Secure Data Storage and Recovery in Industrial Blockchain Network Environments

Wei Liang, Yongkai Fan, Kuan-Ching Li, *Senior Member, IEEE,* Dafang Zhang, and Jean-Luc Gaudiot, *Fellow, IEEE*

***Abstract*—The massive redundant data storage and communication in network 4.0 environments have issues of low integrity, high cost, and easy tampering. To address these issues, a secure data storage and recovery scheme in the blockchain-based network is proposed by improving the decentration, tampering-proof, real-time monitoring and management of storage systems, as such design supports the dynamic storage, fast repair, and update of distributed data in the data storage system of industrial nodes. A local regenerative code technology is used to repair and store data between failed nodes while ensuring the privacy of user data. That is, as the data stored are found to be damaged, multiple local repair groups constructed by vector code can simultaneously yet efficiently repair multiple distributed data storage nodes. Based on the unique chain storage structure, such as data consensus mechanism and smart contract, the storage structure of blockchain distributed coding not only quickly repair the nearby local regenerative codes in the blockchain but also reduce the resource overhead in the data storage process of industrial nodes. Experimental results show that the proposed scheme improves the repair rate of multi-node data by 9% and data storage rate increased by 8.6%, indicating to be promising with good security and real-time performance.**

***Index Terms*—Blockchain network, distributed storage, consensus mechanism, local regeneration code, repair rate**

I. Introduction

THE amount of data in our world is rapidly increasing with accelerated pace, and it is known that most of the mind-boggling amounts of digital data around the world have been generated in the past years [1]. With the rapid development of blockchain technology, an increasing number of users have adopted blockchain-based cloud services to store a massive amount of sensitive data [2][3]. Blockchain cloud storage service has brought convenience to the users, with the advantages of low cost and ease of management to data storage, despite it also brings issues related to low security and reliability.

This work was supported in part by the National Science Foundation of China under Grants 61572188, 61976087, the Scientific Research Program of the New Century Excellent Talents in Fujian Province University, Fujian Provincial Natural Science Foundation of China under Grant 2018J01570, and by the Hunan Provincial Natural Science Foundation under Grant 2016jj2058.

W. Liang is with College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China (e-mail:idlink@163.com)

Y. Fan is with Dept. of Computer Science and Technology, China University of Petroleum, Beijing 100024, China (e-mail:fanyongkai@gmail.com)

K.-C. Li is with Department of Computer Science and Information Engineering, Providence University, Taichung 43301, Taiwan (e-mail:kuancli@pu.edu.tw)(Correspongding Author)

D. Zhang is with College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China (e-mail:dfzhang@hnu.edu.cn)

J-L. Gaudiot is with Department of Electrical Engineering and Computer Science, University of California, Irvine, USA (e-mail: gaudiot@uci.edu)

In recent years, several disasters related to storage data loss have occurred, such as the downtime event of Amazon S3 in 2008 [4], the post-launch downtime event of Apple MobileMe[5], and the Google's Gmail incident in 2011. Statistics provided by Baidu Company indicate that approximately 100-200 nodes fail in a single storage cluster every day, and as of 2014, the failure rate is approximately 1-2% [6]. Still, with the increasing scale of cloud storage in-vehicle network, the probability of simultaneous failure on multiple storage nodes also increases [7][8][9].

The failure of the data node and consequent loss of data brings a considerable loss of data information to the entire distributed cloud storage system in the blockchain. Therefore, coding and repairing the cloud storage data will be essential to ensuring the reliability of the entire system. Whenever a blockchain system is harmed, researchers can use redundant data to repair the failure of the data stored in the node. As some nodes are invalid, the surviving storage nodes can recover the original data. The data redundancy repair features of blockchain cloud storage systems are: 1) as some of the nodes fail, the original file can still be reconstructed using the data present in the surviving nodes, and 2) the process to create adequate redundant data so that the failed nodes can be repaired. Despite the data can be reconstructed and repaired, the performance index of redundancy strategies must be measured, such as redundancy rate, repairing bandwidth (i.e., the amount of data that must be retrieved from the other surviving nodes to repair the data of failed nodes), and input/output. From this, the storage system for blockchain distributed data has become a popular topic in the data storage technology of the blockchain network[10][11].

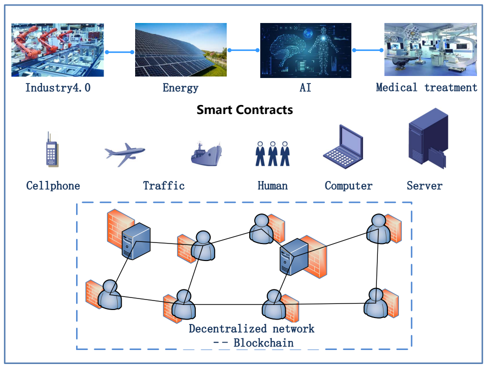


Fig. 1. Application of industrial blockchain network

Redundant data in the blockchain data storage scheme are cooperatively owned by the network nodes and can be managed and supervised online without being controlled or modified by any single node. Considering the security issues on the storage of distributed network data, Blockchain technology is not a single technology, but a new technology scheme that reliably stores and records data by combining several technologies [12][13]. Besides, Blockchain is a distributed database that cannot be arbitrarily tampered with. At this point, the term "distributed" represents the distribution not only of data storage but also the data recording process. Generally speaking, encrypting data file in a blockchain network ensures privacy and security, though whether the encrypted file has been maliciously modified, how to remain the file entirely surviving, and how to control the adequate sharing time of data are problems that must be solved urgently by using existing security sharing methods. At present, several ongoing researches in blockchain distributed storage systems are using blockchain technology for coding and repairing network data storage, as depicted in Fig. 1 applications of the Blockchain industrial network. Traditional encoding technologies must reconstruct the original data to produce new encoded data. Thus, whenever a node fails, the Blockchain-based network storage technology must download large amounts of data and ensure the reliability of network data through coding technology.

In order to solve such technical problems, this investigation proposes a secure data storage and recovery method based on Blockchain technology. First, this method evaluates whether the encrypted files in the storage system have been maliciously modified. The entire system includes network data acquisition, Blockchain storage, and security detection modules. The network data acquisition module is used for capturing network data packets and transfer them after signature to the Blockchain storage module, in which checks the signatures of the received network data packets, and the verified data packets are stored on the Blockchain. The security detection module is used to fetch data packets from the Blockchain storage module for security monitoring and early warning processing.

The remaining of this paper is structured as follows. Section II introduces the background of Blockchain storage technology, and the current research status of Blockchain storage technology is discussed in Section III. The mathematical model and fault-tolerant distributed storage structure are presented in Section IV, the fault-tolerant distributed storage and recovery algorithms are depicted in Section V, and the data authentication scheme is proposed in Section VI. Experimental comparison and comparative analysis are presented in Section VII, and finally, concluding remarks and future work are presented in Section VIII.

II. Related Work

With the rapid development of cloud storage and mass data processing technologies in Blockchain-based industrial technology, researches of reliable data storage technology are also under rapid advancement. To ensure the reliability and availability of large amounts of redundant data storage in the industrial Blockchain-based network, the storage system of network node data adopts strategies of "replication" and "erasure code" to generate redundant data information. However, most of these "replication" strategies must store large amounts of duplicated data to ensure high system reliability that leads to the problem of excessive-high storage cost. The "erasure code" strategy during network node repair requires the storage system to have a high network bandwidth, yet the bandwidth overhead is excessively large. Given the limitations of these redundancy strategies, regeneration, cooperative regeneration, local repair, and local regeneration codes have been successively proposed. Therefore, solving the problem of high-capacity data storage yet rapidly and accurately repairing industrial nodes in the network storage system of industrial Blockchain are critical issues in this field.

In recent years, network storage coding and applications have been extensively investigated in distributed data storage systems based on industrial Blockchain. Kamath et al. and Rawat et al. proposed the concept of local regenerative code [14][15] that can effectively repair node failures in storage systems, by significantly reducing disk I/O overhead during repair as well as achieving the best compromise between storage and bandwidth overhead.

Kamath et al. presented the upper bound of the minimum distance of local regenerative codes [16] and constructed MSR-LRC and MBR-LRC codes that can reach the upper bound of the minimum distance. Rawat et al. proposed the construction of the optimal local regeneration code based on the two-layer coding structure of Gabidulin and MSR (or MBR) codes. Though, the size of the finite field required for the local regeneration code increases exponentially with the number of nodes in the distributed storage system[17]. Silberstein et al. stated that MSR-LRC and MBR-LRC codes could obtain low disk I/O and repair bandwidth overheads [18], and particularly MSR-LRC codes can retrieve the local minimum storage overhead. In 2017, Gligoroski et al. constructed a type of local regeneration code based on HashTag code, namely, local regeneration HashTag code [19]. Failed nodes can be repaired through either local checking nodes or joint repair of local and global checking nodes. Nevertheless, when the local regenerative codes constructed above are used to repair the faults of multiple nodes in the local codes individually, the bandwidth performance cannot be improved [20].

Presently, the application of distributed storage technology based on industrial blockchain has attracted considerable attention [21][22][23]. A distributed storage network for industrial Blockchain is a reliable data security storage technology, in which many industrial network nodes participate in the recording and storing data. Various characteristics, such as data storage decentralization, tamper-proof, and traceability, can support people to solve problems of data security and reliability in industrial environments. The user's privacy in plain text-stored data is easily accessed by cloud service providers (CSPs), so industrial users aim to encrypt the industrial production data and submit to CSPs in ciphertext form. Yet, the use of cloud storage technology by industrial users will not only reduce the control of industrial data but also bring new challenges to data sharing. Therefore, secure file storage and sharing have become urgent problems in industrial cloud storage technology. Industrial users aim to achieve complete control of industrial data, including flexible access control strategy and controllable data sharing scope, on the premise to ensure the safety of industrial data. The user's privacy of industrial cloud-stored in plain text data will be easily accessed by CSPs, which cannot effectively guarantee data confidentiality, integrity, availability, and effective access control for encrypted files. During the file sharing, implementing a reasonable and effective control of the sharing population and the productive sharing time of files are problems that must be solved urgently on the existing methods of data security sharing in the industrial storage environment.

III. Concept and Model of Distributed Storage & Recovery

A. Mathematical Model

In a (n, k, d) regenerative code, the source file M is divided into n data blocks and coded into n nodes, each of which contains α blocks. The data receiver DC can recover the original data entirely through any k nodes. For a failed node, the new node can repair the node by connecting any $d(d \geq k)$ surviving nodes and downloading β blocks from each node, which processing is called node repair. The bandwidth consumed by node repair is named as repair bandwidth and expressed by γ and $\gamma = d\beta$.

The maximum flow minimum cut theorem [24] in graph theory shows that the maximum flow and the minimum cut capacity of all separated source nodes and data receivers are the same, so thus required to find the minimum cut. From the analysis, the corresponding capacity of the minimum cut is:

$$\sum_{i=0}^{k-1} \min\{\alpha, (d-i)\beta\} \tag{1}$$

For the source file M, if the minimum cut between the source node and the data receiver is greater than or equal to the size of the file, then the data receiver can obtain all the files successfully, so the parameters of the regeneration code must satisfy the following conditions:

$$M \leq \sum_{i=0}^{k-1} \min\{(d-i)\beta, \alpha\} \tag{2}$$

The lower bound of storage and repair bandwidth can be obtained by minimizing α and β in the process of data restoration, and thus, the following theorem holds.

Given the parameter (n, k, d), the minimum storage $a_{\min}$ and repair bandwidth of each node should satisfy the following equation[25].

$$\alpha_{mm} = \begin{cases} \frac{M}{k}, & \gamma \in [f(0), +\infty) \\ \frac{M - g(i)\gamma}{k-i}, & \gamma \in [f(i), f(i-1)) \end{cases} \tag{3}$$

in which

$$g(i) = \frac{(2d - 2k + i + 1)i}{2d} \tag{4}$$

$$f(i) = \frac{2Md}{(d-k+1)2k + (2k-i-1)i} \tag{5}$$

Here,$i = 1, 2, \cdots, k$. The code that obtains the minimum storage is called MSR code, and the code that obtains the minimum repair bandwidth called MBR code. Deriving from above, we have the following:

$$(\alpha_{MSR}, \gamma_{MSR}) = \left(\frac{M}{k}, \frac{M}{k}\frac{d}{d-k+1}\right) \tag{6}$$

$$(\alpha_{MBR}, \gamma_{MBR}) = \left(\frac{M}{k}\frac{2d}{2d-k+1}, \frac{M}{k}\frac{2d}{2d-k+1}\right) \tag{7}$$

In the following process of constructing a regenerative code, it is defined that each storage node contains three blocks, i.e., $\alpha = 3$. The first block of each node stores the original data (data blocks), the second block stores the redundant data (check blocks, for data repair and reliability protection), and the third block stores the timestamp (for fast find nodes). Assuming that $M = n$, since the regenerative code constructed next is also a linear network coding, the redundant data of its second block is a linear combination of system codes. Therefore, this coding scheme successfully reduces the additional network repair bandwidth caused by data repair, so that the data downloaded from each connected node is as small as possible.

The encoding and decoding operations of regenerative codes usually refer to operations in finite fields $GF(2^m)$. Moreover, the efficiency of encoding and decoding can be greatly improved as the special properties of Cauchy matrix. Thus, a definition of Cauchy matrix C [25] of order $n \times n$ is given as the following:

Let F be a finite field, $a_i, b_j \in F, a_i \neq b_j, i, j = 1, 2, \cdots, n$ and $a_i \neq a_j, b_i \neq b_j (i \neq j)$, then

$$C = (c_{ij})_{n\times n} = \left(\frac{1}{a_i - b_j}\right)_{n\times n} = \begin{pmatrix} \frac{1}{a_1-b_1} & \frac{1}{a_1-b_2} & \cdots & \frac{1}{a_1-b_n} \\ \frac{1}{a_2-b_1} & \frac{1}{a_2-b_2} & \cdots & \frac{1}{a_2-b_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{a_n-b_1} & \frac{1}{a_n-b_2} & \cdots & \frac{1}{a_n-b_n} \end{pmatrix} \tag{8}$$

The matrix C is called a Cauchy matrix of order $n \times n$.

As seen, it is not difficult to find that every sub-matrix of Cauchy matrix is a non-singular matrix, and there exists an inverse matrix. In addition, the time complexity of Cauchy matrix inversion $o(n^2)$ in finite field is much lower than that of Vandermonde matrix $o(n^3)$.

According to the properties of Cauchy matrix, the second block of each data storage node in Blockchain checks block, so it is constructed by linear transformation coding.

Let a matrix $U = (u_{ij})_{n\times n}$ of order $n \times n$ be a non-singular matrix in a finite field, the row vectors of the matrix defined as $U_i = \{u_{i1}, u_{i2}, \cdots, u_{in}\}, i = 1, 2, \cdots, n$ and let $U' = (U^{-1})^T$, where the row vectors of the matrix are defined as $U'_i = \{u'_{i1}, u'_{i2}, \cdots, u'_{in}\}, i = 1, 2, \cdots, n$. $(U^{-1})^T$ means the transpose of the inverse matrix of U, and the matrices U and U' satisfy

$$U_i U_j'^T = \begin{cases} 1, i = j \\ 0, i \neq j \end{cases} \tag{9}$$

Particularly U can be taken as a unit matrix. A matrix is defined as $V=(v_{ij})_{n\times n}$ that satisfies $\lambda V=U'C$, $1-\lambda^2\neq 0$, in which the row vectors of the matrix V are defined as $V_i=\{v_{i1},v_{i2},\cdots,v_{in}\},i=1,2,\cdots,n$. And, E represents a unit matrix of order $n\times n$.

Let the source file M be partitioned into n data blocks $X_i=\{x_{i1},x_{i2},\cdots,x_{in}\},i=1,2,\cdots,n$ from which n check blocks $Y_i=\{y_{i1},y_{i2},\cdots,y_{in}\},\quad i=1,2,\cdots,n$ are obtained as:

$$Y_i=X_1A_{11}+X_2A_{i2}+\ldots X_kA_{m},i=1,2,\cdots,n \tag{10}$$

in which

$$A_{ij}=V_i^TU_j+c_{ij}E,i,j=1,2,\cdots,n \tag{11}$$

Therefore, the encoding process of data storage can be represented by the following matrix transformation.

$$(X_1X_2\cdots X_n)\begin{pmatrix}E & 0 & \cdots & 0 & A_{11} & A_{21} & \cdots & A_{n1}\\ 0 & E & \cdots & 0 & A_{12} & A_{22} & \cdots & A_{n2}\\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & E & A_{1n} & A_{2n} & \cdots & A_{nn}\end{pmatrix}=(X_1X_2\cdots X_nY_1Y_2\cdots Y_n) \tag{12}$$

In this investigation, $B_i=\{X_i,Y_i,T_i\},where 1\leq i\leq n$ are represented as the coding vectors of node i. In a distributed storage system with n nodes, a codeword can be represented as $B=\{B_1,B_2,\cdots,B_n\}$ and the complete set of such codewords turn into coding. At the same time, let G be the encoding matrix of the regenerative code (n,k,d), and:

$$G=\begin{pmatrix}E & 0 & \cdots & 0 & A_{11} & A_{21} & \cdots & A_{n1}\\ 0 & E & \cdots & 0 & A_{12} & A_{22} & \cdots & A_{n2}\\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \vdots & E & A_{1n} & A_{2n} & \cdots & A_{nn}\end{pmatrix} \tag{13}$$

From the construction of G, it can be seen that any sub-matrix of order $n\times n$ of the matrix in Eq. 13 is a non-singular matrix; that is, there exists an inverse matrix. Therefore, when $r(1\leq r\leq k)$ nodes fail, it can be decoded by the data of other nodes to repair the damaged data and recover the source file M.

(1) When $r=1$, i.e., if one node fails, it is advisable to assume that node 1 fails. The source file can be recovered by locating the other $n-1$ nodes other than node 1 and downloading its first block $\{X_2,X_3,\cdots,X_n\}$ by the timestamp in third block of storage node, and downloading next its second block Y_j from any surviving node $j(j\neq 1)$. The repair process is shown in Eq. 14:

$$\begin{pmatrix}A_{j1}^T & A_{j2}^T & A_{j3}^T & \cdots & A_{jn}^T\\ 0 & E & 0 & \cdots & 0\\ 0 & 0 & E & \cdots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \cdots & E\end{pmatrix}^{-1}\begin{pmatrix}Y_j^T\\ X_2^T\\ X_3^T\\ \vdots\\ X_n^T\end{pmatrix}=\begin{pmatrix}X_1^T\\ X_2^T\\ X_3^T\\ \vdots\\ X_n^T\end{pmatrix} \tag{14}$$

(2) When $r=i\left(1<i\leq\left\lfloor\frac{n}{2}\right\rfloor\right)$, i.e. i nodes fail, it is assumed that node 1 to node i fails. The source file can be restored by locating the other $n-i$ nodes other than node 1 to i node and downloading its first block $\{X_{i+1},X_{i+2},\cdots,X_n\}$ by the timestamp in the third block of storage node, and then downloading the second blocks $Y_{j_1},Y_{j_2},\cdots Y_{j_i}$ from any surviving nodes $j_1,j_2,\cdots,j_i\,(j_l>k,l=1,2,\cdots,i)$. The repair process is shown in Eq. 15:

$$\begin{pmatrix}A_{j_11}^T & A_{j_12}^T & \cdots & A_{j_1i+1}^T & A_{j_1i+2}^T & \cdots & A_{j_1n}^T\\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\ A_{j_i1}^T & A_{j_i2}^T & \cdots & A_{j_ii+1}^T & A_{j_ii+2}^T & \cdots & A_{j_in}^T\\ 0 & 0 & \cdots & E & 0 & \cdots & 0\\ 0 & 0 & \cdots & 0 & E & \cdots & 0\\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\ 0 & 0 & \cdots & 0 & 0 & \cdots & E\end{pmatrix}\begin{pmatrix}Y_{j_1}^T\\ \vdots\\ Y_{j_i}^T\\ X_{i+1}^T\\ X_{i+2}^T\\ \vdots\\ X_n^T\end{pmatrix}=\begin{pmatrix}X_1^T\\ X_2^T\\ X_3^T\\ \vdots\\ X_n^T\end{pmatrix} \tag{15}$$

B. Blockchain-based Fault-tolerant Distributed Storage Structure

Distributed storage fault-tolerant structure of industrial Blockchain include namespace of a high fault-tolerant local code block, index record and mapping from code block to storage node. As shown in Fig. 2, the perception module is used to perceive whether the storage node has lost data and feedback the relevant information to the metadata information management module. When the client reads the file, it first retrieves the offset information of the file encoding block from the storage server. Taking into consideration that the use of system local codes to construct high fault-tolerant local codes, the original terminal data can be read directly from the Blockchain storage nodes.

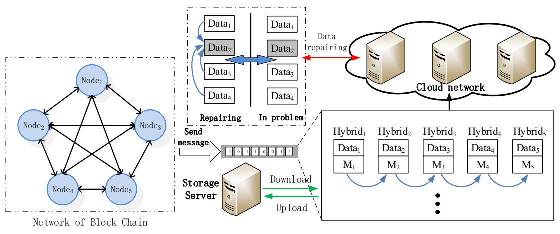


Fig. 2. Blockchain based fault-tolerantistributed storage structure

In order to improve the reliability and robustness of the complete industrial Blockchain storage system, a cache server is configured for each storage server. All high fault-tolerant local code encoded data is sent to the storage server and backed up in the cache server. Typically, the cache server is only responsible for the file backup cache. As the storage server is physically damaged, the cache server takes over the storage server and starts to work. Whenever the storage server fails, the cache server is used to reduce the delay overhead of data file reconstruction by increasing the storage overhead.

IV. Blockchain for Fault-tolerant Distributed Storage and Recovery Algorithm

A. Fault-tolerant Distributed Storage Algorithm

Multiple storage nodes are occasionally damaged due to a large number of nodes in the Blockchain cloud storage system, so local codes that can effectively repair the failed nodes must be designed. This investigation selects appropriate n and k based on the size of cloud storage system nodes in the Blockchain to construct a local code with a locality of (n, k). The usage of vector codes for constructing one local code with locality (n, k) is considered, so that the vector codes contain n discrete repair groups, and multiple failed nodes can be simultaneously repaired from k discrete local repair groups. Among them, the number of local repair groups determines the number of failed nodes that can be simultaneously repaired, as the number of nodes involved in the repair group determines the repair locality. On this basis, a local code with a locality of (n, k) can be constructed using Cauchy matrix and multidimensional linear vector codes.

First, the source file M is divided into n data blocks $\{X_1, X_2, \cdots, X_n\}$ and stored in nodes in which each node contains three blocks. The first block is a data block X_i, the second block is a check block $Y_i = X_1A_{i1} + X_2A_{i2} + \cdots X_kA_{in}$, and the third block is a timestamp block T_i, combined as a coding vector $B_i = \{X_i, Y_i, T_i\}, 1 \le i \le n$, so that fault-tolerant local regeneration code is constructed. Whenever one node fails, it is assumed that node 1 fails, so the source file M can be restored by Eq.14 that fastly locate the node other than node 1 and download its first block $\{X_2, X_3, \cdots, X_n\}$, and downloading next its second block Y_j by any surviving node $j (j \ne 1)$. Whenever $i \left(i \le \left\lfloor \frac{n}{2} \right\rfloor\right)$ nodes fail, the source file M can be recovered by quickly locating the other $n - i$ nodes and download the first block $\left\{X_{i_1}, X_{i_2}, \cdots, X_{i_{n-1}}\right\}$, and then download the second block $Y_{j_1}, Y_{j_2}, \cdots Y_{j_i}$ from any i surviving nodes $j_1, j_2, \cdots, j_i \, (j_l > i, l = 1, 2, \cdots, i)$.

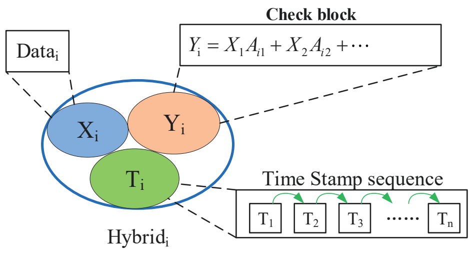


Fig. 3. Single fault-tolerant local regeneration code structure

Each regeneration code has three parts, namely, data domain X_i, check domain Y_i, and timestamp T_i, as shown in Fig.3. As soon as some node fails, the source file can be recovered by other surviving nodes, as depicted in algorithm 1 the construction of distributed storage regeneration code.

Aiming at the structure of fault-tolerant local regenerative codes in a Blockchain network, we design the corresponding information flow graph for the Blockchain network, as depicted in Fig. 4, since adjacent local codes can recover any fault local code in fault-tolerant local regeneration codes.

In industrial Blockchain networks, nodes upload data to the Blockchain network. When the data in the nodes are damaged, the data recovery process based on the blockchain network can recover completely the data of nodes by downloading any corresponding data X_i, Y_i, T_i from the surviving nodes.

B. Efficient BlockChain based Recovery Algorithm

1) Data Recovery for Single Node Failure: When constructing the regenerative codes, data storage and node repair are usually separated yet disclosed. When repairing the data in nodes, no encoding and decoding operation to download data, as also precise repair of a single failed node in the system by simple XOR operation. We identified that, by modifying the existing simple regenerative codes, we could retrieve the encoding form that can repair multiple failed nodes accurately at the same time. The repair process and bandwidth are given in detail. This new regenerative code coding method is simple, and its storage capacity is close to the minimum theoretical value of node storage. When specific parameters are satisfied, several nodes in the connection system can quickly repair multiple failed nodes, due to excellent local repairability and reduced overhead of disk input/output, so it has high practical application value.

Algorithm 1 Distributed storage regeneration code construction algorithm.

```
Input:
    Source file $M$, non-singular matrix $U, V$;
Output:
    Coding matrix $G$;
1: Divide souce file $M$ into $n$ data blocks $X_i = x_{i1}, x_{i2}, ..., x_{in}, i = 1, 2, ..., n$;
2: for $i \le i, j \le n$ do
3:     Compute $A_{ij} = (V_i)^T U_j + c_{ij}E$;
4: end for
5: Compute $Y_i = X_1A_{i1} + X_2A_{i2} + ... + X_kA_{in}$;
6: Record $T_i$ to get coding vector $B_i = X_i, Y_i, T_i$;
7: Generate code $B = B_1, B_2, ..., B_n$;
8: return  Coding matrix $G$.
```

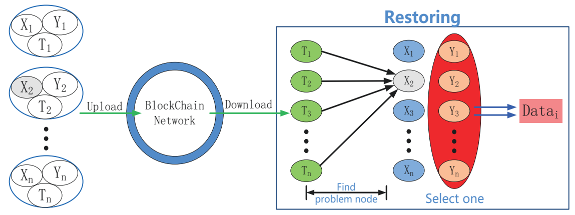


Fig. 4. Blockchain network corresponding to fault-tolerant local regeneration codes

When a new storage node is added to the blockchain cloud storage system, it is considered to distribute the new storage node evenly to each local code also to ensure that the constructed high fault-tolerant local regenerative code keeps the number of local codes unchanged with the increase of the length of each local code. If l storage nodes are added to the local code to reduce the bandwidth overhead on the construction of high fault-tolerant local regenerative code and the extended code, the information bits in the local code of the system are kept unchanged and the check bits increased to $l/2$. The extended code $(n_L + l/2, k, d'_{min} + l/2)$ corresponding to the local code (n_L, k, d'_{min}) of the system is constructed. At this time, the minimum distance of the extended code in the system is $d'_{min} + l/2$. Furthermore, the extended code of the system has a stronger ability to repair failed nodes.

Based on the system extended codes constructed above, the extended codes $(n_L+l/2, k, d'_{min}+l/2)$ of high fault-tolerant local regenerative codes are constructed by network coding to generate the distributed local check of local codes. At this time, the distributed local check also adds $l/2$ bits, and the local codes add l bits altogether.

Algorithm 2 Single node failure recovery algorithm.

```
Input:
    Coding matrix $G$, failure node 1;
Output:
    First block $X_1$ stored in node 1 ;
1: Locate other $n-1$ nodes other than node 1 by the third
   block of storage nodes quickly;
2: Download its first block $X_2, X_3, ..., X_n$;
3: Select any surviving node $j (j \neq 1)$ to download its second
   block $Y_j$;
4: Use formula (14) to recover the first block $X_1$ in node 1;
5: return First block $X_1$ in node 1.
```

When one node is damaged, such as node 1 is damaged, we will use the following method: finding the damaged node 1 through the blockchain network and undamaged nodes, then using all data except node 1 in the network and generating checking domain $Y_i (i \neq 2)$ of any other nodes, we can recover the data of node 1. The pseudocode is depicted in algorithm 2.

2) Data Recovery for Multiple Nodes Failure: Aimed at reducing the regeneration time and bandwidth overhead of fault node repair, it is investigated the selection of repair nodes for fault-tolerant local regeneration codes, that is, the optimal selection of repair nodes from surviving nodes and the transmission path from repair nodes to new nodes. Fig. 5 shows the repair process of multiple failure nodes, which includes the following steps:

Step 1: We construct a mathematical model of repair node selection in the heterogeneous cloud storage system, that is, minimize $\sum_{i=1}^{n} c_i \lambda_i$, in which c_i is the download cost of node i and $\{\lambda_1, \lambda_2, \mathrm{L}, \lambda_n\}$ is the download distribution of the selected repair node. When the fault node is located in the system local code part of each local code in the fault-tolerant local regeneration code, multiple local repair groups may be available for each fault node, and the repair group with the lowest repair cost is considered.

Step 2: We consider all constraints that must be obeyed in selecting repair nodes, that is, the repair and regeneration time is not less than the data transmission time from any repair node to the new node, the link bandwidth limits the transmission rate from the repair node to the new node, and the flow conservation on cloud storage nodes of the heterogeneous network.

Step 3: We find the optimal value of the function Minimize $\sum_{i=1}^{n} c_i \lambda_i$. When the scale of the cloud storage system of the industrial blockchain is small, the linear programming method is used to optimize the selection of new nodes, repair nodes, and data transmission paths to achieve the best selection. If the scale of the cloud storage system of the industrial blockchain expands, the computational complexity of the linear programming method increases dramatically. Moreover, heuristic selection algorithms, such as genetic, simulated annealing, and ant colony algorithms, are considered to obtain the approximate optimal new nodes, repair nodes, and their transmission paths for reducing the regeneration time and bandwidth overhead of fault node repair.

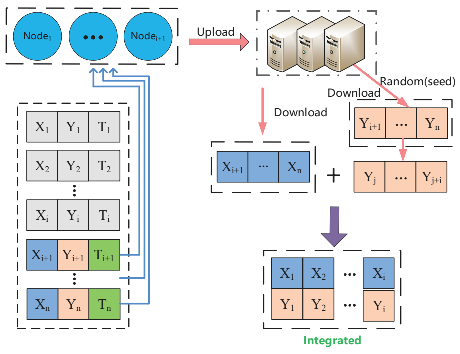


Fig. 5. Repair process of multiple failure nodes

When i nodes are damaged in the n industrial nodes, assuming that node 1 to node i fails is advisable. We can use the first blocks $X_{i+1}, X_{i+2}, \cdots, X_n$ of $n-i$ nodes other than node 1 to node i and the second blocks $Y_{j_1}, Y_{j_2}, \cdots Y_{j_i}$ of any surviving nodes $j_1, j_2, \cdots, j_i$ $(j_l > k, l = 1, 2, \cdots, i)$ to repair node 1 to node i. The pseudocode is depicted as algorithm 3.

Algorithm 3 Multiple nodes failure recovery algorithm.

```
Input:
    Coding matrix $G$, failure nodes $1-i$;
Output:
    First block $X_1, X_2, ..., X_i$ stored in node 1 to node $i$ ;
1: Locate other $n-1$ nodes other than node 1 by the third
   block of storage nodes;
2: Download its first block $X_{i+1}, X_{i+2}, ..., X_n$;
3: Select any surviving nodes $j_1, j_2, ..., j_i (j_l > k, l =
   1, 2, ..., i)$ to download its second blocks $Y_{j_1}, Y_{j_2}, ..., Y_{j_i}$;
4: Use formula (15) to recover the first blocks $X_1, X_2, ..., X_i$
   in node 1 to node $i$;
5: return First blocks $X_1, X_2, ..., X_i$ in node 1 to node $i$.
```

V. Data Integrity Verification Scheme

This section uses elliptic bilinear mapping to propose an integrity authentication scheme based on a third-party auditor (TPA). This scheme includes three entities: the users, CSPs, and TPA. The interaction process is as follows. The user stores the encrypted file M and digital tags into the CSP, and then the processed authentication metadata are sent to the TPA. Thus, the integrity challenges generated by the TPA are sent to the CSP, which then sends the integrity response to the TPA. Therefore, the TPA will send the comparative results between the integrity challenges and respond to the users. Finally, the CSP provides feedback to the user to judge whether data are complete or not.

The data integrity proof for blockchain is shown in Fig.6 and the detailed process is illustrated as follows:

1) Setup phase: (1) Generate a public-private key pair KeyGen $\left(1^k\right) \rightarrow (pk, sk)$: Let G and G' be the multiplicative cyclic group of a prime p, g be a generator of G and a bilinear mapping $f : G \times G \rightarrow G'$.Let $\varphi : \{0,1\}^* \rightarrow G$ be a hash function, which maps strings to G, and $\psi : G \rightarrow Z_p$ be another hash function, indicating that the elements of G will be mapped uniformly to Z_p. In addition, in order to a generate random challenge index S_j and the corresponding coefficient v_{s_j}, pseudo-random functions $f_{key}\{0,1\}^* \times K \rightarrow Z_p$ and pseudo-random permutations $\pi_{key} : \{0,1\}^* \times K \rightarrow \{0,1\}^{\log_2 n}$ are defined, which key belong to the key space K. For any $x \in Z_p$, u $\in$ G, and v $= g^x \in$ G, the user generates a pair of random signature key (SK, PK), and then the private key is $sk = (x, \mathrm{SK})$,the public key is $pk = (u, v, g, PK)$.

(2) Generate digital tag TagBlock $(pk, sk, m) \rightarrow T_m$. Initialize the data information file M, and divide it into blocks $X_1, X_2, \ldots, X_n$. Users randomly select an element id on Z_p for the file $M = \{X_1, X_2, \ldots, X_n\}$ and calculate its file tag $T = id \| Sig_{sk}(id)$. Then for each file block $X_i \in Z_p$, users generate a signature as follows:

$$\sigma_i = \left(\varphi(i) u^{X_i}\right)^x \in G, 1 \leq i \leq n \tag{16}$$

Finally, the user will send $\{M, \phi, T\}$ to CSP, and send $\{\phi, T\}$ to TPA as the authentication metadata, where $\phi = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$.

2) Challenge phase: Generate challenge information GenChal $(c, k, k') \rightarrow chal$: TPA receives the audit request from users and executes corresponding policies according to the data protection component of the request. The audit scheme management component will choose the appropriate audit scheme. When tasks are assigned to a specific TPA, the GCM module in TPA generates challenge information, $(c, k, k') \rightarrow chal$ where $c(1 \leq c \leq n)$ is the number of data blocks and $k \in Z_p$ and $k' \in Z_p$ are the random replacement keys generated for each TPM audit. Finally, it will be sent to CSP.

3) Data integrity verification phase: (1) Generating evidence $GenProof(pk, M, chal) \rightarrow V$: CSP receives the challenge $chal$ from TPA. After CSP confirms that is the challenge information, it is necessary to determine the subset $I = \{s_j\}, 1 \leq j \leq c$ of the challenge to be carried out in $[1, n]$ at first, and then calculate $S_j = \pi_k(j)$, $v_{s_j} = f_{k'}(j)(1 \leq j \leq c)$ by pseudo-random permutation π_{key} and pseudo-random function f_{key}. For any $i \in I$, CSP calculates the formula as follows:

$$\mu' = \sum_{i=s_1}^{s_c} v_i X_i, \quad \sigma = \prod_{i=s_1}^{s_c} \sigma_i^{v_i} \tag{17}$$

then the evidence $P = \{\mu', \sigma\}$ is sent to TPA.

When TPA receives the evidence sent by CSP, firstly, it protects its privacy in the data protection component, and randomly selects an element $r \in Z_p$, and uses the same random function $r = f_{k^*}(chal)$, in which the random function key k^* generated for each audit by TPM . Also, calculate:

$$R = u^\gamma \in G, \mu = \mu' + r\psi(R) \in Z_p \tag{18}$$

Finally, the evidence (μ, σ, R) is sent to TPA.

(2) Testing Evidence Check Proof $(sk, V) \rightarrow$ result. According to the generated challenge $chal$ and the authentication metadata base, make the integrity verification, that is, use the evidence (μ, σ, R) to calculate $s_j = \pi_k(j)$ and $v_{s_j} = f_{k'}(j)$, $1 \leq j \leq c$. Finally, the module is verified as follows.

$$f(\sigma, g) = f\left(\prod_{j=s_i}^{s_c}\left(\varphi(i)^{v_i} u^{\mu - r\psi(R)}\right), v\right) \tag{19}$$

If the Eq.19 holds, the integrity of the data is proved.

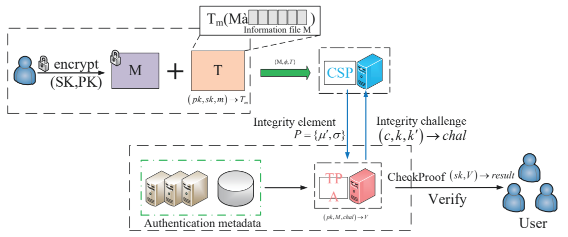


Fig. 6. The data integrity proof for blockchain

VI. Numerical Results

In the experiments, a Linux server with 8G memory and 2.4GHz CPU is utilized to implement the simulation in the industry 4.0 environment. The 51% attack method is used to simulate attacks on data storage in an industrial network. The simulated attack strength is set within the range of 10% to 70%. It is assumed that the industrial blockchain nodes are randomly distributed in an area of $100 \times 100m^2$. The nodes in this area include surviving, routing, and newly generated nodes. The blockchain can realize distributed accounting and decentration. This work uses blockchain technology to store the detailed record of each transaction in the industrial network. Network data can be added into the ledger in chronological order and stored as a series of blocks. Each block is linked to the front one as a chain. When the industrial nodes store data in the blockchain, the data integrity should be firstly verified. The ledger is distributed into multiple nodes. Each node stores a complete data copy. The blockchain automatically synchronizes and verifies the transactions of all nodes. The ledger is transparent to all members. A central institute or third party does not have to provide verification service. This section mainly compares and analyzes the performance in terms of security and the overhead of storage and recovery.

A. Overhead Evaluation

The resource overhead of data storage in Industry 4.0 can be used to determine whether the data storage and repairing function runs normally and stably. To ensure the repair quality of the regeneration code, three similar methods in this area are selected to evaluate the validity of the proposed method.

To balance the relationship between data storage and repair overhead in the industrial blockchain network, the node storage can be transformed into the storage method of hierarchical network coding. A fault-tolerant data storage and repair method is proposed based on the blockchain technology. The overhead performance is compared in Table 1. The data repair overhead is reduced to $n/2$ of the original, which illustrates the superiority in data storage and repair overhead. The blockchain-based data repair method can significantly reduce the repair overhead by comparing it with other methods.

Furthermore, the computation complexity, storage overhead, and application condition are compared. Results show that the proposed method is suitable for the energy-limited network. The performance in data storage and repair of multiple nodes is less than that of other methods [26][27][28].

TABLE I
THE COMPARISON OF OVERHEAD PERFORMANCE FOR SEVERAL METHODS

The repair method	Time complexity	Repair overhead	Storage overhead	Application condition
Ours	$o(1)$	$n/2$	n	$\forall n, k$
Literature [26]	$o(n)$	n	n	$\forall n, k$
Literature [27]	$o(k^2)$	n	n	$\frac{k}{n} \geq \frac{1}{2}$
Literature [28]	$o(1)$	n	$< \frac{3n}{2}$	$\forall n, k$

B. Data Recovery Performance Evaluation

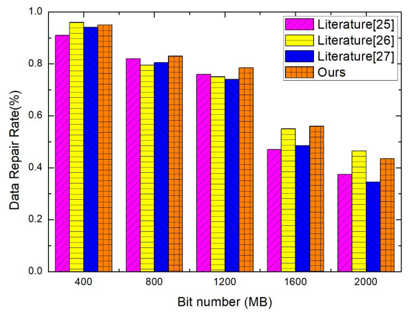
In Industrial Network 4.0, the application of the data storage algorithm is greatly related to the environment. Any blockchain-based industrial network has a high requirement for data security. The secure data storage in [26][27][28]is realized at the cost of computation resource. The identity of the industrial nodes should be initially authenticated. Then, the rate of secure data storage and data repair can be evaluated and compared, as shown in Fig. 7.

With the decrease in data storage rate and increase in data repair rate, the proposed method has improved real-time performance and security in data storage and repair. This finding proves the superiority in data storage and repair compared with that of [26][27][28]. Specifically, when the repaired file exceeds 2000 Mbit, secure data storage and repair rates are increased by 9% and 8.6%, respectively.

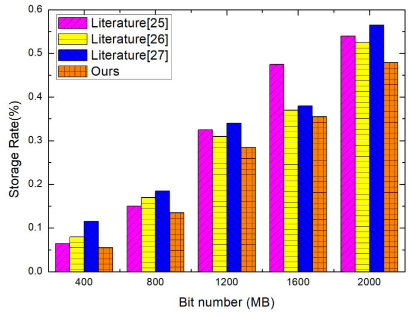
This section conducts experiments to evaluate the data integrity of nodes in Industrial Network 4.0, and Fig. 8 presents the results. Fig. 8(a) shows the integrity ratio of various data amounts and the distributed allocation. Fig. 8(b) displays that the data integrity ratio is affected when the distributed data storage system is illegally attacked. However, the integrity of the proposed scheme is better than the similar schemes due to the use of the decentralized storage structure. The decline of integrity ratio in the proposed scheme is slower than that in other comparative schemes. Hence, the damage of illegal attacks to the proposed scheme is less, thereby proving the enhanced security.

C. Ability against 51% of Attacks

In this section, we evaluate the ability against 51% of attacks. It is implemented by simplifying the complex storage

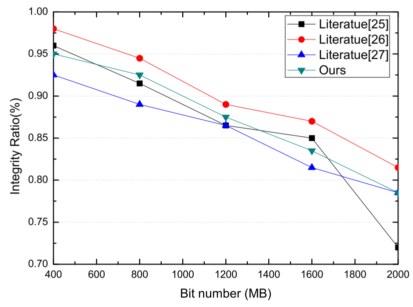


(a)

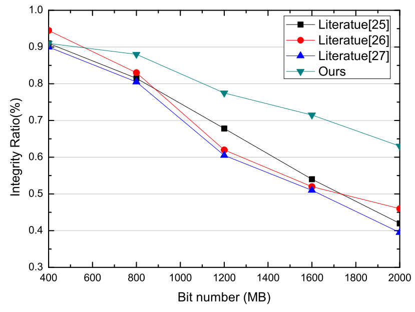


(b)

Fig. 7. The comparison of data repair rate and storage rate



(a)



(b)

Fig. 8. The comparison of data integrity for industrial nodes

proof procedure in the Blockchain system, which primarily includes the proof of work (POW), timestamp, etc. As shown in Fig. 9, the storage node is set as $\alpha = 3$. With the increase of stored data, the proposed algorithm achieves a higher false alarm rate than that of [21]. It is due to the use of a data storage and secure repair model. The high fault-tolerant regenerative coding is simple with good local repairability. With the increase of stored data, the algorithm can realize the parallel restoration of multiple data storage nodes when the data is damaged. Therefore, the proposed algorithm has good ability against 51% attacks.

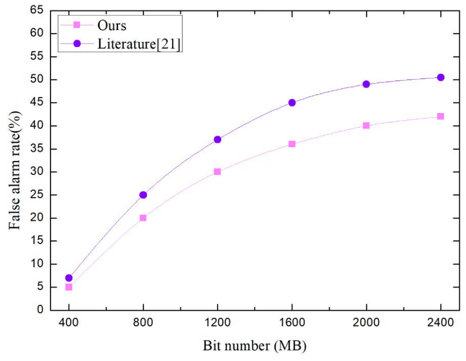


Fig. 9. Evaluation of the false alarm rate

VII. Conclusions

With the widespread and advances in Industrial Network 4.0, several emerging technologies will fundamentally impact how industrial data storage stores and connect with increased rates yet secure, as each technology represents a substantial opportunity to improve some aspect of the proposed research. Due to the particularity of blockchain-based industrial network, the data storage management faces enormous challenges. This work focuses on data security issues in the industrial network and designs a storage and repair scheme for fault-tolerant data coding. This scheme realizes a regeneration code with high precision and repairability in Industrial Network 4.0. The regeneration code has simple coding and excellent capability of local repair. When the stored data in a blockchain-based network are impaired, multiple nodes of data storage can be repaired with high efficiency. Also, the unique linked storage structure involving data consensus and the intelligent contract can be used to realize the fast local code storage of neighboring stored data in the blockchain-based network.Experiments show that the proposed scheme can reduce the repair overhead of local code in data storage and has good security and integrity.

References

[1] ScienceDaily. (2013) Big data, for better or worse: 90generated over last two years. [Online]. Available: https://www.sciencedaily.com/releases/2013/05/130522085217.htm

[2] Y. Xu *et al.*, "A blockchain-based non-repudiation network computing service scheme for industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 00, pp. 1–1, 2019.

[3] C. Yang, X. Chen, and Y. Xiang, "Blockchain-based publicly verifiable data deletion scheme for cloud storage," *Journal of Network and Computer Applications*, vol. 13, pp. 185–193, 2018.

[4] The Amazon S3 Team. (2008) Amazon s3 availability event. [Online]. Available: http: //status.aws.amazon.com/ s3-20080720.html

[5] M. Krigsman. (2013) Apple's mobileme experiences post-launch pain. [Online]. Available: http://www.zdnet.com/blog/projectfailures/apples-mobileme-experiences-post-launch-pain/908

[6] Y. Yu, Y. Li, and J. Tian, "Blockchain-based solutions to security and privacy issues in the internet of things," *IEEE Wireless Communications*, vol. 25, no. 6, pp. 12–18, 2018.

[7] D. Ford *et al.*, "Availability in globally distributed storage systems," in *9th USENIX conference on Operating Systems Design and Implementation (OSDI)*, Vancouver, BC, Canada, 2010, pp. 61–74.

[8] K. Hwang, J. Dongarra, and C. F. Geoffrey, *Distributed and Cloud Computing: From Parallel Processing to the Internet of Things*, Y. Cheng, Ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2012.

[9] W. Liang and M. Tang, "A secure fabric blockchain-based data transmission technique for industrial internet-of-things," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3582–3592, 2019.

[10] L. Jiang *et al.*, "Blockchain empowered wireless power transfer for green and secure internet of things," *IEEE Network Magazine*, 2019.

[11] H. Dai, Z. Zhen, and Y. Zhang, "Blockchain for internet of things: A survey," *IEEE Internet of Things*, 2019.

[12] K. Gai *et al.*, "Permissioned blockchain and edge computing empowered privacy-preserving smart grid networks," *IEEE Internet of Things Journal*, 2019.

[13] W. Liang *et al.*, "Tbrs: A trust based recommendation scheme for vehicular cps network," *FGCS*, vol. 92, no. 0, pp. 383–398, 2019.

[14] G. M. Kamath *et al.*, "Codes with local regeneration and erasure correction," *IEEE Transactions on Information Theory*, vol. 60, no. 8, pp. 4637–4660, 2014.

[15] A. S. Rawat *et al.*, "Codes with local regeneration and erasure correction," *Optimal locally repairable and secure codes for distributed storage systems*, vol. 60, no. 1, pp. 212–236, 2014.

[16] G. M. Kamath *et al.*, "Codes with local regeneration," in *IEEE International Symposium on Information Theory Proceedings (ISIT)*, Istanbul, 2013, pp. 1606–1610.

[17] A. S. Rawat *et al.*, "Optimal locally repairable codes with local minimum storage regeneration via rank-metric codes," in *Information Theory and Applications Workshop (ITA)*, San Diego, CA, 2013, pp. 10–15.

[18] N. Silberstein, A. S. Rawat, and S. Vishwanath, "Error-correcting regenerating and locally repairable codes via rank-metric codes," *IEEE Transactions on Information Theory*, vol. 61, no. 11, pp. 5765–5778, 2015.

[19] D. Gligoroski *et al.*, "Repair duality with locally repairable and locally regenerating codes," in *The 3rd IEEE International Conference on Big Data Intelligence and Computing (IEEE DataCom 2017)*, Orlando, Florida, USA, 2017, p. 6.

[20] J. L. X. Li, J. Niu and W. Liang, "Cryptanalysis of a dynamic identity based remote user authentication scheme with verifiable password update," *International Journal of Communication Systems*, vol. 28(2), pp. 374–382, 2015.

[21] W. Liang *et al.*, "An industrial network intrusion detection algorithm based on multi-feature data clustering optimization model," *IEEE Transactions on Industrial Informatics*, vol. 1, no. 1, 2019.

[22] L. Zhou *et al.*, "Beekeeper: A blockchain-based iot system with secure storage and homomorphic computation," *IEEE Access*, vol. 6, pp. 43 472–43 488, 2018.

[23] A. Reyna *et al.*, "On blockchain and its integration with iot. challenges and opportunities," *Future Generation Computer Systems*, vol. 88, pp. 173–190, 2018.

[24] D. Chen *et al.*, "An interactive image segmentation method in hand gesture recognition," *Sensors*, vol. 17, no. 2, pp. 539–550, 2017.

[25] X. Xie *et al.*, "A multi-node exact repair method in cloud storage based on interference alignment," *Acta Electronica Sinica*, vol. 42, no. 10, pp. 1873–1881, 2014.

[26] Y. Ren *et al.*, "Blockchain-based trusted electronic records preservation in cloud storage," *Computers, Materials & Continua*, vol. 58, no. 1, pp. 135–151, 2019.

[27] J. T. et al., "Date hierarchical storage strategy for data disaster recovery," *IEEE Access*, pp. 1–1, 2018.

[28] Y. K. et al., "Efficient local secret sharing for distributed blockchain systems," *IEEE Communications Letters*, vol. 23, no. 2, pp. 282–285, 2019.

Wei Liang is currently an Associate Professor at the College of Computer Science and Electronic Engineering, Hunan University, China. He received his Ph.D. degree at Hunan University in 2013 and was a postdoctoral scholar at Lehigh University, USA, during 2014-2016. He served as Application Track Chair of IEEE Trustcom 2015, a Workshop Chair of IEEE Trustcom WSN 2015 and IEEE Trustcom WSN 2016. He has published more than 110 journal/conference papers such as IEEE Transactions on Industrial Informatics, IEEE Transactions on Emerging Topics in Computing, IEEE Transactions on Computational Biology and Bioinformatics, and IEEE Internet of Things Journal. His research interests include Blockchain security technology, Networks Security Protection, embedded system and Hardware IP protection, Fog computing, and Security management in WSN. He is a Member of the IEEE.

Yongkai Fan received the Bachelor, Master and Ph.D. degrees from Jilin University, Changchun, China, in 2001, 2003, 2006, respectively. From 2006 to 2009, he was an Assistant Researcher at Tsinghua University, China. Since 2009, he is an Assistant Professor at China University of Petroleum, China. Prof. Fan's research interests include theories of software engineering and software security.

Kuan-Ching Li is currently a Distinguished Professor at Providence University, Taiwan, where he also serves as the Director of the High Performance Computing and Networking Lab. He is a recipient of awards and funding support from several agencies and industrial companies, as he also received distinguished chair professorships from universities in China and other countries. Professor Li published more than 250 scientific papers and articles and is co-author or co-editor of more than 20 books published by Taylor & Francis, Springer, and McGraw-Hill. Professor Li's research interests include parallel and distributed computing, Big Data, and emerging technologies. He is a senior member of the IEEE and a fellow of the IET.

Dafang Zhang is a professor in the college of computer science and Electronic Engineering, Hunan University, China. He received a Ph.D. degree in applied mathematics from Hunan University in 1997. Prof. Zhang was a visiting fellow in Regina university, Canada during 2002-2003, and senior visiting fellow in Michigan state university, USA in 2013. He has published more than 230 journal/conference papers and PI for more than 30 large scale scientific projects. Prof. Zhang's research interests include dependable systems/networks, network security, network measurement, hardware security and IP protection.

Jean-Luc Gaudiot received the Diplôme d'Ingénieur from the École Supérieure d'Ingénieurs en Electronique et Electrotechnique, Paris, France in 1976 and the M.S. and Ph.D. degrees in Computer Science from UCLA in 1977 and 1982, respectively. He is currently a Distinguished Professor in the Department of Electrical Engineering and Computer Science at UC Irvine. Prior to joining UCI in 2002, he was Professor of Electrical Engineering at the University of Southern California since 1982. His research interests include multithreaded architectures, fault-tolerant multiprocessors, and implementation of reconfigurable architectures. He has published over 250 journal and conference papers. His research has been sponsored by NSF, DoE, and DARPA, as well as a number of industrial companies. He has served the community in various positions and was the President of the IEEE Computer Society in 2017.