J Med Syst (2018) 42: 141
https://doi.org/10.1007/s10916-018-0997-3

SYSTEMS-LEVEL QUALITY IMPROVEMENT

Blockchain-Based Data Preservation System for Medical Data

Hongyu Li[1] · Liehuang Zhu[1] · Meng Shen[1] · Feng Gao[1] · Xiaoling Tao[2] · Sheng Liu[3]

Received: 2 March 2018 / Accepted: 12 June 2018 / Published online: 28 June 2018

Abstract

Medical care has become an indispensable part of people's lives, with a dramatic increase in the volume of medical data (e.g., diagnosis certificates and medical records). Medical data, however, is easily stolen, tampered with, or even completely deleted. If the above occurs, medical data cannot be recorded or retrieved in a reliable manner, resulting in delay treatment progress, even endanger the patient's life. In this paper, we propose a novel blockchain-based data preservation system (DPS) for medical data. To provide a reliable storage solution to ensure the primitiveness and verifiability of stored data while preserving privacy for users, we leverage the blockchain framework. With the proposed DPS, users can preserve important data in perpetuity, and the originality of the data can be verified if tampering is suspected. In addition, we use prudent data storage strategies and a variety of cryptographic algorithms to guarantee user privacy; e.g., an adversary is unable to read the plain text even if the data are stolen. We implement a prototype of the DPS based on the real world blockchain-based platform Ethereum. Performance evaluation results demonstrate the effectiveness and efficiency of the proposed system.

Keywords Data preservation · Blockchain · Medical data · Sensitive data · Ethereum

Introduction

Medical care has become an indispensable part of people's lives. Medical data is essential for patient diagnosis and follow-up. Since traditional paper medical record have certain disadvantages [1], such as being easily lost or damaged, there is an urgent need for rapid conversion to paperless and electronic data. For instance, hospitals are encouraged to use electronic record instead of paper record [2]. However, medical data, e.g., prescription records and medical history, exist in the form of electronic data, which are prone to falsification[3]. Therefore, it is necessary to use electronic data preservation technology, which notarizes data to provide legal evidence for medical dispute and medical negligence.

Many hospitals use electronic data to record the whole process of treating. These hospitals store patient's medical records in the database. In this way, when a doctor needs a prescription record, the service system extracts the required data from the database for the doctor [4]. However, as data stored in the database can be stolen or tampered or even deleted completely, patient's actual condition cannot be provided. If the data are submitted to a third-party notarized company for preservation, personal information may be leaked, and it is difficult to guarantee the reliability and availability of preservation because the third party's credibility cannot be confirmed.

Blockchain is an open distributed ledger based on peer-to-peer networks and consensus algorithms [5]. A

This article is part of the Topical Collection on *Blockchain-based Medical Data Management System: Security and Privacy Challenges and Opportunities*

✉ Meng Shen
shenmeng@bit.edu.cn

Hongyu Li
lihongyu@bit.edu.cn

Liehuang Zhu
liehuangz@bit.edu.cn

Feng Gao
gaofengbit@foxmail.com

Xiaoling Tao
txl@guet.edu.cn

Sheng Liu
liusheng@umfintech.com

[1] Beijing Institute of Technology, Beijing, China

[2] Guilin University of Electronic Technology, Guilin, China

[3] Union Mobile Finacial Technology Co., Ltd., Beijing, China

blockchain contains a continuously growing linked list of records, which ensures that a piece of data recorded in a block cannot be tampered with. Its distributed storage mechanism also guarantees that the data will not be lost. Although blockchain is suitable for solving the above-mentioned problems, challenges are the storage of large amounts of data in the limited space of blockchain transactions [6] and proof that the preserved data have not been tampered with.

In this paper, we propose a novel blockchain-based data preservation system (DPS) for medical data that provides a reliable data storage solution that guarantees the stored data will not be tampered with, verifies the validity of data conveniently, and ensures privacy for users. In particular, inspired by the concept of *proof of possession of balance* [7], we present the concept of *proof of primitiveness of data* here. That means the characteristics of data will be preserved forever in blockchain, we don't have to worry about data been tampered with, and the system can validate whether current data is identical to the original data or not. First, the actual overall system requirements are determined. Then, we introduce a full-fledged protocol that allows users to store the preservation in the blockchain and prove the primitiveness of the data. Compared to general electronic data preservation systems [8], an important advantage of the proposed DPS is that it addresses the issues of easy loss or tampering of data or farfetched data. Even if the data in the database have been tampered with or damaged, the data can be retrieved and verified through the blockchain. Based on the proposed protocol, the blockchain-based DPS is implemented on the *Ethereum* platform. The performance evaluation results show that the cost of preservation with the proposed DPS is less than US$2, even if the size of the preserved files is 50 MB.

The rest of this paper is organized as follows. "Preliminaries" reviews previously published papers and existing blockchain application systems, including the *Ethereum* platform. The actual operation of the proposed DPS and contracts are described in detail in "The proposed DPS" and "Algorithm design", respectively. The detailed algorithm processes are illustrated in "System process". Security analysis and performance evaluation are illustrated in "Security analysis" and "Performance evaluation", respectively. "Discussion" discusses the limitations of the DPS. Finally, we conclude the paper in "Conclusion".

Preliminaries

In this section, we first explain the fundamentals of blockchain and *Bitcoin*, as well as the reasons for choosing *Ethereum* as the underlying platform, and then present an overview of existing studies.

Bitcoin and blockchain

To realize a data preservation system, we resort to blockchain, which is an open and distributed ledger based on peer-to-peer networks and consensus algorithms. In particular, we replace the concept *proof of possession of balance* first proposed in *Bitcoin* with an equivalent concept, which is referred to as *proof of primitiveness of data*.

To implement the DPS effectively, we must ensure that preserved data cannot be tampered with, damaged, or lost. Blockchain has the characteristics of decentralization. For an attack to be effective, a hacker must attack all copies in all nodes simultaneously. Blockchain can also prevent malicious tampering with data, as data changes can be found immediately by synchronization nodes in the blockchain network [9]. In addition, the blockchain's synchronization method and consensus algorithm ensure that every node in the blockchain network can share data and that all copies in synchronous conditions are exactly the same as those of other nodes.

Bitcoin can be viewed as a distributed database consisting of many nodes in a P2P network. Figure 1 illustrates an example of a *Bitcoin* transaction. If A wants to pay two *Bitcoins* to B, then B needs to specify not only the amount on the transaction sheet but also the source of the two *Bitcoins*. As each transaction records the previous owner, current owner and the next owner of the funds, the transaction can be traced to the whole process.

Because each transaction is relatively fragmented, the *Bitcoin* system created the concept of the *block* to better account for transactions. The previous block ID is contained in each block to enable identification of its previous node. To avoid false or repeated transactions, it is necessary to construct a *Proof of Work* (PoW) [10, 11] system that will enable the new block to be trusted. To modify the transaction information within a block, a user must complete all the work of the block and its subsequent connected blocks, which greatly increases the difficulty of tampering with the data.

In the real world, every non-cash transaction is recorded by the banking system, and if the banking system collapses, all data are likely to be lost [12]. In the world of *Bitcoin*, all transaction records are kept in countless computers around the world. The main blockchain can be fully read as long as there is one computer with a *Bitcoin* program that works. Such a highly redundant storage architecture makes it very unlikely that the *Bitcoin* blockchain will be lost completely [5, 13, 14].

Ethereum

Ethereum's core idea is to make the blockchain a programmable distributed credit infrastructure to support

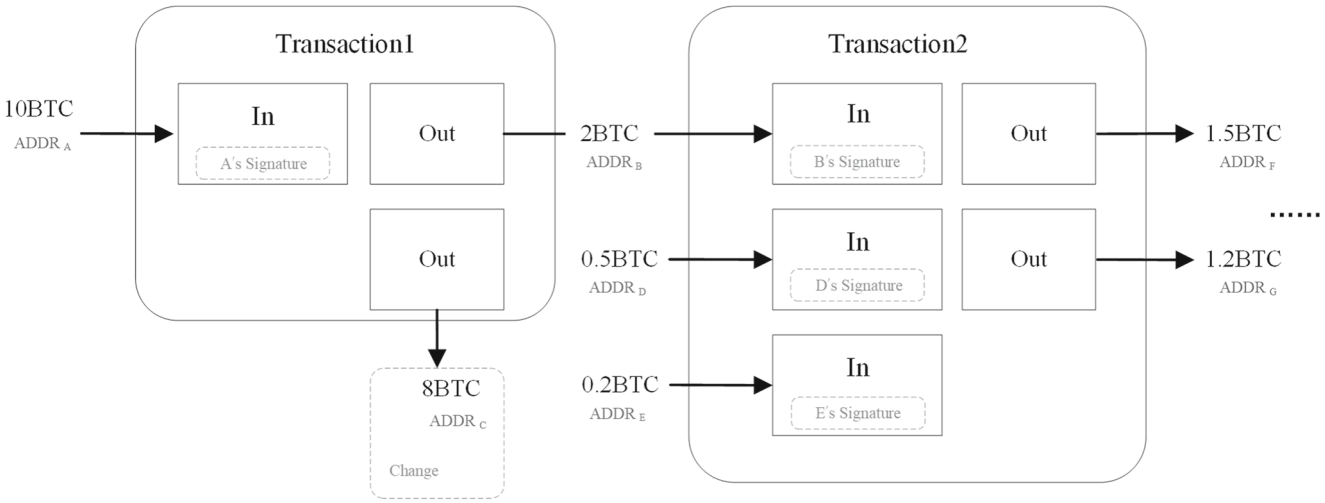

Fig. 1 An example of *Bitcoin* transactions between address A, B, C, D, E, F and G

smart contract applications [15–18]. Specifically, *Ethereum* not only uses the blockchain as a decentralized virtual currency and payment platform but also expands the technology of a decentralized market by increasing the expansibility function on the chain. The basic content includes real estate contracts, equity and debt obligations, intellectual property, etc.

A transaction in *Ethereum* refers to the signature packet of messages sent from an external account. It contains the receiver of the message, signature of the sender, *Ethereum* account balance, and data to be sent. The latter two values are called *STARTGAS* and *GASPRICE*. To prevent exponential explosion and infinite loops of code, each transaction requires a limit on the computational steps that are caused by the execution of the code, including the initial message and all messages raised in execution. *STARTGAS* is the limit, and *GASPRICE* is the cost of every step that needs to be paid to the miners.

Compared with the *Bitcoin* blockchain, the *Ethereum* blockchain provides the following three major improvements.

- *Ethereum*'s confirmation time is shorter. *Ethereum* block's confirmation time is approximately 14 s, much shorter than the 10 min required for *Bitcoin*.
- *Ethereum* have smaller blocks. In *Bitcoin*, the maximum size of the block is limited to 1M, whereas the block size in *Ethereum* is based on the complexity of smart contracts, called the *Gas* limit. The maximum size of a block depends on the situation. Currently, the largest block in *Ethereum* is approximately 1,500,000 *Gas*. A basic transaction or payment (not a smart contract) from one account to another consumes approximately 21,000 *Gas*. Therefore, each *Ethereum* block can probably contain 70 (1,500,000/21,000) transactions, whereas a *Bitcoin* block probably contains 1500 to 2000 transactions. Most *Ethereum* blocks are currently less than 2 KB.
- *Ethereum* can record more data. Since 2014, the *OP_Return* field has been added to the *Bitcoin* block. This field contains 80 bytes of space and is utilized to write data [6]. The *Ethereum* block contains a similar structure called *payload*, which is derived from the smart contract parameter. Users can fill in data in the corresponding field. The *payload* parameter in *Ethereum* can record at least 4000 characters, compared to the 80-byte limit in *Bitcoin*. Note that overlong data will lead to transaction failure.

Related work

Medical data warrant preservation because these data deal with life. Loss or tampering of medical data will lead to inconvenience of treatment, resulting in delay treatment progress, even endanger the patient's life.

Existing sensitive information and medical data-management systems are generally based on a group of centralized servers, which build a large site system or centralized relational database system [3]. The system usually adds a series of network and communication devices to ensure the client and core server can achieve normal interactions, which also increases the additional cost and complexity [19]. The probability of a highly centralized system failure is high. One link failure can lead to the collapse of the entire system. It also increases the likelihood of being hacked. Unpredictable problems to the user are likely if hackers modify a few fields in a database.

Some architectures use cloud storage to solve existing data preservation problems. The provable data possession (PDP) scheme is used to verify the integrity of outsourced

data without downloading such data [20–23]. Certificate management and key escrow problems, as well as ensuring privacy protection, are addressed in [24]. A new storage scheme that divides big data into sequenced parts and stores them among multiple cloud storage service providers to secure data is presented in [25]. However, issues of privacy, primariness, and security cannot be solved simultaneously. In particular, third parties should always be considered untrusted. Before data storage space is released or redistributed to other cloud users, it is also uncertain that relevant information will be completely cleared, whether stored in hard disk or in memory, to avoid data residues and leaking of sensitive information. The use of cloud storage also risks the loss of data due to server failure; for example, AWS (Amazon Web Services) had 4 h of downtime on February 28th, 2017, resulting in thousands of websites and applications becoming completely inaccessible [28].

At present, some organizations have started to study using blockchain to store data. T. McConaghy et al. [26] released a software called BigchainDB that has blockchain properties (e.g. decentralization, immutability, owner-controlled assets) and database properties (e.g. high transaction rate, low latency, indexing and querying of structured data). E. Gaetani et al. [27] delineate the importance of data integrity in cloud computing, then design a blockchain-based database which solves limitations of low throughput, high latency, and weak stability, for cloud computing environments.

The proposed DPS

This section presents an overview of the proposed DPS. Our basic idea is to introduce the concept of *proof of primitiveness of data* and use the unique data structure and de-centralization of blockchain to develop a data preservation system.

Preservation data analysis

In the process of diagnosis and treatment, people produce a large amount of medical data, including allergic history, assay data, anamnesis, and so on. Although each person produces various data in the process of diagnosis and treatment, the formats can be divided into type of text and multimedia. The proposed DPS supports rich preservation needs in different environments, different applications and different objects.

These data are vital for both individuals and doctors. The content must be true and complete when it is produced, and the data must be traceable and resistant to alteration, forgery, or deletion when validated. Because of its unique characteristics, medical data must also be safe and anonymous when stored. Medical data contain sensitive personal information, and thus prevention mechanisms must be established to prevent unconcerned or unauthorized staff from obtaining and extracting information. In addition, the data should be encrypted so that once the data are stolen, they cannot be understood without decryption.

System framework

Figure 2 illustrates a typical data preservation operation flow that includes two programs, namely the data access program and the blockchain interaction program. The first consists of four major operations, i.e., data submission, data manipulation, data query, and data verification. Users submit data for preservation, which is processed by the DPS. Users can query the preserved data and verify primitiveness. In the blockchain interaction program, the system stores unpreserved data in the blockchain and extracts the stored data back to the data access application. Preservation is transmitted to the blockchain network in the form of transactions, which miner nodes package and receive a reward for. The greater the amount of preservation, the larger the reward that miners will receive. Once a block is packaged, the preservation is stored forever in the blockchain.

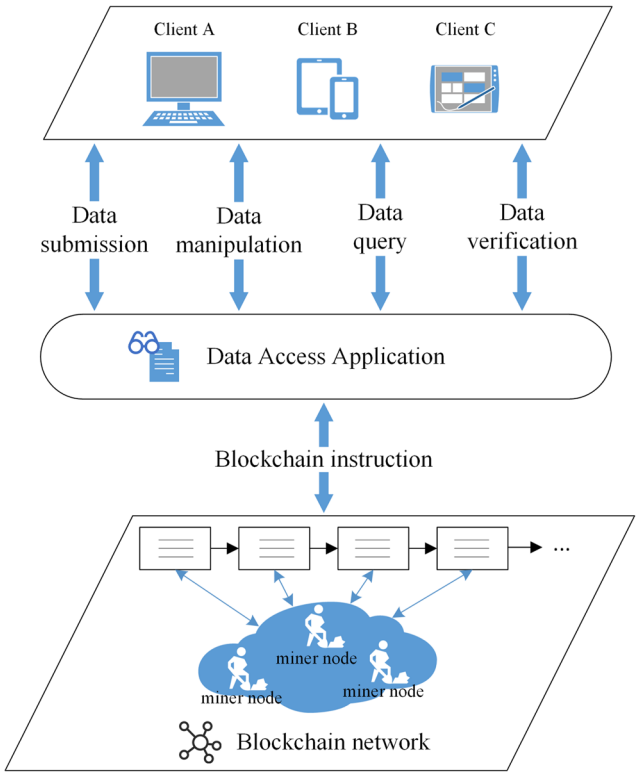

Fig. 2 Structure of preservation and blockchain interaction

We consider adding a digital signature to each interaction and verifying it after reception to avoid modification or replacement due to an unreliable Internet. We also assume that each user has a computer or other intelligent device connected to the Internet.

Threat model

There are four main threats to the proposed DPS.

Data modification or deletion

In all following possible situations, let us consider that the DPS preserves medical data, which involve diagnosis certificates and medical records. It is easy to tempt criminals to hurt others' body and health deliberately by modifying or deleting some of the data in the database [29]. In addition, the multimedia data are stored in a server that criminals have the possibility to operate illegally, such as modifying some data in a file or replace the original image.

Fraud by using false data

Taking into account the fact that preservation can appear to be a judicial credential, the following unfortunate situation might occur: two doctors perform a preservation operation for the same event but with slightly different or even opposite contents. Medical data can serve as proof of rights or judicial credentials, but if there are two pieces of different or conflicting data, how the proposed DPS identifies the real data is crucial.

Data interception

One of the major features of blockchain is transparency, which means that anyone can view the details of the blockchain transactions, including the data that the DPS preserves. Adversaries can obtain all preservations and take advantage of them. For example, Adversaries can know one person's physical characteristics or illness according to medical records, thereby threatening their life security [30].

User's privacy

Any user wallet address [31] is stored in the DPS, which could be a privacy issue for users.

Choice of blockchain platform

According to the analysis of blockchain in "Ethereum" and "Related work", we can conclude that there are two disadvantages of *Bitcoin* as an actual application:

- Slow confirmation: It takes approximately 10 min for a transaction to be verified by the whole net and approximately 1 h to receive safe confirmation.
- Small storage space: *Bitcoin* files are too small for developers to write data and cannot meet the space requirements needed for massive data.

Based on the comparison of *Bitcoin* and *Ethereum*, we choose to use *Ethereum* blockchain to implement the DPS. There are many third-party development platforms based on *Ethereum* blockchain in the Internet. Because the security and development quality of these platforms are unknown, we choose to use the original *Ethereum* blockchain instead of secondary development bases on other platforms. There are now more than 20,000 nodes [32] linked and synchronized with the original *Ethereum*, which ensures the security of the data.

Design goals

According to the requirements and threats discussed above, the proposed DPS should meet the following design goals:

Ensure the data are consistent with the user local's after submission

Unpreserved data are stored in the user's computer and need to be transferred to the DPS server. As quality and security cannot be guaranteed during transmission, the data packet may be lost or obtained by others, the contents may be subject to malicious tampering, or transmission failure may occur [33]. The preservation of these invalid or incorrect contents will not only increase the burden on the system but also cause loss of the user. Therefore, to ensure that the content the user submitted is identical to the content the DPS receives, the system must apply a digital signature algorithm and show the data to the user upon receipt. A careful check and confirmation are needed to avoid the occurrence of invalid preservation and content error. This part solves the "fraud by using false data" threat declared in "Threat model".

Prevent data from being tampered, forged or deleted

During the process of data preservation, users, system managers and notary parties have access to preserved data. Thus, the first task is to ensure that the data are tamper-proof. For example, with respect to wrong medical records, the doctors may use all available opportunities to modify the preserved data to avoid punishment by law. As another example, some enemies may take advantage of their job convenience or use large amounts of money to induce

system managers to modify real data in the system [34]. Some people, to avoid responsibility, may even fabricate data and delete or replace some information in a system with abandon. These actions will seriously hurt others' body and health. Thus, it is necessary to set up a guard mechanism so that data cannot be tampered with or deleted. Even if an illegal situation occurs, the system can thoroughly detect it. This part could address the "data modification or deletion" threat declared in "Threat model".

Anonymity and encryption

The preserved medical data may contain sensitive personal information. Therefore, the data must be anonymized, including the connections between the username and preserved contents, the filename of the stored file and the preserved data, and the username and personal information. In addition, the preserved multimedia data must be stored in the system in the form of files, which inevitably will be accessible to the outside world. To prevent illegal operators from performing unlawful operations, the preserved contents should be encrypted to ensure that readability and operability will be lost without decryption. The key should be properly stored to avoid encryption failure causing by leakage. This part could solve the "data Interception" and "user's privacy" threats declared in "Threat model".

Algorithm design

To satisfy the above requirements, we have implemented two contracts, called PS (*Preservation Submission*) and PV (*Primitiveness Verification*). First, the user uploads the text or files that need to be preserved during the *PS* phase. After confirming that the uploaded content is correct, the content will be stored in the DPS officially and cannot be changed. In the *PV* phase, the user can view the original content that has been saved or upload new files requiring verification of primitiveness. The DPS detects whether it has been changed and feeds the results back to the user automatically.

Because preservation in the blockchain entails some cost, the DPS restricts the permissions of user operations strictly; only users who are authenticated and successfully logged in to the DPS can operate on the preserved data. Unlogged users can only browse the already generated blockchain transaction. Because blockchain transactions are anonymized, viewing the transaction profiles and the encrypted contents will not have any impact on the preservation. This also achieves the goal of universal supervision and strengthens the credibility of the DPS.

Next, the pseudo-code of *PS* and *PM* will be presented with details.

Preservation submission (*PS*)

PS is mainly composed of three functions:

1. Receiving the unpreserved data uploaded by the user and processing it.
2. Encrypting the processed data for protection.
3. Storing the preservation in the blockchain.

Algorithm 1 *dataProcessing*() for receiving the unpreserved data uploaded by the user and process it

Input: The data to be preserved, the user information, description of the data to be preserved
if the format of data to be preserved is file **then**
    Generate random folders
    Store the file
    $hash_r \leftarrow$ calculate hash of the file
else if the format of data to be preserved is text **then**
    Store the text in database temporarily
    $hash_r \leftarrow$ calculate text hash
else
    Do nothing
end if
Generate a pair of asymmetrical key

Algorithm 2 *dataProtection*() for encrypting the processed data for protection

Input: User instructions
Output: Encrypted preservation
if the user confirms the preservation **then**
    Generate a symmetric key K_{sym}
    $C \leftarrow$ Use K_{sym} to encrypt the data to be preserved M
    $pubKey \leftarrow$ Get the public key in the asymmetric key generated in Algorithm 1
    $En_r \leftarrow$ Use *pubKey* encrypt K_{sym}
    return C
else
    Delete the temporary storage of data preservation
    return Null
end if

Algorithm 1 shows the pseudo-code for *dataProcessing*(), which receives the user's uploaded data for preservation and processes it. The DPS can accept data in two formats, a file type or a text type. If the preserved data are of the file type, the system uses the SHA-256 algorithm to calculate the $hash_r$ of the file, then processes the file and stores it in randomly generated folders. Data that are text files are stored in the database temporarily, and the hash of the text is calculated by using the SHA-256 algorithm. Finally, the ECC algorithm is used to generate a pair of asymmetric keys that will be used in Algorithms 2 and 4.

Algorithm 3 *writeInBlockchain*() for storing the preservation in the blockchain

Input: The encrypted perservation, hash
Output: Blockchain transaction hash
1: **if** The identity of the user is valid **then**
2: Store data E in the blockchain after recoding
3: $Tx \leftarrow$ Get blockchain transaction hash
4: **return** Tx
5: **else**
6: **return** Null
7: **end if**

Algorithm 2 shows the pseudo-code of *dataProtection*(), which encrypts the processed data for protection. This function is invoked when the user verifies the validity of the data to be preserved and will proceed to the next step. If the user confirms the preservation, the DPS uses the *AES* algorithm to generate a symmetric key K_{sym} to encrypt the unpreserved data. If the data to be preserved are files, the content M that needs to be encrypted is the location index of the file *index*. Content M can be divided into files M_f and text M_t, which are encrypted and converted into C_f and C_t.

$$C_f = AES(K_{sym}, M_f) \tag{1}$$

If it is text type, the content M is the original text.

$$C_t = AES(K_{sym}, M_t) \tag{2}$$

Next, the public key *pubKey* in the asymmetric key generated in Algorithm 1 is used to encrypt the K_{sym}. Using this multiple encryption strategy provides greater prevention against data leakage. Finally, the function returns the encrypted data. Users can also cancel the preserve operation, and the stored files or data in the database will be deleted.

Algorithm 3 shows the pseudo-code of *writeInBlockchain*(), which stores the preservation in the blockchain. First, the user's legitimacy is confirmed based on the user's registration information and past preserve operations and can only be set by the administrator. If the user is legal, the data to be preserved will be stored in the blockchain, and then the transaction hash *Tx* will be returned. The preserved contents E and implementation methods will be elaborated in "System process". If the user is on the blacklist, the DPS refuses to perform any operation.

User accounts are generated randomly and automatically when they register, and have nothing to do with user identity. So the input of each transaction is from the user's own account. The output of each transaction is to some accounts automatically generated by the system, which are managed by the system and used for collection.

Primitiveness verification (*PV*)

The *PV* module comprises three main functions.

1. Viewing the preserved data.
2. Verifying the consistency with the original data.
3. Extracting data from the blockchain.

All three of these functions are performed if the data have been preserved and the blockchain transaction hash is recorded.

Algorithm 4 *checkPreservedData*() for viewing the preserved data

Input: Private key, the blockchain transaction hash Tx
Output: Decrypted data
1: $E \leftarrow$ Extract data from blockchain according to Tx
2: $C \leftarrow$ Extract the encrypt data from E
3: $priKey \leftarrow$ Get the private key in the asymmetric key generated in Algorithm 1
4: $M \leftarrow$ Use *priKey* to decrypt En_K to decrypt C
5: $hash_r \leftarrow$ Extract the hash from E
6: $hash_c \leftarrow$ Calculate the hash of M
7: **if** $hash_r$ is consistent with $hash_c$ **then**
8: **return** M
9: **else**
10: **return** Null
11: **end if**

Algorithm 5 *validatePreservedData*() to verify the consistency with the original data

Input: Data to be verified, the blockchain transaction hash Tx
Output: A Boolean (True or False)
1: $hash_c \leftarrow$ Calculate the hash of the data to be verified U
2: $E \leftarrow$ Extract data from blockchain according to Tx
3: $C \leftarrow$ Extract the encrypt data from E
4: $hash_r \leftarrow$ Extract the hash from E
5: **if** $hash_r$ is consistent with $hash_c$ **then**
6: **return** TRUE
7: **else**
8: **return** FALSE
9: **end if**

Algorithm 6 *getChainData*() for extract data from the blockchain.

Input: The blockchain transaction hash Tx
Output: The parsed data
1: **if** the Tx is legal and exists **then**
2: Obtain all the information contained in the transaction according to the hash
3: Extract the stored data
4: Recode and parse
5: **return** Parsed data
6: **else**
7: **return** Null
8: **end if**

Algorithm 4 shows the pseudo-code of *checkPreservedData*(), which views the data that were stored in the blockchain by the DPS. First, the DPS retrieves the stored encrypted data from the blockchain by Tx. Then, the private key *priKey*, which is the asymmetric key generated in Algorithm 1, is used to decrypt the data C.

$$M = AES(ECC(priKey, En_K), C) \tag{3}$$

Next, the stored $hash_r$, which is the hash of the original data, is obtained, and the SHA-256 algorithm is used to calculate the $hash_c$ of the decrypted preserved files.

$$hash_c = SHA_{256}(M) \tag{4}$$

If $hash_r$ is consistent with $hash_c$, the stored file is not modified, and the decrypted preservation is returned. Otherwise, null is returned. The methods of data extraction in Algorithms 4 and 5 will be elaborated in Algorithm 6.

Algorithm 5 shows the pseudo-code of *validatePreservedData*(), which receives the data uploaded by the user and checks whether the data are identical with the preservation. Users upload the data that need to be verified to the DPS, which uses the SHA-256 algorithm to calculate the $hash_c$.

$$hash_c = SHA_{256}(U) \tag{5}$$

Next, the data stored in the blockchain are obtained from Tx, which contains the hash of the original data - $hash_r$, and proofread with $hash_c$. If they are consistent, the data to be verified are identical to the data preserved, and true is returned. Otherwise, the validated data have been modified, and false is returned.

Algorithm 6 shows the pseudo-code of *getChainData*(), which extracts and recodes data stored in the blockchain. If Tx is valid and exists, the DPS obtains the whole information of the transaction and extracts the data that were preserved previously. The data are then recoded, parsed into the original format, and returned. Otherwise, null is returned.

System process

Figure 3 illustrates the detailed system model of the proposed DPS. Two master procedures of preservation are identified. The first procedure commits unpreserved data when the user wants to secure the new data. The second procedure identifies the primitiveness of the data, which is divided into two main categories: the first is to view the data that have been preserved, and the second is to submit the data that need to be verified and check whether they are consistent with the content of the security. Each procedure involves an *Ethereum* blockchain operation, and thus each user has a wallet application operating on personal computers or smart phones. Next, the detailed process of each procedure will be presented.

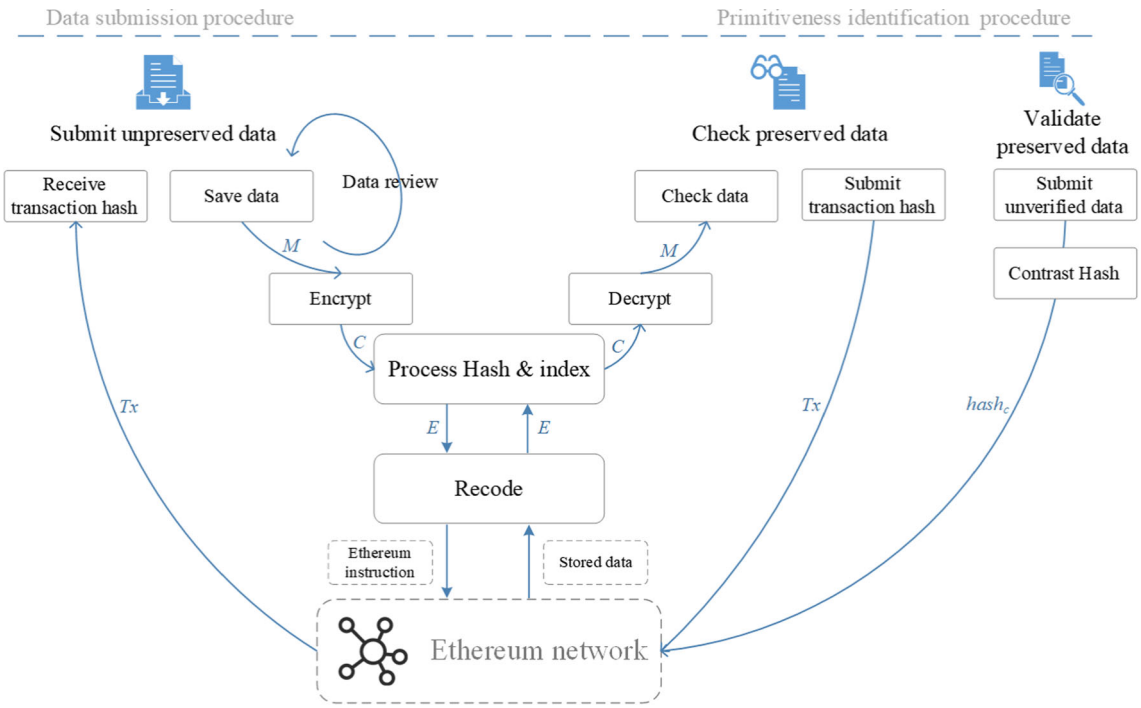

Fig. 3 Detailed block diagram of the proposed DPS

Data submission procedure

In the data submission procedure, the following four operational steps are executed.

Data submission and processing

The legitimate users detected by the DPS can submit data that they want to preserve, which can be either text or multimedia. Invoke the *dataProcessing*() procedure to save the multimedia data in a randomly generated folder or store the text in the database temporarily based on the type of data. Then generate a hash of data $hash_r$. To prevent the direct view of the content of multimedia data from being stolen, the DPS splits files into files of 1MB each and randomly scatters the files to different folders.

Data validation

Feedback on the user's uploaded data is provided, and a hash is generated as confirmation. This step ensures the accuracy of the data because the data can be lost or tampered with or fail to transfer in the communication process when transmitted over the Internet. Next, the *dataProtection*() program is invoked; the specific algorithm flow is shown in detail in Algorithm 2 in "Preservation submission (*PS*)". The asymmetric encryption algorithm uses Elliptic Curves Cryptography (ECC). The generated private key *priKey* will be processed and sent to the user's designated mailbox so that the user can decrypt and view the security content in the future.

Data storage

The *writeInBlockchain*() program is invoked to write the data that need to be preserved in the blockchain. Because the size of the field that can be written in the *Ethereum* blockchain is limited, the contents cannot be written intactly, and the storage mode and content are determined according to the type of preservation. On the one hand, if the data that need to be preserved are text type, the $hash_r$ and encrypted text C_t will be combined to E and then written in the blockchain directly.

$$E =< hash_r, C_t > \tag{6}$$

On the other hand, if the data are multimedia files, along with the $hash_r$, the encrypted index of the file location will be stored in the blockchain. Because the data are split into multiple files, there are multiple file location indexes, which are respectively encrypted. The indexes are then arranged in order and stored in the blockchain.

$$M_f =< index_i > (i = 1 \rightarrow n, n = files' \ amount) \tag{7}$$

$$E =< hash_r, C_f > \tag{8}$$

Cancel to preservation

The user can cancel the preserve operation at any time, and the text or file stored temporarily will be deleted. If a user has an unexpected operation, such as closing off the client or killing the process, the DPS automatically deletes any unpreserved text or file that has been stored temporarily for more than an hour.

Primitiveness identification procedure

There are two types of authentication methods in the primitiveness identification procedure, which will be described respectively in detail below.

Acquisition of preservation content

The user can directly view the contents of a previous preservation, which requires the user to provide the private key *priKey* generated when the data were preserved. First, the *getChainData*() program is executed to extract the data from the blockchain and parse the stored $hash_s$ and the encrypted preservation. Next, use *priKey* to decrypt the data to obtain the stored text or file location index. On one hand, the text is directly obtained from the blockchain, and thus the absence of tampering can be confirmed and displayed directly to the user. On the other hand, a multimedia file requires the reassembly of the stored files based on the location index, followed by the calculation of the $hash_c$. If $hash_s$ is consistent with $hash_c$, the contents of the file saved in the DPS have not been tampered with and can be displayed to the user.

Verify consistency

Users can verify whether the existing data are exactly identical to the preserved data. The user uploads the file that needs to be validated. Then, the DPS invokes the *validatePreservedData*() program to calculate the $hash_c$ and extract the data from the blockchain and parses the stored $hash_r$, which is compared to the $hash_c$ of the data to be validated. If the program returns *True*, the data are completely consistent, and the file uploaded by the user has not been modified. Otherwise, the DPS will warn users that the data are not exactly the same.

Security analysis

In this section, we prove the security of our protocols through the following theorems. Each theorem corresponds to one threat model in "Choice of blockchain platform".

Theorem 1 *Suppose adversaries can access the preserved data or files stored in the data server by other users. However, he/she cannot modify or delete the preservation and cannot view the saved files, and users will know if the files are destroyed. Thus, the data can be safely preserved.*

Proof The uploaded data M are actually stored in the blockchain, whose characteristics ensure that the preservation cannot be modified and that blockchain transactions cannot be deleted once confirmed. If an adversary wants to launch an attack to modify the data in the blockchain, extensive work is required to construct a new main chain, which is nearly impossible. In addition, all multimedia files M_f are stored in random locations with irregular filenames after being split, and the indexes of the file locations are encrypted and stored in the blockchain. In this case, it is almost impossible for lawbreakers to recover and view the contents of the original files. Furthermore, if the file is tampered with as M'_f, it can be verified by comparison with the $hash_r$ preserved in the blockchain. Then, the user can use the second method described in "Primitiveness identification procedure" to verify the primitiveness of the data, knowing $M_f \neq M'_f$ accordingly. □

Theorem 2 *Suppose adversaries attempt to preserve a piece of data that is different from the existing data to cover the real content. However, early data always exist and are more legally effective, and thus adversaries cannot use false data to defraud.*

Proof When there are two opposing security data, M and M', in general, the solution is to introduce a judicial system to adjudicate by obtaining evidence. However, our DPS does not deal with judicial evidence because it is out of the scope in terms of primitiveness proof. Actually, we encourage the preservation of data as early as possible because each security has a timestamp field that shows the time of preservation in the blockchain. As described in "Primitiveness identification procedure", we can learn the truth by viewing the raw data that were initially preserved. We advocate that users upload data that must be validated or are considered suspicious as the DPS can make more accurate judgments. Generally, we believe that data that have been preserved earlier are more legally effective. That is to say, if $time_M < time_{M'}$, M is legal. □

Theorem 3 *Suppose adversaries can pinpoint all blockchain transactions produced by the DPS and read the data. If he/she cannot understand the encrypted data, the adversaries will not be able to steal users' sensitive data.*

Proof As described in "Algorithm design", all data are encrypted and then stored, and thus no one can see the real contents of the preservation as long as the user protects his/her private key. In addition, because of the characteristic anonymity of the blockchain [35], all transactions are sent by an *Ethereum* address, which is independent of the user's personal information and cannot relate the preservation data to the individual. This demonstrates the anonymity of the DPS. □

Theorem 4 *Suppose adversaries can obtain all blockchain addresses of all users stored in the DPS. If the address is generated randomly and is not related to the user's operation, the user's privacy will not be threatened.*

Proof All blockchain addresses were originally public to the whole network because they were designed to be used to receive transactions. All transactions are automatically sent by the wallet application of *Ethereum*, and only users can send a transaction with their private key; consequently, users do not have to take any precautions. □

Performance evaluation

We evaluate the proposed DPS in terms of its operational cost. In particular, the total cost is equivalently estimated by measuring the total *Gas*, which can only be spent on one function involved in the progress, that is, storing the preservation in the blockchain (*writeInBlockchain*()). In other words, the cost will be generated only when the data are preserved and not when the data are verified. The amount of *Gas* will be determined when an *Ethereum* transaction is produced, e.g., a balance operation costs 20 *Gas* [36]. Table 1 shows the parameters of the computer used in this evaluation. In particular, we use an original implementation of the *Ethereum* protocol, *geth* [37], to send a transaction or deploy a contract since this tool is easy to operate and has the ability to automatically count the *Gas* amount. In the end, the total *Gas* amount is converted in

Table 1 Parameters of the computer used in this evaluation

Parameter	Value
Machine specs	MacBook Pro 2015 (OS:macOS 10.11, CPU: 2.2GHz quad-core Intel Core i7, RAM: 16GB of 1600MHz DDR3L
Ethereum client	Geth
Gas rate	$8.213 * 10^{-}6$ USD/gas

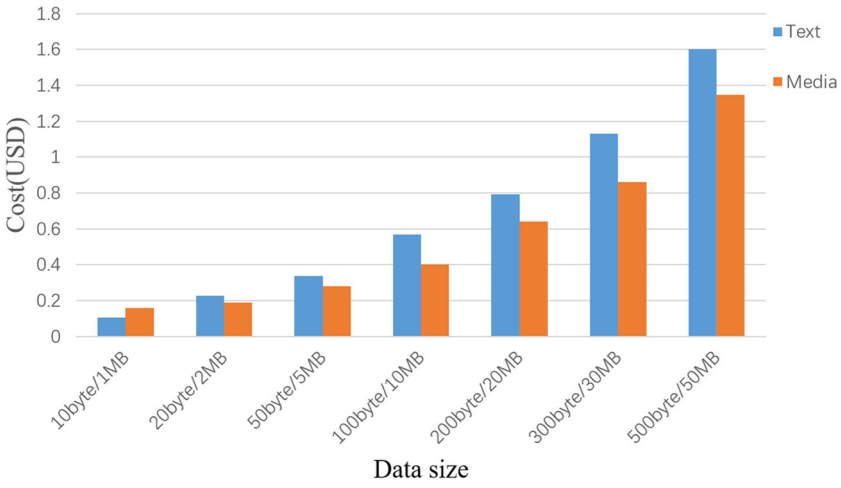

Fig. 4 Operation cost for a preservation in different data type in USD

USD by referring to the exchange rate chart *coinbase* [38] and *myetherwallet* [39]. At the time of evaluation, the rate was 1 Gas $= 2 \times 10^{-8}$ ETH $= 1.642 \times 10^{-6}$ USD.

A basic transaction requires 21,000 *Gas*. For every 100 bytes text or 10MB file added, it costs about 5400*Gas*. Figure 4 shows the total cost, in USD, to save data against the size of preservation. Because the length of the preserved data determines the *Gas* cost, the cost increases as the size of preservation increases. However, the obtained results shown in the chart illustrate that the total cost remains very low; even if the size of the multimedia file increases to 50 MB or the number of words increases to 500bytes, the total cost is less than US$2. It is important to note that this cost and the actual value of the preservation data are completely unrelated. Obviously, the proposed DPS is more applicable to relatively valuable data. In particular, in the proposed DPS, the multimedia data are split to store each 1 MB as a file, and the data are stored in the blockchain as is the location index, thus avoiding cost explosion due to the storage of large amounts of data. Figure 5 shows the concurrency performance for different preservation types and sizes. The figure illustrates that when the number of concurrent preservations exceeds 350, the response time will increase rapidly. In addition, the performance is related to the type of storage content; the file type is generally longer than the text type due to the division and storage operation of the file. The 500 bytes text and 50 MB multimedia files in the Fig. 4 are not the upper limit of storage capacity. In theory, the system can store files of unlimited size.

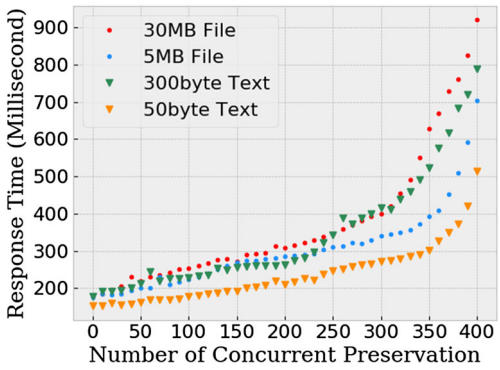

Fig. 5 Concurrency performance in different type and size

Discussion

The proposed DPS has a number of advantages compared to previous data preservation systems. The first is that it can effectively resist data tampering or deletion operations. The DPS can detect illegal operations of the data and notify users in time. This is very important because we use the blockchain technique, whose untampered features have been confirmed by many proofs. The second advantage is the anonymity of the DPS. We use a variety of cryptographic algorithms and file storage schemes to prevent the sensitive data stored by users from being leaked or stolen by illegal elements. Compared to cloud storage, the DPS implements permanent storage and untampering of data. At the same time, we avoid binding user identity information with medical information to reduce the possibility of privacy disclosure.

However, the current system still needs to be improved. There are some limitations to the proposed DPS. The first is related to the optimization of the structure of the data stored in the blockchain. Currently, when the DPS writes preservation data in the blockchain, each transaction contains only one security content. If the amount of data stored is small, some usable space will be wasted. A better

data structure and algorithm can be designed to write a number of different preservations in the blockchain each time in a feasible range of time and space to implement high concurrency.

The second limitation is related to image content recognition. The preserved multimedia data might be images. Let us consider the following possible situation: When an user executes the primitiveness verification program, the contents of the uploaded image are exactly the same in reality. However, the DPS mistakenly assumes that the data are tampered with because the image is compressed or re-shot so that the coding is inconsistent with the original one. Therefore, we should consider combining artificial intelligence and image content recognition techniques with the DPS to extract the core information to avoid the above "fake tampering" situation.

Conclusion

In this paper, we present a novel data preservation system (DPS) for medical data. This DPS provides a reliable storage solution to ensure the primitiveness and verifiability of stored data while preserving users' privacy. After illustrating the actual overall system requirements, we introduced a full-fledged protocol that allows users to store the preservation in the blockchain and to prove the primitiveness of the data. Compared to general electronic data preservation systems, an important advantage of the proposed DPS is that it can deal with situations in which data are easily lost and tampered with. Even if the data in the database have been tampered with or damaged, the data can be retrieved and verified through the blockchain. The protocol validation demonstrated the validity of our DPS. Based on the proposed protocol, the blockchain-based DPS was implemented on the *Ethereum* platform. The performance evaluation results showed that the cost of preservation with the proposed DPS is less than US$2, even if the size of the preserved files is 50 MB. When the number of concurrent preservations is 400 and all files are 30 MB, the response time of the system is not more than 1 s.

Funding Information This work was supported in part by the Guangxi Cooperative Innovation Center of cloud computing and Big Data (No.YD16E14), National Science Foundation of China under Grant 61602039, CCF-Venustech Open Research Fund and BIT-UMF research and development fund.

Compliance with Ethical Standards

Conflict of interests Mr. H. Li declares that he has no conflict of interest. Mr. Z. Lie declares that he has no conflict of interest. Mr. M. Shen declares that he has no conflict of interest. Mr. F. Gao declares that he has no conflict of interest. Ms. X. Tao declares that she has no conflict of interest. Mr. S. Liu declares that he has no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Lefeuvre, D., Pavillon, G., Aouba, A., Lamarche-Vadel, A., Fouillet, A., Jougla, E., and Rey, G., Quality comparison of electronic versus paper death certificates in France, 2010. *Popul. Health Metrics* 12(1):3. https://doi.org/10.1186/1478-7954-12-3, 2014.
2. Oskolkov, I., and Shishkov, R., Converting paper invoice to electronic form for processing of electronic payment thereof. Jan. 21 2014, US Patent 8,635,156. [Online]. Available: https://www.google.com/patents/US8635156.
3. Berman, F., Got data?: A guide to data preservation in the information age. *Commun. ACM* 51(12):50–56. https://doi.org/10.1145/1409360.1409376, 2008.
4. Miller, A., Juels, A., Shi, E., Parno, B., and Katz, J., Permacoin: Repurposing bitcoin work for data preservation. In: *2014 IEEE Symposium on Security and Privacy*, pp. 475–490, 2014.
5. Swan, M., *Blockchain: Blueprint for a New Economy*. Sebastopol: O'Reilly Media, Inc., 2015.
6. Wijaya, D. A., Extending asset management system functionality in bitcoin platform. In: *2016 International Conference on Computer, Control, Informatics and its Applications (IC3INA)*, pp. 97–101, 2016.
7. Nakamoto, S., Bitcoin: A peer-to-peer electronic cash system, 2008.
8. Raghav, S., and Saxena, A. K., Mobile forensics: Guidelines and challenges in data preservation and acquisition. In: *2009 IEEE Student Conference on Research and Development (SCOReD)*, pp. 5–8, 2009.
9. Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J. A., and Felten, E. W., Research perspectives and challenges for bitcoin and cryptocurrencies, vol. to appear, pp. 104–121, 2015.
10. Vukolić, M., The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication. In: Camenisch, J., and Kesdoğan, D. (Eds.). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-39028-4_9, 2016.
11. Gervais, A., Karame, G. O., Wüst, K., Glykantzis, V., Ritzdorf, H., and Capkun, S., On the security and performance of proof of work blockchains. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, ser. CCS '16*, pp. 3–16. New York: ACM, 2016. https://doi.org/10.1145/2976749.2978341.
12. Back, A., Hashcash - a denial of service counter-measure. In: *USENIX Technical Conference, Conference Proceedings*, 2002.
13. Pass, R., Seeman, L., and Shelat, A., Analysis of the blockchain protocol in asynchronous networks:643–673. https://doi.org/10.1007/978-3-319-56614-6_22, 2017.
14. Gao, F., Zhu, L., Shen, M., Sharif, K., Wan, Z., and Ren, K., A blockchain-based privacy-preserving payment mechanism for vehicle-to-grid networks. In: *IEEE Network*. https://doi.org/10.1109/MNET.2018.1700269, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8338177&isnumber=7593428, 2018.
15. Kosba, A., Miller, A., Shi, E., Wen, Z., and Papamanthou, C., Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In: *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 839–858, 2016.

16. Atzei, N., Bartoletti, M., and Cimoli, T., *A Survey of Attacks on Ethereum Smart Contracts (SoK)*, pp. 164–186. Berlin: Springer, 2017. https://doi.org/10.1007/978-3-662-54455-6_8.
17. Wood, G., Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper* 151:1–32, 2014.
18. Christidis, K., and Devetsikiotis, M., Blockchains and smart contracts for the internet of things. *IEEE Access* 4:2292–2303, 2016.
19. Wan, Z., Deng, R. H., and Lee, D., *Electronic Contract Signing Without Using Trusted Third Party*, pp. 386–394. Cham: Springer International Publishing, 2015. https://doi.org/10.1007/978-3-319-25645-0_27.
20. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., and Song, D., Provable data possession at untrusted stores. In: *Proceedings of the 14th ACM Conference on Computer and Communications Security, ser. CCS '07*, pp. 598–609. New York: ACM, 2007. https://doi.org/10.1145/1315245.1315318.
21. Shen, M., Ma, B., Zhu, L., Mijumbi, R., Du, X., and Hu, J., Cloud-based approximate constrained shortest distance queries over encrypted graphs with privacy protection. *IEEE Trans. Inf. Forensics Secur.* 13(4):940–953, 2018.
22. Zhu, L., Tang, X., Shen, M., Du, X., and Guizani, M., Privacy-preserving ddos attack detection using cross-domain traffic in software defined networks. *IEEE J. Sel. Areas Commun.* 36(3):628–643, 2018.
23. Shen, M., Wei, M., Zhu, L., and Wang, M., Classification of encrypted traffic with second-order markov chains and application attribute bigrams. *IEEE Trans. Inf. Forensics Secur.* 12(8):1830–1843, 2017.
24. He, D., Kumar, N., Wang, H., Wang, L., and Choo, K.-K. R., Privacy-preserving certificateless provable data possession scheme for big data storage on cloud. *Appl. Math. Comput.* 314(Supplement C):31–43, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0096300317304599.
25. Cheng, H., Rong, C., Hwang, K., Wang, W., and Li, Y., Secure big data storage and sharing scheme for cloud tenants. *China Communications* 12(6):106–115, 2015.
26. McConaghy, T., Marques, R., and Müller, A., Bigchaindb: A scalable blockchain database, 2016.
27. Gaetani, E., Aniello, L., Baldoni, R., Lombardi, F., Margheri, A., and Sassone, V., Blockchain-based database to ensure data integrity in cloud computing environments, 2017.
28. Summary of the amazon s3 service disruption in the northern virginia (us-east-1) region. https://aws.amazon.com/cn/message/41926/. Accessed 20 Aug 2017.
29. Bengtsson, S., and Solheim, B., Enforcement of data protection, privacy and security in medical informatics. *MEDINFO* 92:6–10, 1992.
30. Barnaby Jack Could Hack Your Pacemaker and Make Your Heart Explode. https://www.vice.com/en_ca/article/avnx5j/i-worked-out-how-to-remotely-weaponise-a-pacemaker. Accessed 11 May 2017.
31. Ethereum Project. https://www.ethereum.org/. Accessed 11 May 2017.
32. ethernodes.org - The ethereum node explorer. https://www.ethernodes.org/network/1. Accessed 31 Jan 2017.
33. Arent, L. M., Brownstone, R. D., and Fenwick, W. A., Ediscovery: Preserving, requesting & producing electronic information. *Santa Clara Computer & High Tech. LJ* 19:131, 2002.
34. Cooper, B. F., and Garcia-Molina, H., Bidding for storage space in a peer-to-peer data preservation system. In: *Proceedings 22nd International Conference on Distributed Computing Systems*, pp. 372–381, 2002.
35. Wang, H., He, D., and Ji, Y., Designated-verifier proof of assets for bitcoin exchange using elliptic curve cryptography. Futur. Gener. Comput. Syst., 2017.
36. Example transaction cost. http://ethdocs.org/en/latest/contracts-and-transactions/account-types-gas-and-transactions.html#example-transaction-cost. Accessed 31 Jan 2017.
37. ethereum/go-ethereum: Official go implementation of the ethereum protocol. https://github.com/ethereum/go-ethereum. Accessed 30 Jan 2017.
38. Bitcoin, ethereum, and litecoin price charts - coinbase. https://www.coinbase.com/charts. Accessed 11 May 2017.
39. Myetherwallet.com. https://www.myetherwallet.com/helpers.html. Accessed 11 May 2017.