel Sig(Hash(D), PriKey_A),$$

where $Sig(\cdot)$ is digital signing algorithm, $PriKey_A$ is Alice's private key, $Hash(\cdot)$ is any cryptographic hash function, D is the big data of Alice, and W is the watermark which embeds into D is the mark of D generated by Alice.

Seller (e.g., Alice):

(1) Compute $D' \Leftarrow Enc(D, K)$, where D is the traded data; K is the symmetric key generated by Alice; $Enc(\cdot)$ is any symmetric encryption function; and D' is the encrypted traded data.

(2) Upload D' to $AddrData$, where D' is the encrypted traded data as in (1); $AddrData$ is the address of traded data configured by Alice.

(3) Compute $MAC \Leftarrow Hash(D')$, where MAC is the message digest for D' from Alice; $Hash(\cdot)$ is any cryptographic hash function; D' is the encrypted traded data as in (1).

Algorithm 2: Design of Verification Module

Input: M_A, SD_A, D, D', $AddrData$, MAC, MAC', MAC_K, MAC'_K, TL_{trans}, K, K', $PubKey_B$, Rep_B
Output: the Time Alice Completes process TAC, Verification Processing Result VPR

```
Alice generates D' and uploads D' to AddrData;
Alice uploads MAC, MAC_K and AddrData to the smart contract;
Alice sends SD_A to the smart contract, TAC = now;
if Bob decides to buy D then
    Bob downloads D' in AddrData and calculates its MAC';
    if MAC' = MAC then
        if Bob responds to the transaction in (TAC + TL_trans) then
            Bob sends M_A and uploads PubKey_B to the smart;
            Alice generates K' and uploads K' to the smart contract;
            Bob gets K by K' and calculates MAC'_K;
            if MAC'_K = MAC_K and W in D is correct then
                Bob completes the transaction;
                The smart contract sends SD_A and M_A to Alice;
                VPR = True;
            else
                Bob sends Rep_B to the smart contract;
                VPR = False;
        else
            Alice ends the transaction;
            The smart contract refunds SD_A to Alice;
            VPR = False;
    else
        Bob gives up the transaction;
else
    Bob gives up the transaction;
```

Algorithm 3: Design of Report Module

Input: W, M_A, SD_A, D, K, Rep_B, TL_{rep}
Output: the Time Alice Completes process TAC, the Judgment of Receiving the Report JRR, Report Processing Result RPR

```
Alice sends K' to the smart contract, TAC = now;
if Bob sends Rep_B to the smart contract in (TAC + TL_rep) then
    if The smart contract receives Rep_B and K then
        Another miner(not Alice or Bob) verifies Rep_B;
        JRR = True;
        if D is incorrect or K is incorrect or D has existed at least two W or a messy W is in D then
            RPR = True;
        else
            RPR = False;
    else
        Alice or Bob ends the transaction;
        The smart contract sends M_A and SD_A to Alice;
        JRR = False;
else
    Alice or Bob ends the transaction;
    The smart contract sends M_A and SD_A to Alice;
The smart contract judges the result of Rep_B;
if RPR = True then
    Bob ends the transaction;
    The smart contract sends M_A and SD_A to Bob;
else
    Alice or Bob ends the transaction;
    The smart contract sends M_A and SD_A to Alice;
```

(4) Compute $MAC_K \Leftarrow Hash(K)$, where MAC_K is the message digest for K from Alice; $Hash(\cdot)$ is any cryptographic hash function; K is the symmetric key generated by Alice as in (1).

(5) Upload MAC, MAC_K and $AddrData$ to the smart contract, where MAC is the message digest for D' from Alice; MAC_K is the message digest for K from Alice; $AddrData$ is the address of traded data configured by Alice.

(6) Send SD_A to the smart contract, where $SD_A > M_A$; M_A is the token amount in smart contract configured by Alice; SD_A is the token amount of security deposit in smart contract configured by Alice.

The above processes (1) and (2) are shown in Fig. 1 and the overall process is shown in Fig. 2.

We also remark that the inclusion of MAC_K allows one to verify that D Alice uploads to $AddrData$ is consistent with the data Bob downloads. There are two possible situations: 1) Alice uploads D which is not the digital commodity she claimed; 2) some attackers may tamper or destroy D.

Besides, we set TL_{trans} and TL_{rep} in the smart contract, whose function is to guarantee the interests of Alice and Bob and complete the transaction. For example, if Alice receives M_A Bob paid, without Alice actually sending K to Bob, the smart contract will end the transaction and refund M_A to Bob after TL_{trans} runs out. Also, if the transaction process has been completed, and Bob does not send Rep_B, after TL_{trans} exhausting, Alice can complete the transaction. In addition, the smart contract will send M_A and SD_A to Alice.

4.3. Transaction response

The buyer who wants to buy D should first get the $AddrData$ in the smart contract.

Buyer (e.g., Bob):

(1) Download D' from $AddrData$, where D' is the encrypted traded data; $AddrData$ is the address of traded data configured by Alice.

(2) Get MAC and MAC_K from the smart contract, where MAC is the message digest for D' from Alice; MAC_K is the message digest for K from Alice.

(3) Compute $MAC' \Leftarrow Hash(D')$, where MAC' is the message digest for D' from Bob; $Hash(\cdot)$ is any cryptographic hash function; D' is encrypted traded data as in (1).

(4) Check if $MAC' = MAC$.

On the one hand, if Bob finds MAC' and MAC in the smart contract are equal, Bob pays M_A to the smart contract, and then uploads $PubKey_B$ to the smart contract. This process is shown in Fig. 3. On the other hand, if Bob finds that MAC' and MAC are not equal, Bob does not pay and gives up the transaction.

4.4. Transaction execution

Next, the process of transferring keys between the seller and the buyer is described.

Seller (e.g., Alice):

(1) Verify M_A in the smart contract, if M_A is correct then downloading $PubKey_B$, where M_A is the token amount in the smart contract configured by Alice; $PubKey_B$ is the public key of Bob.

(2) Encrypt K as the way of $K' \Leftarrow Enc(K, PubKey_B)$, where $Enc(\cdot)$ is an algorithm for encrypting keys; K is the symmetric key of Alice; $PubKey_B$ is the public key of Bob as in (1).

(3) Upload K' to the smart contract, where K' is the symmetric key of Alice which is encrypted by algorithms $Enc(\cdot)$ through K and $PubKey_B$ as in (2).

Buyer (e.g., Bob):

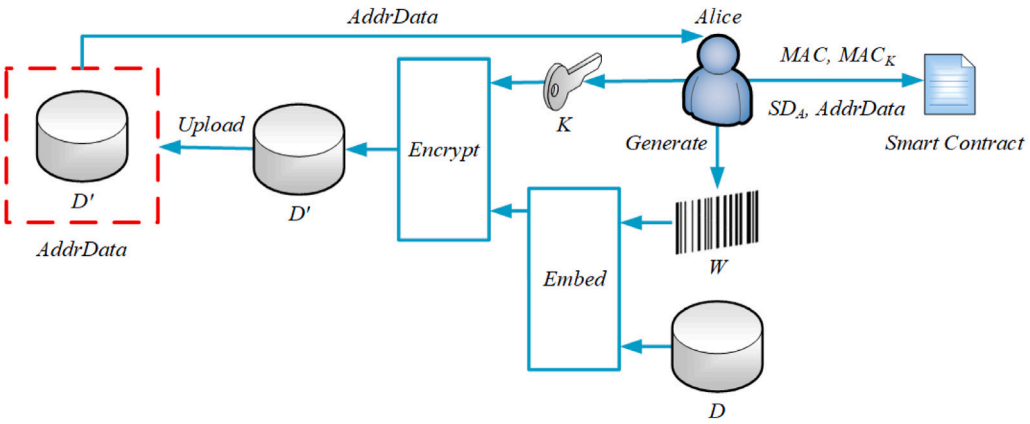


Fig. 2. Initialization of the smart contract.

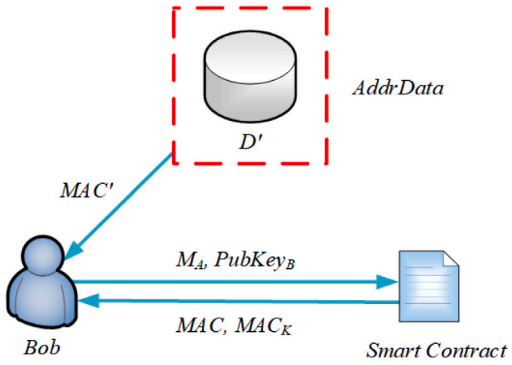


Fig. 3. Transaction response process.

(1) Obtain K', where K' is the symmetric key of Alice which is encrypted by algorithms $Enc(\cdot)$ through K and $PubKey_B$.

(2) Decrypt as the way of $K \Leftarrow Dec(K', PribKey_B)$, where K is the symmetric key of Alice; $Dec(\cdot)$ is the decryption key algorithm corresponding to $Enc(\cdot)$; K' is the symmetric key of Alice which is encrypted by algorithms $Enc(\cdot)$ through K and $PubKey_B$; $PribKey_B$ is the private key of Bob.

(3) Decrypt D' by K, where D' is the big data of Alice encrypted by K; K is the symmetric key of Alice as in (2).

(4) Try to detect and extract W in D, where D is the big data of Alice; W is the watermark.

This process is shown in Fig. 4.

4.5. Transaction monitoring

If Bob finds that the extracted W is problematic, then it could be one of these two cases: 1) K provided by Alice is incorrect and this lead to a messy W; 2) Bob finds two or more W in the D, which indicates that the big data has been traded twice or more.

Buyer (e.g., Bob):

(1) Find the W in the D is incorrect or has some problems, where W is the watermark; D is the big data of Alice.

(2) Send Rep_B to the smart contract, where Rep_B is the report of Bob.

Verifier (e.g., Cindy):

(1) Check Rep_B, where Rep_B is the report of Bob.

(2) Send the result of the Rep_B to the smart contract, where Rep_B is the report of Bob as in (1):

- If Rep_B is verified correct by Cindy, then the smart contract will refund M_A to Bob and compensate for Bob by SD_A.
- If Rep_B is verified incorrect by Cindy, the smart contract will send M_A and SD_A to Alice.
- If the verifier does not respond to Rep_B, when TL_{rep} is run out, both Alice and Bob can end the transaction and the smart contract will refund M_A to Bob and SD_A to Alice.

Also, miners can be rewarded for participating in the validation, which can be easily achieved in a smart contract. For example, if Bob's report is correct, the miners participating in the verification can get 5% of SD_A as a reward for the verification. If Bob's report is incorrect, the miners participating in the verification can get 5% of M_A as a reward for the verification. This may serve as a motivation for invoking miners to participate in the validation. In addition, this is an anonymous network and we set a time limit for verification reports. Once the time limit is exceeded, the transaction will fail, and the smart contract will return M_A to Bob and SD_A to Alice. Also, the validation message is broadcast. Since there is a reward for validation, miners will try to participate. In summary, it is very difficult to collude with a dishonest buyer in our proposed system.

4.6. Enhancement of flexibility

We will now describe how we improve the flexibility of our scheme, by carrying different trading strategies in watermarks for different types of transaction authorizations.

Different types of transactions can be specified by various types of watermarks. For example, data embedded with watermark $type_A$ can be traded multiple times, but data embedded with watermark $type_B$ can be traded only once (i.e., re-selling disallowed). Data embedded with watermark $type_C$ can only be traded at specific time-span, and data embedded with watermark $type_D$ can only be traded after N days, e.g., $N = 30$ (this is also referred to as embargo).

Different types of watermarks correspond to different strategies. Data users can embed different watermarks into the data according to different needs. In addition, verifiers can also determine whether the transaction is legitimate according to the type of watermarks.

It is also necessary to initialize the authorization corresponding to the watermark type in the smart contract, which expands the function of the smart contract and makes the scheme we proposed more flexible.

In order to add new functions for our scheme, we first propose a method for generated watermarks with type information. The approach of generating watermarks is

$$W = Hash(D)||Sig(Hash(D)||Type, PriKey_A),$$

where $Sig(\cdot)$ is digital signing algorithm; $PriKey_A$ is Alice's private key; $Hash(\cdot)$ is any cryptographic hash function; D is the big data of Alice; W is the watermark which embeds into D is the mark of D generated by

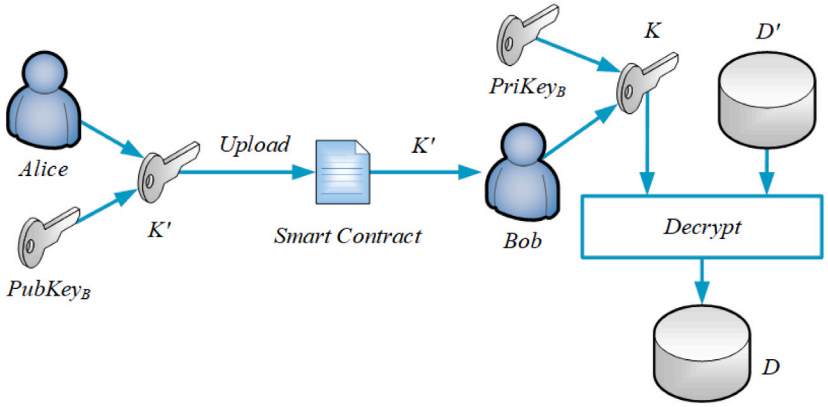


Fig. 4. Transaction execution process.

Alice; $Type$ is the type information corresponding to specific transaction strategies;

We assume that there are N types of watermarks, where $N = 2$, for example, $type_A$ and $type_B$. N can also take on other values greater than 2, but an overlarge value of N is likely to result in a decrease in trading efficiency according to the transaction process of our scheme. Algorithm 4 describes the improved method.

Algorithm 4: Process of Sorting Watermarks by Types

Input: W, D, public key of Alice $PubKey_A$, Rep_B, watermarks — strategies information $Inf_{water-stra}$
Output: Sorting Processing Result SPR

Bob extracts W in D;
Bob obtains type information of W by $PubKey_A$;
Bob gets $Inf_{water-stra}$ from the smart contract;
if $Type = Type_A$ **then**
  Bob finds the strategy corresponding to $Type_A$ in $Inf_{water-stra}$;
  if *The transaction satisfies the strategy corresponding to* $Type_A$ **then**
    $SPR = True$;
  else
    Bob sends Rep_B to the smart contract;
    $SPR = False$;
else if $Type = Type_B$ **then**
  Bob finds the strategy corresponding to $Type_B$ in $Inf_{water-stra}$;
  if *The transaction satisfies the strategy corresponding to* $Type_B$ **then**
    $SPR = True$;
  else
    Bob sends Rep_B to the smart contract;
    $SPR = False$;
else
  The process of matching ends;
  $SPR = False$;

5. Discussion

5.1. Novel watermarks

In addition, we consider a method to enhance our proposed scheme. It may be possible to detect and extract watermarks in encrypted big data.

Due to the special construction of big data, we can use whitespace and other methods to construct the watermark. When encrypting big data, it is possible to encrypt readable content without encrypting watermark information such as whitespace. The result is that the available content is encrypted, but the watermark is not. Thus, buyers can verify the watermarks in the data without the need to first decrypt it.

5.2. Improvement of report module

In our scheme, if one buyer (e.g., Bob) sends the incorrect report, then Bob loses M_A he pays to the smart contract but he still obtains the data. Such a penalty is too light. Therefore, to improve our scheme and prevent malicious reports, we consider adding the following step before Bob sends the report.

- If Bob wants to send a report to the smart contract, then he should pay the security deposit with a value that is commensurate with the value of the data to the smart contract first. Only upon the successful payment of such a deposit will the smart contract accept his report.

Algorithm 5 describes this module.

5.3. Reliable methods to verify reports

The scheme we propose only lets one miner (not Alice or Bob) verify Rep_B and sends the result to the smart as the final result. In real-world trading situations, it may not be appropriate to verify Rep_B in such a manner, because that one miner has too much power. Therefore, to solve the problem, we present three effective rules to enhance our scheme:

1. *Only miners who have successfully traded a certain number of times can verify the report.* A reputable miner is defined to be one who has completed many transactions and has a good reputation (since false/dishonest verification will impact on the miner's reputation). To select the miners of good repute, we can trace logs and collect the trading information to analyze their reputation, and then design an algorithm to achieve the goal. For example, it is not hard to find a miner who had successfully completed at least X number of transactions in the recent Y number of days.
2. *Set the minimum number of miners involved.* If not only one miner is involved in the verification, we can set a number N that the report should be verified by at least N miners, or the verification is invalid. At the same time, we should make rules for judging the result of the verification finished by N miners.

Algorithm 5: Improvement of Report Module

Input: M_A, SD_A, Security Deposit of Bob SD_B, K, K', Rep_B, TL_{rep}
Output: the Time Alice Completes process *TAC*, the Judgment of Receiving the Report *JRR*, Report Processing Result *RPR*

```
Alice sends K' to the smart contract, TAC = now;
if Bob sends SD_B to the smart contract then
    if Bob sends Rep_B to the smart contract in (TAC + TL_rep) then
        if The smart contract receives Rep_B and K then
            Another miner(not Alice or Bob) verifies Rep_B, JRR = True;
            if The miner finds a certain kind of error then
                RPR = True;
            else
                RPR = False;
        else
            Alice or Bob ends the transaction;
            The smart contract sends M_A and SD_A to Alice and SD_B to Bob, JRR = False;
    else
        Alice or Bob ends the transaction;
        The smart contract sends M_A and SD_A to Alice and SD_B to Bob;
else
    The smart contract reminds Bob to send SD_B first;
The smart contract judges the result of Rep_B;
if RPR = True then
    Bob ends the transaction;
    The smart contract sends M_A, SD_A and SD_B to Bob;
else
    Alice or Bob ends the transaction;
    The smart contract sends M_A, SD_A and SD_B to Alice;
```

3. *Set threshold values for verification.* In order to solve the problem of judging the verification above, we consider that it is necessary to set threshold values. More specifically, for example, we can set a threshold value for the smart contract judging the result. We assume the threshold value is 75% and 50%. That means if there are at least 75% miners who join the verification agree the report is correct, then the smart contract can judge that the report is correct, and send all tokens to the buyer. But if the proportion is between 75% and 50%, the transaction is controversial, and the smart contract refunds tokens to both the buyer and the seller. And if the proportion is below 50%, the smart contract will believe the report is incorrect, thus it sends all tokens to the seller.

6. Evaluation and analysis

6.1. Experiment setting

We use *Remix*, an online editor to compile and run the code.[1] And we refer interested readers to see our work on GitHub[2] for the specific method and examples to use our scheme. Next, we will analyze the security and performance of our scheme.

[1] https://remix.ethereum.org.
[2] https://github.com/Y-Xiang-hub/A-Copyright-Protection-Method-in-Big-Data-Trade.

6.2. Security analysis

We choose three typical and recent related work, namely: BDCP (Yang et al., 2018), TBDCT model (Guan et al., 2018) and SDTE (Dai et al., 2019), and compare them with our proposed scheme — see Table 2.

Proposition 1. *Our scheme can defend against repeat transaction attacks.*

Proof. When the seller sells whole unauthorized big data (so-called repeat transaction) to others, the specific process to prevent the repeat transaction attacks is shown as the following steps:

1. *The buyer receives D and extracts W in D.* D is the big data of the seller and W is the watermark.
2. *The buyer finds there is one and only one W in D.* The buyer continues to verify the signature of W:
   (a) *The signature of W is correct.* The transaction is successful, and the buyer can end the transaction. Then the smart contract will send M_A to the seller and refund SD_A to the seller.
   (b) *The signature of W is incorrect.* That means the seller sells unauthorized data, and the buyer can report the seller to the smart contract. After the verification of the Rep_B, the smart contract will send SD_A to the buyer and refund M_A to the buyer.
3. *The buyer finds at least two W in D.* The big data sold by the seller must be unauthorized, and the buyer can report the seller to the smart contract. After the verification of Rep_B, the smart contract will send SD_A to the buyer and refund M_A to the buyer.
4. *The buyer finds no W in D or unrecognized W in D.* In our scheme, the seller should embed the watermarking before the transaction starts and ensures watermarking integrity. Therefore, it is the seller that breaks the rule of the transaction, and the buyer can report the seller to the smart contract. After the verification of Rep_B, the smart contract will send SD_A to the buyer and refund M_A to the buyer.

Driven by the rewards raised by the smart contract according to the different situations, buyers always choose to report repeat transaction behaviors for maximizing the interests. Therefore, sellers will not gain benefits from repeat transactions, which also means repeat transaction attacks will never happen. □

Proposition 2. *Our scheme can defend against partial transaction attacks.*

Proof. When the seller divides the data that has been previously purchased from another participant into several pieces and partially trades them, the buyer can still extract W in the partial data since we use robust watermarking. Thus, the partial transaction attack is a special kind of repeat transaction attacks, and the way to prove that it will never happen is the same as Proposition 1. □

Proposition 3. *Our scheme can defend against combination transaction attacks.*

Proof. When the seller entirely or partially combines two or more pieces of different big data that the seller has bought from other participants previously together to sell, the buyer can still extract W in the combined data because of the robustness of watermarking we use. Therefore, the combination transaction attack is also a special type of repeat transaction attacks, and the method to prove that it will never happen is also the same as Proposition 1. □

Proposition 4. *Our scheme can defend against leakage transaction attacks.*

Table 2
Comparison of transaction schemes' capabilities against different types of attacks.

	BDCP (Yang et al., 2018)	TBDCT (Guan et al., 2018)	SDTE (Dai et al., 2019)	Our scheme
Repeat transaction attack	×	×	×	✓
Partial transaction attack	×	×	×	✓
Combination transaction attack	×	×	×	✓
Leakage transaction attack	✓	✓	✓	✓
Transaction Fraud	×	✓	✓	✓
Faked transaction	×	✓	✓	✓

Proof. When the buyer trades the obtained big data in the black market, due to the application of the robust watermarking, leakage transaction attacks can be prevented as below:

1. *Other participants in the black market want to buy the data sold by the buyer.* The watermarking can be used as legal evidence, therefore they are subject to legal risk when they are found owning unauthorized data and they will be sentenced and should pay a large amount of money that is quite higher than the original price of the data for their wrongful trading practices.
2. *Other participants in the black market want to re-sell the data sold by the buyer.* It is hard to implement in our system according to the rules we set, and it is also difficult to trade it in the black market for the high legal risk as mentioned above.

Taking the high legal risk for modest profits, the participants will not leak the data to the black market. □

Proposition 5. *Our scheme can defend against transaction fraud.*

Proof. When the seller or the buyer cheats the other party in the transaction, the designed smart contract and rewards and punishment mechanism in the system can be used to resist transaction fraud as follows:

1. *The buyer cheats the seller.* This will never happen, because the transaction will be activated only when the buyer has paid M_A, then:
   (a) *The buyer receives K' but (s)he does not confirm before the time limit.* The smart contract will send M_A to the seller and refund SD_A to the seller.
   (b) *The buyer sends Rep_B to the smart contract for cheating the seller.* If the result of verification on the smart contract shows that Rep_B is incorrect, the smart contract will send M_A to the seller and refund SD_A to the seller.
2. *The seller cheats the buyer.* If the seller does not upload the required parameters to the smart contract before the time limit in any step, the buyer can terminate the transaction, and then the smart contract will refund SD_A to the seller and M_A to the buyer.

As shown above, the rewards and punishment mechanism and the process controlled by the smart contract can ensure that both parties cannot implement transaction fraud during the transaction. □

Proposition 6. *Our scheme can defend against the faked transaction.*

Proof. When any party implements the faked transaction, the methods to prevent such attacks is shown below:

1. *The buyer implements the faked transaction.* The seller sets the amount of the deposit for the buyer as SD_B and after the buyer pays M_A and SD_B to the smart contract, the transaction starts. If the buyer does not continue the transaction in any step of the transaction, the smart contract will refund M_A and SD_B to the buyer and SD_A to the seller after reaching the time limit of the transaction.

Table 3
Performance analysis on response time.

	$Time_1$	$Time_2$	$Time_3$	$Time_4$	$Time_5$	Average
Deploy smart contract	0.73	0.83	0.91	0.85	0.88	0.840
Set amount	0.45	0.50	0.46	0.48	0.48	0.474
Upload hash of data	0.63	0.53	0.61	0.66	0.56	0.598
Send security deposit	0.46	0.50	0.55	0.45	0.41	0.474
Upload public key	0.81	0.80	0.75	0.82	0.76	0.788
End transaction	0.58	0.61	0.55	0.63	0.57	0.588

2. *The seller implements the faked transaction.* The seller should upload the data and calculate the hash values of each essential component in the transaction. The seller should also set the parameters for the smart contract initialization. Finally, the seller should send SD_A to the smart contract to start waiting for the transaction.

On the basis of the above explanation, we can see that any party that launches the faked transaction will waste himself/herself a lot of time, tokens and the computing resources, but it only interferes with one transaction in the system. That also means if participants aim to attack the performance and availability of the system through launching a large number of faked transactions, they must take the huge consumption into consideration. Therefore, due to the enormous spending with little rewards, the faked transaction will never happen. □

In summary, digital watermarking mainly serves as protection of data copyright. The smart contract and reward and punishment mechanism are mainly used to realize automatic transaction and supervision, as well as to prevent transaction fraud and faked transactions.

6.3. Performance analysis

The proposed scheme does not rely on the third trust party, which can be applied in current edge computing scenarios and break individual security management constraints by self-governance. Also, it can avoid the burden of the centralized data market. The scalability of this architecture is better.

For decentralized self-trading, we propose a fair mechanism based on the smart contract, which can be executed at the client, which further improves the scalability. The proposed scheme is compatible with current mainstream blockchain infrastructure, e.g., Ethereum. The extra watermark enabling modules can be installed or as a plug-in with current blockchain client, thus the deployment is easy and portability can be guaranteed.

Since the transactions are time-sensitive, if each step of the transaction takes too long, it will have a negative impact on the entire trading system. Therefore, we test the response time of some vital steps of the smart contract.

In order to reduce the error, we tested multiple times and took the average of the response time of each important step as the result. The compiler version for testing is 0.4.24+commit, and the test environment is JavaScript VM. The result is shown in Table 3. And the unit of the number in Table 3 is seconds.

From Table 3, we can see that the longest response time is 0.91 s, the shortest response time is 0.41 s, and the average response time of each

step is less than 1 s. Also, the data fluctuation is small, indicating that the stability of the system is strong, for the maximum and minimum differences in time are respectively 0.18 s and 0.05 s in the same set of data in the table. In summary, our solution can meet the transaction time requirements.

7. Conclusions

As the value of data becomes increasingly important, there is a need to secure such valuable data, particularly those that are intangible (e.g., intellectual property). Hence, in this paper we proposed a scheme to prevent unauthorized re-selling of big data. Specifically, our scheme uses smart contracts and watermarks, where we designed the smart contract template and adopted robust watermarks for the data trading. By using smart contracts to automate the entire process, the efficiency is significantly improved. Our proposed scheme can also resist a broader range of attacks compared to other recent approaches. In addition, the smart contract can be used to monitor transactions, and the reward and punishment mechanism can guarantee the fairness of buyers by reporting illegal transactions.

Future research includes implementing a refined prototype of our proposed approach in a commercial setting, for implementation in collaboration with a real-world service provider. This will allow us to more accurately evaluate the real-world utility of such an approach. In addition, some proofs will also be further improved in the future version.

CRediT authorship contribution statement

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

Acknowledgments

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, PR China (No. GCIS201913), and the Foundation of Guizhou Provincial Key Laboratory of Public Big Data, PR China (No. 2018BDKFJJ009, No. 2019BDKFJJ003, No. 2019BDKFJJ011). K.-K. R. Choo was funded only by the Cloud Technology Endowed Professorship.

References

Agrawal, R., Haas, P.J., Kiernan, J., 2003. Watermarking relational data: framework, algorithms and analysis. VLDB J. 12 (2), 157–169.

Agyekum, O., Opuni-Boachie, K., Xia, Q., Sifah, E.B., Gao, J., Xia, H., Du, X., Guizani, M., 2019. A secured proxy-based data sharing module in IoT environments using blockchain. Sensors 19 (5), 1235.

Ali, S., Wang, G., Bhuiyan, M.Z.A., Jiang, H., 2018. Secure data provenance in cloud-centric internet of things via blockchain smart contracts. In: 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI). IEEE, Guangzhou, China, pp. 991–998.

Bandara, E., Ng, W.K., De Zoysa, K., Fernando, N., Tharaka, S., Maurakirinathan, P., Jayasuriya, N., 2018. Mystiko—blockchain meets big data. In: 2018 IEEE International Conference on Big Data (Big Data). IEEE, Seattle, USA, pp. 3024–3032.

Bartoletti, M., Pompianu, L., 2017. An empirical analysis of smart contracts: platforms, applications, and design patterns. In: International Conference on Financial Cryptography and Data Security. Springer, Sliema, Malta, pp. 494–509.

Bhargava, N., Sharma, M., Garhwal, A.S., Mathuria, M., 2012. Digital image authentication system based on digital watermarking. In: 2012 International Conference on Radar, Communication and Computing (ICRCC). IEEE, Tiruvannamalai, India, pp. 185–189.

Dai, W., Dai, C., Choo, K.-K.R., Cui, C., Zou, D., Jin, H., 2019. Sdte: A secure blockchain-based data trading ecosystem. IEEE Trans. Inf. Forensics Secur. 15, 725–737.

Gilcrest, J., Carvalho, A., 2018. Smart contracts: Legal considerations. In: 2018 IEEE International Conference on Big Data (Big Data). IEEE, Seattle, USA, pp. 3277–3281.

Guan, Z., Zhao, Y., Li, D., Liu, J., 2018. Tbdct: A framework of trusted big data collection and trade system based on blockchain and tsm. In: 2018 IEEE International Conference on Smart Cloud (SmartCloud). IEEE, New York, USA, pp. 77–83.

Guo, J., Ren, W., Ren, Y., Zhu, T., 2018. A watermark-based in-situ access control model for image big data. Future Internet 10 (8), 69.

Karafiloski, E., Mishev, A., 2017. Blockchain solutions for big data challenges: A literature review. In: IEEE EUROCON 2017-17th International Conference on Smart Technologies. IEEE, Ohrid, Macedonia, pp. 763–768.

Lafaye, J., Béguec, J., Gross-Amblard, D., Ruas, A., 2012. Blind and squaring-resistant watermarking of vectorial building layers. GeoInformatica 16 (2), 245–279.

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

Liu, P.T.S., 2016. Medical record system using blockchain, big data and tokenization. In: International Conference on Information and Communications Security. Springer, Singapore, Singapore, pp. 254–261.

Mao, D., Wang, F., Hao, Z., Li, H., 2018. Credit evaluation system based on blockchain for multiple stakeholders in the food supply chain. Int. J. Environ. Res. Public Health 15 (8), 1627.

Martin-Sanchez, F., Aguiar-Pulido, V., Lopez-Campos, G., Peek, N., Sacchi, L., 2017. Secondary use and analysis of big data collected for patient care: Contribution from the IMIA working group on data mining and big data analytics. Yearb. Med. Inform. 26 (1), 28.

Nasonov, D., Visheratin, A.A., Boukhanovsky, A., 2018. Blockchain-based transaction integrity in distributed big data marketplace. In: International Conference on Computational Science. Springer, Wuxi, China, pp. 569–577.

Vo, H.T., Mohania, M., Verma, D., Mehedy, L., 2018. Blockchain-powered big data analytics platform. In: International Conference on Big Data Analytics. Springer, Warangal, India, pp. 15–32.

Wright, C., Serguieva, A., 2017. Sustainable blockchain-enabled services: Smart contracts. In: 2017 IEEE International Conference on Big Data (Big Data). IEEE, Boston, USA, pp. 4255–4264.

Yang, J., Wang, H., Wang, Z., Long, J., Du, B., 2018. Bdcp: A framework for big data copyright protection based on digital watermarking. In: International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage. Springer, Melbourne, Australia, pp. 351–360.

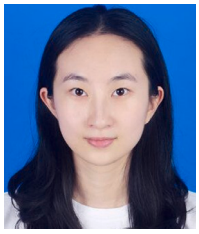
Yang, M., Zhu, T., Liang, K., Zhou, W., Deng, R.H., 2019. A blockchain-based location privacy-preserving crowdsensing system. Future Gener. Comput. Syst. 94, 408–418.

Zhang, L., Luo, M., Li, J., Au, M.H., Choo, K.-K.R., Chen, T., Tian, S., 2019. Blockchain based secure data sharing system for internet of vehicles: A position paper. Veh. Commun. 16, 85–93.

Zhuang, Y., Sheets, L., Shae, Z., Tsai, J.J., Shyu, C.-R., 2018. Applying blockchain technology for health information exchange and persistent monitoring for clinical trials. In: AMIA Annual Symposium Proceedings, Vol. 2018. American Medical Informatics Association, San Francisco, USA, p. 1167.

Yuexin Xiang received the B.Eng. degree in information security from School of Computer Science, China University of Geosciences, China. He is currently pursuing the M.Eng degree with the School of Computer Science, China University of Geosciences, China. His current research interests include blockchain, smart contract, and AI security.

Wei Ren currently is a full Professor at the School of Computer Science, China University of Geosciences, Wuhan, China. He was with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, USA in 2007 and 2008, the School of Computer Science, University of Nevada Las Vegas, USA in 2006 and 2007, and the Department of Computer Science, The Hong Kong University of Science and Technology, in 2004 and 2005. He obtained his Ph.D. degree in Computer Science from Huazhong University of Science and Technology, China. He has published more than 70 refereed papers, 1 monograph, and 4 textbooks. He has obtained 10 patents and 5 innovation awards. He is a distinguished member of the China Computer Federation and a member of IEEE.



Tiantian Li received the B.Eng. degree in information security from School of Computer Science, China University of Geosciences, China. She is currently pursuing the M.Eng degree with the Melbourne School of Engineering, The University of Melbourne, Australia. Her current research interests include blockchain, artificial intelligence, and cyber security.

Xianghan Zheng received a Ph.D. of Information Communication Technology (2011) from University of Agder, Norway. He is currently a professor in the College of Mathematics and Computer Sciences, Fuzhou University, China. He is also one of the founders of Mingbyte Technology, Qingdao, China. His current research interests include new generation network with special focus on cloud computing services, big data, blockchain and security.

Tianqing Zhu received her B.Eng. and M.Eng. degrees from Wuhan University, China, in 2000 and 2004, respectively, and a Ph.D. degree from Deakin University in Computer Science, Australia, in 2014. Dr. Tianqing Zhu is currently a senior lecturer at the School of Software in University of Technology Sydney, Australia. Before that, she was a lecturer at the School of Information Technology, Deakin University, Australia, from 2014 to 2018. Her research interests include privacy preservation, data mining, and network security.

Kim-Kwang Raymond Choo received his Ph.D. in Information Security in 2006 from the Queensland University of Technology, Australia. He currently holds the Cloud Technology Endowed Professorship at The University of Texas at San Antonio (UTSA). He is an IEEE Computer Society Distinguished Visitor from 2021 to 2023 (inclusive), and included in Web of Science's Highly Cited Researcher in the field of Cross-Field - 2020, and in 2015 he and his team won the Digital Forensics Research Challenge organized by Germany's University of Erlangen–Nuremberg. He is the recipient of the 2019 IEEE Technical Committee on Scalable Computing (TCSC) Award for Excellence in Scalable Computing (Middle Career Researcher), the 2018 UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty, the British Computer Society's 2019 Wilkes Award Runner-up, the 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, the Fulbright Scholarship in 2009, the 2008 Australia Day Achievement Medallion, and the British Computer Society's Wilkes Award in 2008. He has also received best paper awards from the IEEE Consumer Electronics Magazine in 2020, EURASIP JWCN in 2019, IEEE TrustCom 2018, and ESORICS 2015; the Korea Information Processing Society's JIPS Survey Paper Award (Gold) 2019; the IEEE Blockchain 2019 Outstanding Paper Award; and Best Student Paper Awards from Inscrypt 2019 and ACISP 2005.