Controllable and trustworthy blockchain-based cloud data management

Liehuang Zhu, Yulu Wu, Keke Gai, Kim-Kwang Raymond Choo

PII: S0167-739X(18)31199-3
DOI: https://doi.org/10.1016/j.future.2018.09.019
Reference: FUTURE 4454

To appear in: *Future Generation Computer Systems*

Received date : 15 May 2018
Revised date : 25 August 2018
Accepted date : 5 September 2018

Controllable and Trustworthy Blockchain-based Cloud Data Management

Liehuang Zhu[a], Yulu Wu[a], Keke Gai[a,*], Kim-Kwang Raymond Choo[b]

[a]*School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China, 100081.*
[b]*Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX 78249-0631, USA.*

Abstract

In recent years, there have been efforts to deploy blockchain in a broad range of applications and in different domains, such as the critical infrastructure sectors. Generally, blockchain can be leveraged to establish a fair and transparent data sharing environment where unauthorized modification to the data can be audited and traced. There are, however, known limitations of blockchain-based solutions. For example, a significant weakened networking control capability due to the distributed nature of blockchain-based solutions. In addition, decisions recorded on a blockchain cannot be changed and there is the risk of majority attack (also known as 51% attack). Seeking to mitigate these limitations, in this paper we propose a *controllable blockchain data management* (CBDM) model that can be deployed in a cloud environment. We then evaluate its security and performance, in order to demonstrate utility.

Keywords: Blockchain, data management, trustworthiness, cloud computing, privacy-preserving

1. Introduction

The recent trend in blockchain is probably due to the success and popularity of bitcoin. The interest in blockchain is also evidenced by the increasing number of blockchain-based solutions in a broad range of fields [1, 2, 3, 4, 5]. This is not surprising, for example due to its capability to provide a transparent data usage and sharing environment [6, 7]. Specifically, a blockchain system is

*Keke Gai (Corresponding Author): School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China, 100081. Email: gaikeke@bit.edu.cn.

Email addresses: `liehuangz@bit.edu.cn` (Liehuang Zhu), `2120171080@bit.edu.cn` (Yulu Wu), `gaikeke@bit.edu.cn` (Keke Gai), `raymond.choo@fulbrightmail.org`, `raymond.choo@utsa.edu` (Kim-Kwang Raymond Choo)

This work is supported by Beijing Institute of Technology Research Fund Program for Young Scholars (Dr. Keke Gai).

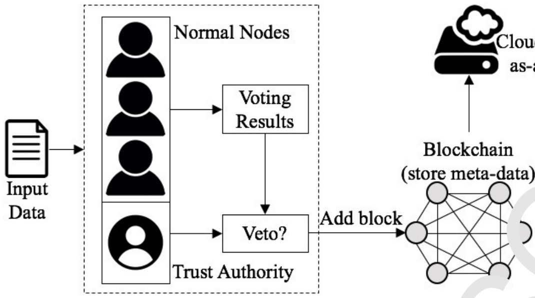

Figure 1: High level architecture of the proposed model.

widely considered a secure platform since all actions made by system participants are recorded on the chain and the continuing expanding chain makes it computationally challenging to change any block without detection (i.e. blocks are iteratively linked by cryptographic hashes).

Blockchain also permits flexible access, which may be used to achieve privacy-preserving feature, for example by permitting the use of alias accounts [8, 9, 10, 11]. For example, on many cryptocurrency systems, the owner of the digital currency is permitted to register an alias account, so that the identity of the owner is neither distributed nor published in the system [12]. The only authentication is the configuration of alias accounts rather than the owner's actual attributes that are associated with his/her personal information.

However, no technology is perfect. In the context of blockchain, for example, there is the risk of a majority (or 51%) attack.

In this paper, we propose a novel approach that seeks to mitigate the lack of control in blockchain. In our proposed *Controllable Blockchain Data Management* (CBDM) model (see also Fig. 1), we introduce a specific node in the network system. This particular node is also referred to as the *Trust Authority* (TA) node, which has a higher level of voting authorization in comparison to other participant nodes. The system configures a TA with a veto power, which is particularly designed for preventing malicious voting. Such an approach differs from a typical blockchain system, as it configures a semi-disruptive working mode with a partial centralized control capability. Moreover, our model leverages cloud storage to achieve storage efficiency (i.e. *Storage-as-a-Service* (StaaS)), and data in each block only store meta data in order to enhance block construction efficiency and minimize distributive storage waste.

We will next review the extant literature (see Section 2. Then, we present the system design of the proposed approach as well as the core definitions in Section 3. The key algorithms in our approach are described in Section 4. Section 5 presents the security analysis and partial results of the experiment evaluation. Finally, this study is concluded in Section 6.

2. Related Work

There is a growing body of work on blockchain in recent years, and in this section we will briefly review relevant literature.

One particular popular and successful application of blockchain is in smart contracts [13, 14, 15, 16, 17, 18]. For example, Kosba et al. [19] designed a decentralized smart contract system, *Hawk*. To ensure privacy, plaintext financial transactions are not stored on the blockchain and a cryptographic protocol can be constructed by the Hawk programmer. In another work, Heilman et al. [9] presented an approach to ensure anonymity for both blockchain-oriented and off-chain transactions. In this approach, there is an untrusted third party tasked to distribute anonymous vouchers that can be considered the monetary transaction. Generally, smart contract systems are associated with openness and traceability.

Blockchain can be used to facilitate secure data management or to enhance the security level in a particular application (e.g. Internet of Things (IoT) environment). For example, Aitzhan et al. [20] attempted to enhance the security of trading data in their decentralized smart grid energy trading approach. More specifically, their approach uses a series of blockchain-oriented techniques to achieve privacy protection in multi-signature, anonymous messaging encryption, anonymous negotiable energy trading, and other applications. Studies such as those of [21] focused on user-centric services, for example using blockchain to enable content delivery management. In such approaches, a content brokering contract is used to specify both contents and user preferences.

Generally, blockchain-based data management approaches leverage the decentralization characteristic of blockchain, without the capability to control its over-centralization. In this presented work, a trust authority is introduced to facilitate blockchain data management. This allows us to monitor data that needed to be recorded and the operations users made to minimize risk.

Blockchain and blockchain-based services can be deployed in environment such as the cloud [22, 23, 24]. This has also resulted in the coining of the term, *Blockchain-as-a-Service* (BaaS). Specifically, BaaSo utilizes blockchain's feature of tamper proof storage for approved / authorized actions. There have also been efforts to utilize blockchain techniques to enhance the performance of cloud systems. For instance, the authors in [25] utilized blockchain techniques to combine cloud computing, *Software-Defined Network* (SDN), and fog nodes, in order to accurately allocate computing resources in the distributed blockchain cloud architecture. In yet another work [26], the authors combined both cloud and blockchain techniques to facilitate secure medical data sharing between cloud service providers. However, the size of cloud data is generally significant and it is not realistic to store cloud data in the blockchain. In addition, as discussed earlier, data stored in the blockchain is open and transparent, and thus it is not advisable to store sensitive data, even when such data are encrypted, on the chain.

Blockchain can also be utilized to optimize key management. For example, Lin et al. [27] demonstrated how blockchain can be used to facilitate user authentication, where a linearly homomorphic signature on blockchain was designed to avoid the drawbacks of public-key certificates. Additionally, a dynamic key management approach was proposed in [28], in order to ensure security in *Vehicular Communication Systems* (VCSs). Blockchain can also be used to faciliate group message broadcast. For example, Guo et al. [29] explained how blockchain can be implemented in *Electronic Health Records* (EHRs) systems, by using an attribute-based signature scheme.

Another recent blockchain research trend is in the design of scalable blockchain. For example, Otte et al. [30] explained that it is possible to build a scalable blockchain for each agent for the purpose of achieving scalability. Such an approach follows the rule of immutable chain, but uses an order of temporal interactions for each agent. Thus, each agent could have a genesis block, and hence achieves a certain scalability. However, such approach does not address the "lack-of-control" limitation.

We can draw the following observations from existing literature. Firstly, the decentralized nature of smart contract can be utilized to achieve other functionality. Secondly, blockchain can be integrated with other consumer technologies, such as cloud systems. Thirdly, key management can potentially be optimized by using blockchain, and finally "lack-of-control" is an ongoing challenge in blockchain-based solutions.

In the next section, we present the system design of the proposed model.

3. System Design

3.1. Preliminaries

In this section, we outline the bilinear pairing technique, which will serve as the basic point of the proposed CP[illegible]DM scheme.

Bilinear Pairing: The bilinear pairing defines three multiplicative cyclic groups G_1, G_2, G_T of the same prime order p. Let $g_1 \in G_1$ and $g_2 \in G_2$ two generators for G_1 and G_2 respectively. A pairing relationship $\hat{e}$: $G_1 \times G_2 \rightarrow G_T$ is defined on the three groups G_1, G_2, G_T and it satisfies the following properties: (i) Bilinear: $\forall$ g $\in G_1$, h $\in G_2$, a, b$\in Z_p$, $e(g^a, h^b) = e(g,h)^{ab}$; (ii) Non-degenerated: $\exists\, g_1 \in G_1$, $g_2 \in G_2$, $e(g_1, g_2) \neq 1_{G_T}$, where 1_{G_T} is identity element in G_T; (iii) Computable: $\forall\, g \in G_1$, $h \in G_2, e(g,h)$ can be computed efficiently;

Definition 3.1. Bilinear Pairing Generator: *A bilinear pairing generator BPG is a probabilistic algorithm that takes a security parameter λ as its input, and outputs some parameters $\{G_1, G_2, \hat{e}, q, P, H\}$, where q is a λ-bit prime number, (G_1, G_2) are two cycle groups of the same order q, G_1 is multiplicative group, G_2 is additive group.*

3.2. Problem Definition

Modifying documents is very common in document management, but it is a serious problem to guarantee changed documents legality and security in trustworthy document management system. When users modify documents, the

DMS will encounter many problems of trustworthiness and reliability, such as invalidated documents changes and illegal operations. For example, modifying architectural blueprints is a common operation in the construction industry. Without validations, document changes are typically based on an assumption that all architects' inputs are validated. However, this assumption is inapplicable in reality due to many reasons, such as distinct capabilities and intended improper modifications for financial concerns.

To address the problem above, a trustworthy, transparent and traceable approach is needed to achieve validate document modifications. We introduce a blockchain-enabled system to record each user's request (documents changes) and their key behavior. We also introduce a TA to monitor users identities and behaviors. The probability of attacks will be reduced on a blockchain system, because attackers must possess more than 51 percent hashing power for launch a malicious activity. In most situations, the cost of the computational resource is greater than the gain of the attack.

Definition 3.2. Controllable Blockchain Data Management (CBDM) Problem: *Input: a set of encrypted messages of requesting to change documents of users,* $\{Enc(\mathcal{M}_i)\}$*, where* $i \in N$*. Output: the vote results set,* $\{\mathcal{R}_i\}$*, where* $i \in N$*.*

In Definition 3.2, a set of encrypted messages ($\{Enc(\mathcal{M}_i)\}$) refers to requests to change documents users sending to administrator in blockchain system, where N means the amount of users. The vote results set ($\{\mathcal{R}_i\}$) refers to the approval or the rejection to each user's request. Users send their encrypted requests of changing documents to administrator of blockchain system to achieve its agreement. Administartor of blockchain system verifies whether user's identity is valid or not. If user's identity is valid, the user's request will be voted by other users and trust authority. If user's identity is invalid, the user's request will be dropped. After the vote of users and trust authority, the final result $\{\mathcal{R}_i\}$ returns to the user to decide if the user has the right to modify documents.

3.3. Threat Model

Due to the *Honest-But-Curious* feature of cloud computing, it will mine the data information to analyze the contents of documents and the relationship between users. So the CBDM scheme can have two types of attackers: the external adversary and the internal adversary. Based on attack types, we define a main threat model that is *User Collusion Attack Model* (UCAM). In this model, we assume that users have the opportunity to collude with each other to modify documents to get invalid documents, then they conspire to malicious votes so that invalid documents are approved as pseudo-legitimate documents. Meanwhile, some users may have some bad behavior, which is also recorded on the blockchain and reviewed by the rest of the users causing bad consequences. Besides, attackers outside may forge some users' secret signatures to vote for illegal documents or achieve some valid users' identity.

3.4. Design Objectives

Considering the threat models mentioned above, our design goal is to propose a secure, reliable and privacy-preserving CBDM with traceability scheme. The following design objectives should be satisfied:

Controllability: We aim at designing a special node with a veto power which is TA (trust authority) to efficiently monitor voting actions $\mathcal{A}$ in the blockchain. Because of public keys U_{pk} and private keys U_{sk} distributed for users U by TA, TA can trace U true identities ids through signatures U_{sig} of U on votes and revoke the voting rights $\mathcal{V}$ of U. U who request to modify documents D will send TA modified documents nD in a secure channel, then TA will compare nD with those which are found on the cloud by hash values dH of changed documents to judge they are valid or not aiming at avoiding documents are tampered with. Besides, attackers can not tamper with voting information vR because they are incapable of decrypting encrypted vR under the situation of not being detected due to the secure encryption algorithm.

Privacy-preserving: When U join the system, TA will distribute U_{pk} and U_{sk} for each user. U will sign their votes by their U_{sk} during voting stage. No user knows others or others' vR. All mentioned for privacy-preserving guarantee users anonymity and privacy. Each user needs to obtain the permission $\mathcal{P}$ of the TA in the blockchain system, that is to say, U only can be nodes in the blockchain system through user identity certificates U_{cert} published by TA which guarantees that U joining the blockchain are valid. Besides, each user should give corresponding fees for voting to avoid malignant vote.

Openness and Transparency: Both modification records cR and voting records vR are published to the blockchain to give convenience to U view which satisfies openness and transparency. If an attacker intercepts a voting message and sends it to counting contracts CC, CC will find that the vote has existed and drop the vote. The attacker also cannot forge a vote because it can not forge the VC's signatures due to signature algorithm security. Any nodes are free and peer-to-peer, they can join or leave blockchain system at anytime, so that the system will not be affected the operation due to a node's crash.

3.5. Proposed Model

The system model is illustrated in Fig. 2, which consists of four parties: trust authority (TA), cloud servers (CS), blockchain system (BS) and a group of users $U = \{U_1, U_2, \cdots, U_N\}$. And in our scheme, we consider a scenario that some of users may request TA to change documents $D = \{D_1, D_2, \cdots, D_M\}$ and send other users their changed documents $nD = \{nD_1, nD_2, \cdots, nD_S\}$ after achieving confirmation for changing documents from TA, where each $nD_i \in nD$ indicates a new version document of documents $D = \{D_1, D_2, \cdots, D_M\}$. Then other users vote for the changed documents in trustworthy document management system. Detailed descriptions of four parties are given in the followings:

Trust Authority (TA): TA is a fully trustable party, which is a legitimate and authoritative institute authorized by government for blockchain system. The most important for TA is that it could examine the changed document to determine it invalid to reject it regardless of other users opinions.

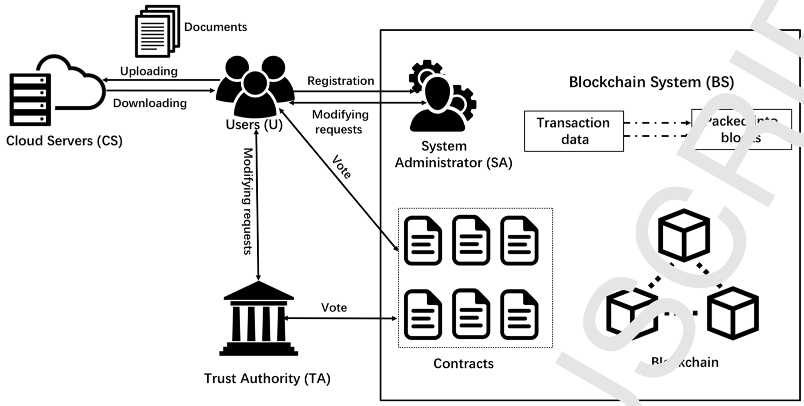

Figure 2: The system model of blockchain-based controllable trustworthy document management approach.

Cloud Servers (CS): Cloud servers only store the encrypted changed documents to save users storage space. It is convenient for users and TA to review the changed documents and avoid users making the same change on documents next time.

Blockchain System (BS): Blockchain system counts the votes for changed documents to decide whether users approve or reject the change. In blockchain system, there are three entities, which are an system administrator SA, a set of voting contracts and a set of counting contracts. SA takes charge of the registration of all users and users requesting to change documents aiming at achieving the confirmation. SA distributes public keys, private keys, address, which is generated by public keys to each user and public and private key pairs to a set of voting contracts $VC = \{VC_1, VC_2, \cdots, VC_R\}$ and counting contracts $CC = \{CC_1, CC_2, \cdots, CC_R\}$. SA also manages the time for registration, voting starting and stoping and system initialization.Voting contracts are used to verify users identities and give a platform for users to vote. Counting contracts are used to count the amount of votes and publish votes and voting results. Blockchain is used to store records of changing documents $cR = \{cR_1, cR_2, \cdots, cR_T\}$ and voting information $vR = \{vR_1, vR_2, \cdots, vR_Q\}$ and the hash value of changed documents $dH = \{dH_1, dH_2, \cdots, dH_T\}$. Blockchain packages the information mentioned above after counting contract publishing these records. Users can retrieve the content of blocks to find the hash value of previously changed documents and download them from cloud servers to check the past changes to documents consisting valid changes and invalid changes. Compared with cloud servers, blockchain is public, transparent,unforgeability and difficult to be attacked.

Users ($U = \{U_1, U_2, \cdots, U_N\}$): each user $U_i \in U$ is registered by SA. They can request SA and voting contracts to change documents, and each of them has the authority to determine whether to approve the proposal of modifing documents or not. Besides, they can trace changed documents from blockchain system and cloud servers.

Specifically speaking, there are three phases in our CBDM scheme: (1) system initialization, (2) document modification, and (3) document management.

System Initialization This phase essentially initializes the system by configuring a group of parameters and completing key generations for user registrations. Descriptions about phases are given in the followings.

(1) *SystemSetup:* The system sets a security parameter λ to create parameters $\{G_1, G_2, \hat{e}, q, P, H\}$. SA generates public key G_{pk} and private key G_{sk} for itself, who takes a responsibility to be a group administrator, and it generates public and private keys pairs to contracts. TA generates public and private key pairs for itself according to parameters $\{G_1, G_2, \hat{e}, q, P, H\}$. Besides, all addresses are created by corresponding public keys. SA also needs to set time for registration starting and stoping and time for voting starting and stoping.

(2) *Registration:* Each user who wants to join the blockchain system should register to SA, then SA gives public keys $U_{pk} = \{U_{pk1}, U_{pk2}, \cdots, U_{pkN}\}$ and private keys $U_{sk} = \{U_{sk1}, U_{sk2}, \cdots, U_{skN}\}$ to user which can make user be valid in blockchain system and generates secret signature keys $U_{sig} = \{U_{sig1}, U_{sig2}, \cdots, U_{sigN}\}$for users.

Document Modification This phase addresses major operations of the voting implementation in smart contract, including request, verification, vote, and count. It represents one of the core functions offered by TA, which is a veto power right. Explanations about these operations are provided below.

(1) *Request:* User $U_i \in U$, where $i \in N$ sends a request message $\mathcal{M}$ to change documents with his private keys encrypting his request $\mathcal{M}$ to SA. Besides, this user needs to send his secret signature key and sends the modified document encrypted by his private key to TA in a secure channel.

(2) *Verification:* SA checks user's request $\mathcal{M}$ decrypted by user's public key and verifies whether the secret signature key is valid or not. If the the secret signature key is valid, SA will agree with the user's request and make this request to be a vote option. Otherwise, the request will be rejected.

(3) *Vote:* Each user sends his hash value of vote option to TA. TA generates some signature parameters for users. Then user generates his own signature for vote option and sends the signature to voting contract. Voting contract checks whether the signature is valid or not. If the signature is valid, voting contract will sign it by its own private key and send it to user. User encrypts the signed signature, his signature and his vote option by counting contract's public key and send them to counting contract. TA also needs to vote for these modified documents and it has a **veto power right**.

(4) *Count:* If counting contract verifies the signatures of users are valid, it will count the vote. Counting contract is used to count the amount of votes and computes the voting results. The vote of TA should be considered separately. If TA approves the modified document, its vote is similar to normal users'. If TA disagrees with the modified document, counting contract will publish the voting result only according to TA's vote option.

Document Management This phase emphasizes a number of primary operations on blockchain, which include recording document changes, document uploads, and document downloads.

(1) *RecordStore:* Records of changing documents, users voting for changed documents and TA's determination of whether accepting changed documents or not will be packaged in blocks with hash values of changed documents every ten minutes. Both TA and users can verify the validity of changing documents and find the documents exactly by the hash values of documents which can make users to review the reasons for the rejection and adoption of the previous changed documents with the use of blockchain.

(2) *Upload:* Users who want to modify documents should upload changed documents to cloud servers in encrypted form and all group members have secret keys to decrypt encrypted changed documents.

(3) *Download:* TA and users can download changed documents to make a reference for their future revision. To TA, it can also compare the modified document downloaded from cloud with the modified document sent from user who wants to modify documents to check whether these two documents are same or not which is a great basis for rejecting the request.

The next section presents main proposed algorithms in our approach.

4. Algorithms

4.1. User Registeration Algorithm

The user registration algorithm is designed to generate corresponding parameters for users. We consider that if users register the blockchain system, they should achieve ids and secret signature keys to modify documents and vote modified documents. The input of user registration algorithm are the number of unregistered users Num_{unreg}, the number of modified documents Num_d and the hash value of modified documents. The output of the user registration algorithm are users' public keys, private keys, addresses, which are achieved by public keys and secret signature keys. Users' private keys are used to make users can do operations, such as voting.

Next, at SystemSetup phase, SA has distributed public and private keys pairs to contracts and addresses to TA, contracts and itself. According to 3.1, SA generates some parameters $\{G_1, G_2, \hat{e}, q, P, H\}$ according to 3.1 and expose them, where $H : \{0,1\}^* \rightarrow G_1$. SA sets $x \in Z_q{}^*$ as SA's private key G_{sk} and

Algorithm 1 **User Registration Algorithm**

Require:
The number of unregistered users Num_{unreg};
The number of modified documents Num_d;
The hash value set of modified documents dH;
Ensure:
Users' public keys set U_{pk};
Users' private keys set U_{sk};
Users' addresses set U_{Addr};
Users' secret signature keys set U_{sig};

for $i = 0; i < Num_{unreg}; i++$ **do**
  SA generates public key U_{pki} and private key U_{ski} for user $U_i \in \mathbf{U}$
  U_{pki} generates address U_{Addri} for U_i
  SA generates a random number $x_i{}^u \in Z_q{}^*$ for user U_i
  SA computes $x_i{}^s = (x - x_i{}^u) \bmod q$ and sends $(x_i{}^s, U_i)$ to TA
  TA generates a random number $y_i{}^u \in Z_q{}^*$ for user U_i, computes $y_i{}^s = (y - y_i{}^u) \bmod q$ and stores $y_i{}^s$
  Each user U_i has their secret signature key U_{sigi} $(x_i{}^u, y_i{}^u)$
end for
return U_{pk}, U_{sk}, U_{Addr} and U_{sig}

computes $X = xP$ as its public key G_{pk}. TA set $y \in Z_q{}^*$ as its private key $T_{sk}$and computes $Y = yP$ as itself's public key T_{pk}.
Algorithm 1 represents the pseudo codes of the user registration algorithm.

The main phases of this algorithm include:

- The administrator of blockchain system SA generates public keys set U_{pk} and private keys set U_{sk} for users $\mathbf{U}$ registering for blockchain system.
- Blockchain system generates users' addresses set U_{Addr} by users' public keys set U_{pk} to let users join the blockchain system.
- SA and TA generate random number $(x_i{}^u, y_i{}^u)$ to achieve secret signature keys set U_{sig} for each user.
- SA computes $x_i{}^s = (x - x_i{}^u) \bmod q$ and sends $(x_i{}^s, U_i)$ to TA. TA computes $y_i{}^s = (y - y_i{}^u) \bmod q$ and stores $y_i{}^s$. When users vote for the request, TA can check whether their identy is valid or not by computing $x_i{}^s$, $x_i{}^u$, $y_i{}^s$ and $y_i{}^u$.

4.2. Voting And Counting Algorithm

The voting and counting algorithm of CBDM scheme is designed to let users vote for modified documents and make contracts count votes to publish the vote results. The input of the voting and counting algorithm are the number of voters Num_v and hash values set of each voter's vote option $H(V)$. The output of the voting and counting algorithm is the vote result. Algorithm represents the pseudo codes of the voting and counting algorithm.

The main phases of this voting and counting algorithm include:

1. For each voter, he sends his vote option $H(V_i)$ and his secret signature key U_{sig_i} to TA, then TA checks whether this voter is valid. We set some requirements for voters to check whether their secret signature keys are valid or not. The detailed requirements are:
   - ${x_i}^u \neq {x_j}^u$ and ${y_i}^u \neq {y_j}^u$, where $i \neq j$. This requirement means that each user's signature should be different from others'.
   - ${x_{i_1}}^u + {x_{i_2}}^u + \cdots + {x_{i_j}}^u \neq x \bmod q$ and ${y_{i_1}}^u + {y_{i_2}}^u + \cdots + {y_{i_j}}^u \neq y \bmod q$ where $j \in \mathbb{Z}$. This requirement indicates that any random numbers of users can't add up to be equal to x and y to avoid x and y being guessed. Once the above formulas can be equalized, users can collude to generate their own legitimate signatures to illegally vote.
   - ${x_{i_1}}^u + {x_{i_2}}^u + \cdots + {x_{i_j}}^u \neq {x_j}^u \bmod q$ and ${y_{i_1}}^u + {y_{i_2}}^u + \cdots + {y_{i_j}}^u \neq {y_j}^u \bmod q$ where $j \in \mathbb{Z}$. This requirement presents that any random numbers of users can't add up to be equal to ${x_j}^u$ and ${y_j}^u$ to avoid ${x_j}^u$ and ${x_j}^u$ being guessed. Once ${x_j}^u$ and ${x_j}^u$ are guessed, x and y will be guessed finally. Then users can collude with each other to illegally vote.
2. When TA has checked the validity of voters' signature keys, they computes a part of signatures of voters' vote option, which are $({v_i}^s, {\sigma_i}^s)$. And ${v_i}^s = {y_i}^s * H(V_i)$, ${\sigma_i}^s = {x_i}^s * H(V_i)$. Besides, TA stores ($U_{Addr}$ and $H(V)$) to open the signature by querying ($U_{Addr} and H(V)$) aiming at check the user identity of the signature of vote option. Then voters compute their full signatures by the random number ${x_i}^u$ generated by SA.
3. Each voter sends his signature σ_i to voting contract, where $\sigma_i = {v_i}^s + {v_i}^u + {\sigma_i}^s + {\sigma_i}^u$. Then, voting contract checks the validity of voters' signatures in Eq. (1). And ${v_i}^u = {y_i}^u * H(V_i)$ and ${\sigma_i}^u = {x_i}^u * H(V_i)$.

Because

$$
\begin{aligned}
\sigma_i &= {v_i}^s + {v_i}^u + {\sigma_i}^s + {\sigma_i}^u \\
&= {y_i}^s * H(V_i) + {y_i}^u * H(V_i) + {x_i}^s * H(V_i) + {x_i}^u * H(V_i) \\
&= {y_i}^s + {y_i}^u * H(V_i) + {x_i}^s + {x_i}^u * H(V_i) \\
&= (x + y) * H(V_i)
\end{aligned}
\tag{1}
$$

Therefore,

$$\hat{e}(P, \sigma) = \hat{e}(X + Y, H(V_i))$$

When voters' signatures are checked and they are valid voters, voting contract will sign voters signtures by voting contract's private key Y_A.

Algorithm 2 **Voting And Counting Algorithm**

Require:
The number of voters Num_v;
The hash values set of each voter's vote option $H(V)$;
Ensure:
The vote result $\mathcal{R}$;

for $i = 0; i < Num_v; i++$ **do**
  Voter v_i sends his vote option $H(V_i)$ and his secret signature key U_{sig_i} to TA
  TA checks the validity of voter's signature.
  if ${x_i}^u \neq {x_j}^u$ and ${y_i}^u \neq {y_j}^u$, where $i \neq j$ **then**
    if ${x_{i_1}}^u + {x_{i_2}}^u + \cdots + {x_{i_j}}^u \neq x \bmod q$ and ${y_{i_1}}^u + {y_{i_2}}^u + \cdots + {y_{i_j}}^u \neq y \bmod q$ where $j \in \mathbb{Z}$ **then**
      if ${x_{i_1}}^u + {x_{i_2}}^u + \cdots + {x_{i_j}}^u \neq {x_j}^u \bmod q$ and ${y_{i_1}}^u + {y_{i_2}}^u + \cdots + {y_{i_j}}^u \neq {y_j}^u \bmod q$ where $j \in \mathbb{Z}$ **then**
        U_{sig_i} is valid.
      end if
    end if
  end if
  while U_{sig_i} is valid **do**
    TA computes ${v_i}^s = {y_i}^s * H(V_i)$ and ${\sigma_i}^s = {x_i}^s * H(V_i)$
    TA stores $(U_{Addri}, H(V_i))$ and sends voter v_i $({v_i}^s, {\sigma_i}^s)$
    Voter v_i computes ${\sigma_i}^u = {x_i}^u * H(V_i)$ and ${v_i}^u = {y_i}^u * H(V_i)$ and sets his signature on vote option is σ_i, where $\sigma_i = {v_i}^s + {v_i}^u + {\sigma_i}^s + {\sigma_i}^u$
    Voter sends voting contract his signature σ_i
    Voting contract checks the signature's validity
    if $\hat{e}(P, \sigma) = \hat{e}(X + Y, H(V_i))$ **then**
      σ_i is valid
    end if
    while σ_i is valid **do**
      Voting contract uses its private key Y_A to sign σ_i
      Voter uses public key X_B of counting contract to encrypt $\{V_i || Y_A(\sigma_i) || \sigma_i\}$ to get $m_i = Enc\{V_i || Y_A(\sigma_i) || \sigma_i\}$ and sends m_i to counting contract
      Counting contract verifies the validity of voter's signature and counts the corresponding vote
    end while
  end while
end for
if TA Votes **then**
  Counting contract sets the vote result $\mathcal{R}$ to be the vote option of TA
  if TA approves **then**
    Pack up the hash value into the block
  else if **then**
    Mark the document as "denied"
  end if
else if **then**
  Pack up the hash value into the block
end if
return $\mathcal{R}$

Table 1: Partial Experiment Settings

Setting	Document Size	User Amount
Setting 1-1	10	[0, 50]
Setting 1-2	20	[0, 50]
Setting 1-3	50	[0, 50]
Setting 1-4	100	[0, 50]
Setting 2-1	10	[0,100]
Setting 2-2	20	[0,100]
Setting 2-3	50	[0,100]
Setting 2-4	100	[0,100]

4. Voters encrypt $\{V_i||Y_A(\sigma_i)||\sigma_i\}$ by counting contract public key X_B and send $Enc\{V_i||Y_A(\sigma_i)||\sigma_i\}$ to counting contract. Counting contract decrypts $Y_A(\sigma_i)$ by voting contract's public key X_A to check whether the signature is valid. Besides, counting contract computes the hash value of vote option V_i to justify whether the vote option is true. After verifying the validity of voters, counting contract counts the vote information according to $H(V_i)$.

5. TA has a **veto power right**. Once TA votes for one vote option, counting contract will set the vote result $\mathcal{R}$ to be the vote option of TA. If TA approves of the users' decision, blockchain system will pack up the final document hash value. If TA denies users vote option, the document hash value will be marked "denied".

5. Experiment Evaluation and Security Analysis

5.1. Experiment Evaluation

This section presented the experiment configuration and partial results to exhibit main findings deriving from our evaluations. We mainly focused on assessing two aspects in our experiments, which were cost and efficiency. The meaning of the cost in our model referred to the financial cost produced during the mining process. An assessment on efficiency meant the time length of constructing blocks. In order to achieve the evaluation objective above, we configured two variable parameters, which were user amount and document size. A variety of application scenarios were evaluated in our experiment. Due to the page length limit, we only presented partial experiment results collected from two types of scenarios. Table 1 showed partial experiment settings that were aligned with the experiment results provided in this section.

Moreover, considering the environment configuration, we used an Ethereum client Geth (1.8.3-stable) and an Ethereum Wallet 0.10.0 running on a desktop computer MacBook Pro 2017 (OS: macOS 10.13.4, CPU: 2.3GHz Intel Core

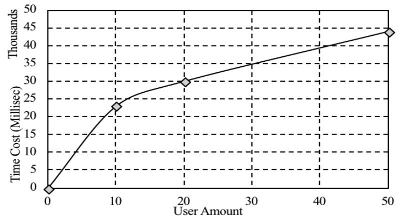

Figure 3: Results under Setting 1-1.

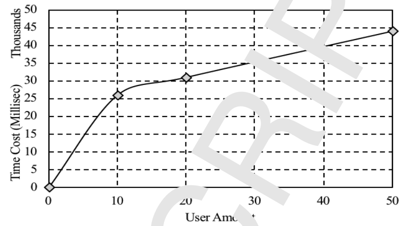

Figure 4: Results under Setting 1-2.

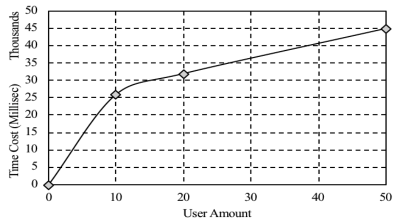

Figure 5: Results under Setting 1-3.

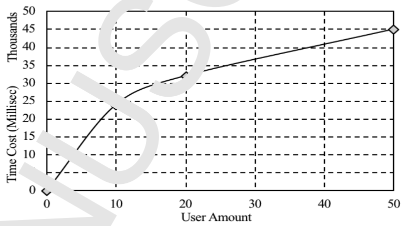

Figure 6: Results under Setting 1-4.

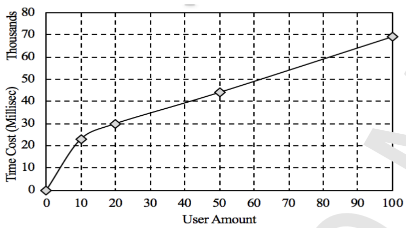

Figure 7: Results under Setting 2-1.

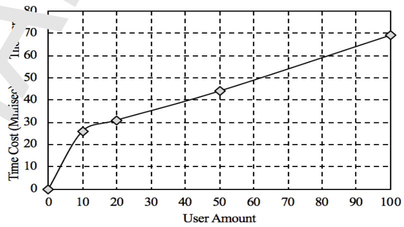

Figure 8: Results under Setting 2-2.

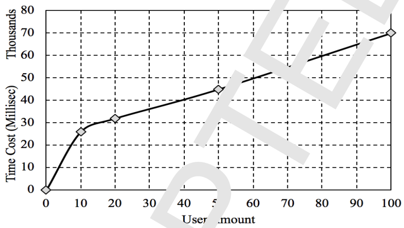

Figure 9: Results under Setting 2-3.

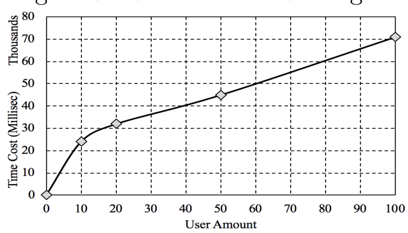

Figure 10: Results under Setting 2-4.

Comparisons of time consumptions for users in different documents' sizes.

i5, RAM: 8 GB of 2133 MHz LPDDR3). Under this environment, the miners' earned fee was equal to the value of multiply of the used gas and the gas unit price. Current price configuration was 0.018 Ether per million gas. The reason for selecting Ethereum as the experiment platform was that it could successfully simulated the application scenarios by constructing a private blockchain system.

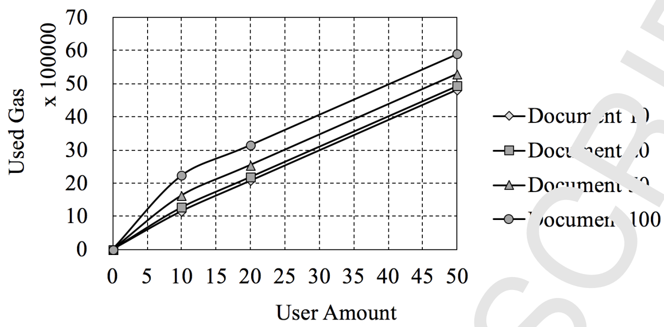

Figure 11: Gas costs of votes in different user amounts with different document sizes.

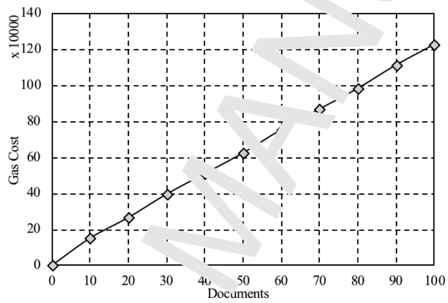

Figure 12: Gas used for packing up each document's hash value into a block.

Implementing experiments on Ethereum could examine a variety of aspects, including correctness, efficiency, and cost.

Moreover, we tested time cost of all votes and publishing results phases. The time of all votes and publishing results phases included time $t_{register}$ for user registration and time t_{vote} for user votes. Besides, the time also included publishing transactions and hash values of modified documents. Fig. 3 - Fig. 8 depicted various time costs for users in different documents. On the whole, we found that the cost had a positive relationship with both user amount and document size in all evaluated scenarios. First, we observed that the time cost tendency grew smoothly along with the increase of users sizes, when documents sizes were set to 10, 20, 50 and 100, respectively. Meanwhile, the incremental trend became lighter when the number of users was increased. According to our model design, the hash value of the document could be selected by users via smart contract rather than directly selecting documents. The advantage of this design was that the size of the document had limited impacts on the efficiency of the document selections.

According to Fig. 11, we found that the gas cost had a positive relationship with the user size. This phenomenon was reasonable due to the increase of the computation workload. At the vote and publishing results phase, users consumed the gas when registering to the blockchain system. Besides, users vote for modified documents and blockchain system published the result as well as the consumed gas.

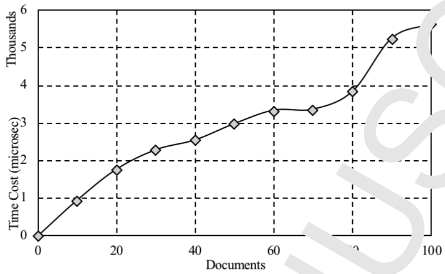

Figure 13: Time cost for packing up each document's hash value into a block.

In Fig. 12, it illustrated that the number of the modified document was linear to the gas cost, which included the committed transactions and the transaction confirmed by miners. Our collected data depicted that the number of documents was associated with the time cost. Meanwhile, the size of the document had a strong impact on the time cost. In Fig. 13, it showed that the time of packing up documents by using hash values also was linear to the number of modified documents. When number of documents turned into 80, the speed of the cost growth tended to be slower. We analyzed the results and found that this phenomenon was caused by the performance fluctuation of the computing resource. The trend still was likely a linear growth.

The proposed CBDM model proved that it is effective and correct according to the use of blockchain. We set a series of changing parameters to verify the CBDM model's feasibility and controllability and verified that the proposed model could offer a controllable document management system while considering trustworthiness and efficiency.

5.2. Security Analysis

In this section, we would prove the security of our proposed model in terms of three aspects of controllability, privacy-preserving and unforgeability.

- Because we introduced TA, our entire system is under the supervision of TA. When users register in the blockchain system (BS), SA distributes the public keys U_{pk} and private keys U_{sk} to users U registered in the BS and generates addresses for U. SA and TA generate signatures U_{sig} for registered users. In addition, TA needs to verify that whether they have voting rights $\mathcal{V}$ by checking U_{sig} when U vote for requests. Besides, it is also necessary to verify that if the users' votes are legal when counting votes. The entire process of requesting for modification of documents is monitored by TA. And any illegal actions users want to make and any illegal requests illegal user want to make cannot be authenticated by TA. Meanwhile, TA also has a veto power to veto illegal documents. Therefore, introducing TA will prevent any illegal behavior caused by excessive decentralization characteristic of blockchain.

- The encrypted requests for modifying documents D made by U on the BS are transmitted to SA, and the permission $\mathcal{P}$ only can be obtained after being verified by SA. Then, users' voting rights $\mathcal{V}$ and the votes are needed to be validated by TA to achieve real $\mathcal{V}$. Recorded on the block are the request data and users' voting data as well as the encrypted document data. If the document is too large, it will be encrypted and stored in the cloud. Of course, the cloud has all encrypted documents. These documents are protected by encryption algorithms to ensure a certain level of privacy. The rest data recorded on the block is the data that can be publicly viewed by registered users.

- When an attacker outside wants to forge a signature to vote, he must satisfy the corresponding signature requirements in order to falsify the real signature to obtain $\mathcal{V}$. Then, the algorithm part describes that there are many requirements needed to be met to obtain a legal signature. In this case, the security of the signature algorithm makes that an illegal user cannot obtain a real signature to obtain $\mathcal{V}$. When some of U want to collude to forge votes to get a fake legal document, TA will veto illegal documents in advance making users' collusion impossible due to TA has a veto.

6. Conclusions

In this paper, we presented a novel approach for achieving a controllable blockchain, CBDM. This allows us to address the concern relating to the lack of control on the posted ledgers. The introduction of a TA node in our approach allows one to terminate any potentially malicious actions even in a majority attack. Although we evluated the security and performance of the proposed approach in this paper, a future extension is to implement a prototype of this approach in a real-world environment. This will allow us to better evaluate the proposed approach and identify and address any shortcomings.

References

[1] G. Liang, S. Weller, F. Luo, J. Zhao, and Z. Dong. Distributed blockchain-based data protection framework for modern power systems against cyber attacks. *IEEE Transactions on Smart Grid*, PP(99):1, 2018.

[2] X. Liu, W. Wang, D. Niyato, N. Zhao, and P. Wang. Evolutionary game for mining pool selection in blockchain networks. *IEEE Wireless Communications Letters*, PP(99):1, 2018.

[3] R. Hen. A traceability chain algorithm for artificial neural networks using T-S fuzzy cognitive maps in blockchain. *Future Generation Computer Systems*, 80:198–210, 2018.

[4] X. Yue, H. Wang, D. Jin, M. Li, and W. Jiang. Healthcare data gateways: found healthcare intelligence on blockchain with novel privacy risk control. *Journal of medical systems*, 40(10):218, 2016.

[5] C. Esposito, A. De Santis, G. Tortora, H. Chang, and K.K.R. Choo. Blockchain: A panacea for healthcare cloud-based data security and privacy? *IEEE Cloud Computing*, 5(1):31–37, 2018.

[6] S. Cha, J. Chen, C. Su, and K. Yeh. A blockchain connected gateway for BLE-based devices in the internet of things. *IEEE Access*, PP(99):1, 2018.

[7] Z. Zhang, W. Cao, Z. Qin, L. Zhu, Z. Yu, and K. Ren. When privacy meets economics: Enabling differentially-private battery-supported meter reporting in smart grid. In *IEEE/ACM 25th International Symposium on Quality of Service*, pages 1–9, Vilanova i la Geltru, Spain, 2017. IEEE.

[8] G. Zyskind and O. Nathan. Decentralizing privacy: Using blockchain to protect personal data. In *IEEE Security and Privacy Workshops*, pages 180–184, San Jose, CA, USA, 2015. IEEE.

[9] E. Heilman, F. Baldimtsi, and S. Goldberg. Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions. In *International Conference on Financial Cryptography and Data Security*, pages 43–60. Springer, 2016.

[10] Z. Zhang, Z. Qin, L. Zhu, J. Weng, and K. Ren. Cost-friendly differential privacy for smart meters: Exploiting the dual roles of the noise. *IEEE Transactions on Smart Grid*, 8(2):619–626, 2017.

[11] K. Gai, K.K.R. Choo, M. Qiu, and L. Zhu. Privacy-preserving content-oriented wireless communication in internet-of-things. *IEEE Internet of Things Journal*, 5(4):3059 – 3067, 2018.

[12] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen. A survey on the security of blockchain systems. *Future Generation Computer Systems*, PP:1–1, 2017.

[13] M. Khan and K. Salah. IoT security: Review, blockchain solutions, and open challenges. *Future Generation Computer Systems*, PP:1–1, 2017.

[14] M. Shen, B. Ma, L. Zhu, R. Mijumbi, X. Du, and J. Hu. Cloud-based approximate constrained shortest distance queries over encrypted graphs with privacy protection. *IEEE Transactions on Information Forensics and Security*, 13(4):940–953, 2018.

[15] L. Zhu, X. Tang, M. Shen, X. Du, and M. Guizani. Privacy-preserving DDoS attack detection using cross-domain traffic in software defined networks. *IEEE Journal on Selected Areas in Communications*, 36(3):628–643, 2018.

[16] K. Gai and M. Qiu. Blend arithmetic operations on tensor-based fully homomorphic encryption over real numbers. *IEEE Transactions on Industrial Informatics*, 14(8):3590 – 3598, 2017.

[17] J. Ziegeldorf, R. Matzutt, M. Henze, F. Grossmann, and K. Wehrle. Secure and anonymous decentralized Bitcoin mixing. *Future Generation Computer Systems*, 80:448–466, 2018.

[18] F. Gao, L. Zhu, M. Shen, K. Sharif, Z. Wan, and K. Ren. A blockchain-based privacy-preserving payment mechanism for vehicle-to-grid networks. *IEEE Network*, PP(99):1–9, 2018.

[19] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *IEEE Symposium on Security and Privacy*, pages 839–858, San Jose, CA, USA, 2016. IEEE.

[20] N. Aitzhan and D. Svetinovic. Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams. *IEEE Transactions on Dependable and Secure Computing*, PP(99):1, 2016.

[21] N. Herbaut and N. Negru. A model for collaborative blockchain-based video delivery relying on advanced network services chains. *IEEE Communications Magazine*, 55(9):70–76, 2017.

[22] K. Gai, M. Qiu, and H. Zhao. Energy-aware task assignment for mobile cyber-enabled applications in heterogeneous cloud computing. *Journal of Parallel and Distributed Computing*, 111:126–135, 2018.

[23] K. Gai, M. Qiu, Z. Ming, H. Zhao, and L. Qiu. Spoofing-jamming attack strategy using optimal power distributions in wireless smart grid networks. *IEEE Transactions on Smart Grid*, 8(5):2431–2439, 2017.

[24] M. Samaniego and R. Deters. Blockchain as a service for IoT. In *IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 433–436, Chengdu, Sichuan, China, 2016. IEEE.

[25] P. Sharma, M. Chen, and J. Park. A software defined fog node based distributed blockchain cloud architecture for IoT. *IEEE Access*, 6:115–124, 2018.

[26] Q. Xia, E. Sifah, K. Asamoah, J. Gao, X. Du, and M. Guizani. MeD-Share: Trust-less medical data sharing among cloud service providers via blockchain. *IEEE Access*, 5:14757–14767, 2017.

[27] Q. Lin, H. Yan, Z. Huang, W. Chen, J. Shen, and Y. Tang. An id-based linearly homomorphic signature scheme and its application in blockchain. *IEEE Access*, 6:20632–20640, 2018.

[28] A. Lei, H. Cruickshank, Y. Cao, P. Asuquo, C. Ogah, and Z. Sun. Blockchain-based dynamic key management for heterogeneous intelligent transportation systems. *IEEE Internet of Things Journal*, 4(6):1832–1843, 2017.

[29] R. Guo, H. Shi, Q. Zhao, and D. Zheng. Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems. *IEEE Access*, 776(99):1–12, 2018.

[30] P. Otte, M. de Vos, and J. Pouwelse. TrustChain: A sybil-resistant scalable blockchain. *Future Generation Computer Systems*, PP:1–1, 2017.

Liehuang Zhu received his Ph.D. degree in computer science from Beijing Institute of Technology, Beijing, China, in 2004. he is currently a professor at School of Computer Science & Technology, Beijing Institute of Technology, Beijing, China. He has published more than 100 peer-reviewed journal or conference papers, including 10+ IEEE/ACM Transactions papers (IEEE TIFS, IEEE TII, IEEE TVT, IEEE TSG, Information Sciences, IEEE Network, Computer & Security, etc.). He has been granted a number of IEEE Best Paper Awards, including IWQoS '17, TrustCom '18. His research interests include security protocol analysis and design, wireless sensor networks, and cloud computing.

Yulu Wu is currently a Master student majored in Computer Science at the at School of Computer Science & Technology, Beijing Institute of Technology. Her research interests include cybersecurity, blockchain, and cloud computing.

Keke Gai received a Ph.D. degree in Computer Science from the Department of Computer Science at Pace University, New York, USA. He also holds degrees from Nanjing University of Science and Technology (BEng), The University of British Columbia (MET) and Lawrence Technological University (MBA and MS). He is currently an Associate Professor in the School of Computer Science and Technology at Beijing Institute of Technology, Beijing, China. Keke Gai has published more than 90 peer-reviewed journals or conference papers, 30+ journal papers (including ACM/IEEE Transactions), and 50+ conference papers. He has been granted five IEEE Best Paper Awards (IEEE TrustCom '18, IEEE HPCC '18, IEEE SSC '16, IEEE CSCloud'15, IEEE BigDataSecurity'15) and two IEEE Best Student Paper Awards (SmartCloud '16, HPCC '16) by IEEE conferences in recent years. His paper about cloud computing has been ranked as the "Most Downloaded Articles" of Journal of Network and Computer Applications (JNCA). He is involved in a number of professional/academic associations, including ACM and IEEE. His research interests include cloud computing, cyber security, combinatorial optimization, edge computing, and blockchain.

Kim-Kwang Raymond Choo received the Ph.D. in Information Security in 2006 from Queensland University of Technology, Australia. He currently holds the Cloud Technology Endowed Professorship at The University of Texas at San Antonio (UTSA). In 2016, he was named the Cybersecurity Educator of the Year - APAC (Cybersecurity Excellence Awards are produced in cooperation with the Information Security Community on LinkedIn), 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 2018 UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty, ESORICS 2015 Best Paper Award, 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, Fulbright Scholarship in 2009, 2008 Australia Day Achievement Medallion, and British Computer Society's Wilkes Award in 2008. He is also a Fellow of the Australian Computer Society, and an IEEE Senior Member.

Highlights

1. In this work, we have improved the design of the blockchain system by introducing a special node that is permitted to have veto power. The proposed blockchain system is different from a typical blockchain system, as it configures a semi-disruptive working mode with a partial centralized control capability.
2. Our approach has a high level adoptability for data storage due to the implementation of StaaS. Its application is suitable for a wide scope of scenarios as the storage restriction on the chain can be avoided.
3. The proposed work is an effective complementarity for current blockchain-based solutions. It solves the problem that illegal actions taken place while the block is created. Following legal monitors, a continuous growing blockchain can be enabled without the concern of malicious votes.