# A Three-Layer Privacy Preserving Cloud Storage Scheme Based on Computational Intelligence in Fog Computing

Tian Wang, Jiyuan Zhou, Xinlei Chen, Guojun Wang, Anfeng Liu, and Yang Liu, *Member, IEEE*

***Abstract*—Recent years witness the development of cloud computing technology. With the explosive growth of unstructured data, cloud storage technology gets more attention and better development. However, in current storage schema, user's data is totally stored in cloud servers. In other words, users lose their right of control on data and face privacy leakage risk. Traditional privacy protection schemes are usually based on encryption technology, but these kinds of methods cannot effectively resist attack from the inside of cloud server. In order to solve this problem, we propose a three-layer storage framework based on fog computing. The proposed framework can both take full advantage of cloud storage and protect the privacy of data. Besides, Hash-Solomon code algorithm is designed to divide data into different parts. Then, we can put a small part of data in local machine and fog server in order to protect the privacy. Moreover, based on computational intelligence, this algorithm can compute the distribution proportion stored in cloud, fog, and local machine, respectively. Through the theoretical safety analysis and experimental evaluation, the feasibility of our scheme has been validated, which is really a powerful supplement to existing cloud storage scheme.**

***Index Terms*—Cloud computing, cloud storage, fog computing, privacy protection.**

## I. Introduction

SINCE the 21st century, computer technology has developed rapidly. Cloud computing, an emerging technology, was first proposed in SES 2006 (Search Engine Strategies 2006) by San Jose and defined by NIST (National Institute of Standards and Technology) [1]. Since it was proposed, cloud computing has attracted great attention from different sectors of society. Cloud computing has gradually matured through so many people's efforts [2]. Then there are some cloud-based technologies deriving from cloud computing. Cloud storage is an important part of them.

With the rapid development of network bandwidth, the volume of user's data is rising geometrically [3]. User's requirement cannot be satisfied by the capacity of local machine any more. Therefore, people try to find new methods to store their data. Pursuing more powerful storage capacity, a growing number of users select cloud storage. Storing data on a public cloud server is a trend in the future and the cloud storage technology will become widespread in a few years. Cloud storage is a cloud computing system which provides data storage and management service. With a cluster of applications, network technology and distributed file system technology, cloud storage makes a large number of different storage devices work together coordinately [4], [5]. Nowadays there are a lot of companies providing a variety of cloud storage services, such as Dropbox, Google Drive, iCloud, Baidu Cloud, etc. These companies provide large capacity of storage and various services related to other popular applications, which in turn leads to their success in attracting humorous subscribers. However, cloud storage service still exists a lot of security problems. The privacy problem is particularly significant among those security issues. In history, there were some famous cloud storage privacy leakage events. For example, Apples iCloud leakage event in 2014, numerous Hollywood actresses private photos stored in the clouds were stolen. This event caused an uproar, which was responsible for the users' anxiety about the privacy of their data stored in cloud server.

As shown in Fig. 1, user uploads data to the cloud server directly. Subsequently, the Cloud Server Provider (CSP) will take place of user to manage the data. In consequence, user do not actually control the physical storage of their data, which results in the separation of ownership and management of data [6]. The CSP can freely access and search the data stored in the cloud. Meanwhile the attackers can also attack the CSP server to obtain the user's data. The above two cases both make users fell into the danger of information leakage and data loss. Traditional secure cloud storage solutions for the above problems are usually focusing on access restrictions or data encryption. These methods can actually eliminate most part of these

Manuscript received June 14, 2017; revised September 25, 2017; accepted October 7, 2017. Date of current version January 19, 2018. This work was supported in part by the National Natural Science Foundation of China under Grants 61672441, 61472451, and 61632009, in part by the Guangdong Provincial Natural Science Foundation under Grant 2017A03030800, in part by the High Level Talents Program of Higher Education in Guangdong Province under Grant 2016ZJ01, and in part by the Foster Project for Graduate Student in Research and Innovation of Huaqiao University under Grant 1611414016. *(Corresponding author: Guojun Wang.)*

T. Wang, J. Y. Zhou, and X. L. Chen are with the Department of Computer Science and Technology, Huaqiao University, Xiamen 361021, China (e-mail: cs_tianwang@163.com; zhoujiyuan1994@foxmail.com; adamwt@163.com).

G. Wang is with the School of Computer Science and Educational Software, Guangzhou University, Guangzhou 510006, China (e-mail: wsnman@gmail.com).

A. F. Liu is with the School of Information Science and Engineering, Central South University, Changsha 410083, China (e-mail: afengliu@mail.csu.edu.cn).

Y. Liu is with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: liu.yang@bupt.edu.cn).

Digital Object Identifier 10.1109/TETCI.2017.2764109

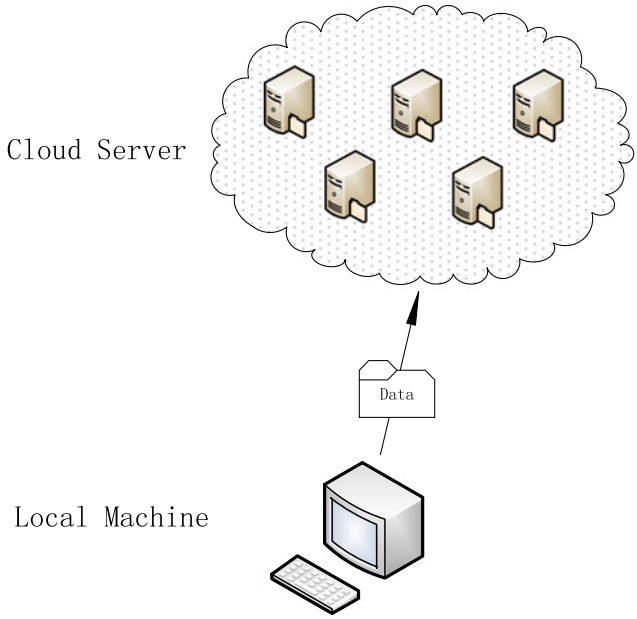


Fig. 1. Traditional cloud storage structure.

problems. However, all of these solutions cannot solve the internal attack well, no matter how the algorithm improves. Therefore, we propose a TLS scheme based on fog computing model and design a Hash-Solomon code based on Reed-Solomon code [7], [8]. Fog computing is an extended computing model based on cloud computing which is composed of a lot of fog nodes. These nodes have a certain storage capacity and processing capability. In our scheme, we split user's data into three parts and separately save them in the cloud server, the fog server and the user's local machine. Besides, depending on the property of the Hash-Solomon code, the scheme can ensure the original data cannot be recovered by partial data. On another hand, using Hash-Solomon code will produce a portion of redundant data blocks which will be used in decoding procedure. Increasing the number of redundant blocks can increase the reliability of the storage, but it also results in additional data storage. By reasonable allocation of the data, our scheme can really protect the privacy of user' data. The Hash-Solomon code needs complex calculation, which can be assisted with the Computational Intelligence (CI). Paradigms of CI have been successfully used in recent years to address various challenges, for example, the problems in Wireless sensor networks (WSNs) field. CI provides adaptive mechanisms that exhibit intelligent behavior in complex and dynamic environments like WSNs [9]. Thus in our paper, we take advantage of CI to do some calculating works in the fog layer. Compared with traditional methods, our scheme can provide a higher privacy protection from interior, especially from the CSPs. The remainder of this paper is organized as follows: Section II reviews related research work, Section III detailedly elaborates the TLS architecture, the Implementation detail of work flow, the theoretical safety analysis of the storage scheme and the efficiency analysis proposed in this paper, Section IV evaluates the scheme by different experiments and Section V concludes this paper at last.

## II. Related Works

The importance of security in cloud storage has attracted a lot of attention no matter in academe or industry. There are a lot of researches about secure cloud storage architectures in recent years. In order to solve the privacy issue in cloud computing, paper [10] proposed a privacy-preserving and copy-deterrence CBIR scheme using encryption and watermarking techniques. This scheme can protect the image content and image features well from the semi-honest cloud server, and deter the image user from illegally distributing the retrieved images. Shen *et al.* think cloud is semi-trusted and propose a framework for urban data sharing by exploiting the attribute-based cryptography. The scheme they proposed is secure and can resist possible attacks [11]. Fu *et al.* propose a content-aware search scheme, which can make semantic search more smart. The experiments results show that their scheme is efficient [12].

In paper [13], Hou, Pu and Fan consider that in traditional situation, user's data is stored through CSP, even if CSP is trustworthy, attackers can still get user's data if they control the cloud storage management node. To avoid this problem, they propose an encrypted index structure based on an asymmetric challenge-response authentication mechanism. When user requests data from cloud server, the user sends a password to the server for identification. Taking it into consideration that the password may be intercepted, the structure uses asymmetric response mode. Hou, Wu, Zhen and Yang point out that the secure core of cloud storage is security and privacy in distributed system. So they propose a secure virtual protection scheme based on SSL and Daoli in paper [14], [15]. By transferring data over SSL and deploying Daoli on the cloud server, the system encrypts data before it is written into the hard disk. In paper [16], Feng points out that in paper [14], the burden of server will increase and data may leak during transmission in cloud servers. Feng proposes a more concise scheme: encrypting data in closed cloud environment. Besides, it can achieve multi-point secure storage with one time encrypting. However, these encryption make search in cloud more difficult. Currently, searchable encryption is a hot topic in the field of cloud computing. Paper [17]–[20] give different solutions to this problem. Each of them achieves high accuracy, security and efficient.

In paper [21], Seny and Kristin concern that the service provider is not complete trusted, so they design a virtual private storage service based on recent developed cryptographic techniques. Such a service achieves the best of both worlds by providing the security of a private cloud and the functionality and cost saving of a public cloud. In paper [22], Wang et al. point out that users no longer have physical possession of the outsourced data and it makes the data integrity protection in cloud computing a formidable task. Thus, enabling public audit ability for cloud storage is of critical importance so that user can resort to a third-party auditor (TPA) to check the integrity of outsourced data. They propose a secure cloud storage system supporting privacy-preserving public auditing and further extend our result to enable the TPA to perform audits for multiple users simultaneously and efficiently. Shen et al. propose an efficient public auditing protocol with global and sampling

blockless verification as well as batch auditing, where data dynamics are substantially more efficiently supported than is the case with the state of the art [23]. In paper [24], Wei *et al.* point out that most of the previous works on the cloud security focus on the storage security rather than taking the computation security into consideration together. Thus they propose a privacy cheating discouragement and secure computation auditing protocol, also named SecCloud which is a first protocol bridging secure storage and secure computation auditing in cloud and achieves privacy cheating discouragement by designated verifier signature, batch verification and probabilistic sampling techniques. In paper [25], Atan R et al. propose a secure framework, consisting of two main layers: agent layer and cloud data storage layer. The architecture includes five types of agents: User Interface Agent, User Agent, DER Agent, Data Retrieval Agent and Data Distribution Preparation Agent.

The researches above are all improvements of privacy protection in cloud storage in different aspects. Some of them use variety encryption policies in different positions. Others solve the privacy problem with the help of auditing or building their own secure framework. However, there is a common defect in these researches. Once the CSP is untrusted, all of these schemes are invalid. They cannot resist internal attacks or prevent the CSP from selling user's data to earn illegal profit. The private data will be decoded once malicious attackers get it no matter how advanced the encryption technologies are because user's data was integrally stored in cloud server. Therefore, we propose a new secure cloud storage scheme in this paper. By dividing file with specific code and combining with TLS framework based on fog computing model, we can achieve high degree privacy protection of data. It does not means that we abandon the encryption technology. In our scheme encryption also help us to protect fine-grained secure of the data.

## III. Secure Cloud Storage Based on Fog Computing

The security degree is an important metric to measure the quality of cloud storage system. Furthermore, data security is the most important part in cloud storage security and it includes three aspects: data privacy, data integrity and data availability. Ensuring data privacy and integrity has always been the focus of relevant researches [26]. On another hand, data privacy is also the most concerned part of the users. From a business perspective, company with high security degree will attract more users. Therefore improving security is an crucial goal no matter in academia or business. In this section, we will detailedly elaborate how the TLS framework protects the data privacy, the implementation details of work flow and the theoretical safety and efficiency analysis of the storage scheme.

### A. Fog Computing

Our scheme is based on fog computing model, which is an extension of cloud computing. Fog computing was firstly proposed by Ciscos Bonomi in 2011 [27]. In Bonomi's view, fog computing is similar to the cloud computing, the name of fog computing is very vivid. Compared to highly concentrated cloud computing, fog computing is closer to edge network and has many advantages as follows: broader geographical distributions, higher real-time and low latency. In considering of these characters, fog computing is more suitable to the applications which are sensitive to delay. On another hand, compared to sensor nodes, fog computing nodes have a certain storage capacity and data processing capability, which can do some simple data processing, especially those applications based on geographical location. Thus we can deploy CI on the fog server to do some calculating works.

Fog computing is usually a three-level architecture, the upmost is cloud computing layer which has powerful storage capacity and compute capability. The next level is fog computing layer. The fog computing layer serves as the middle layer of the fog computing model and plays a crucial role in transmission between cloud computing layer and sensor network layer. The fog nodes in fog computing layer has a certain storage capacity and compute capability. The bottom is wireless sensor network layer [28]. The main work of this layer is collecting data and uploading it to the fog server. Besides, the transfer rate between fog computing layer and other layers is faster than the rate directly between cloud layer and the bottom layer [29]–[31]. The introduction of fog computing can relief the cloud computing layer, improving the work efficiency. In our scheme, we take advantage of the fog computing model, adopt three-layer structure. Furthermore, we replace the WSNs layer by user's local machine.

### B. Three-Layer Privacy Preserving Cloud Storage Scheme Based on Fog Computing Model

In order to protect user's privacy, we propose a TLS framework based on fog computing model. The TSL framework can give user a certain power of management and effectively protect user's privacy. As mentioned, the interior attack is difficult to resist. Traditional approaches work well in solving outside attack, but when CSP itself has problems, traditional ways are all invalid. Different from the traditional approaches, in our scheme, user's data is divided into three different-size parts with encoding technology. Each of them will lack a part of key information for confidentiality. Combining with the fog computing model, the three parts of data will be stored in the cloud server, the fog server and user's local machine according to the order from large to small. By this method, the attacker cannot recover the user's original data even if he gets all the data from a certain server. As for the CSP, they also cannot get any useful information without the data stored in the fog server and local machine because both of the fog server and local machine are controlled by users.

As shown in Fig. 2, the TLS framework makes full use of fog server's storage and data processing capability. The architecture includes three layers, the cloud server, the fog server and the local machine. Each server saves a certain part of data, the storage proportion is determined by users' allocation strategy. Firstly, user's data will be encoded on user's local machine. Then, for example, let 1% encoded data be stored in the machine. Then upload the remainder 99% data to the fog server. Secondly, on the fog server, we do similar operations to the data which

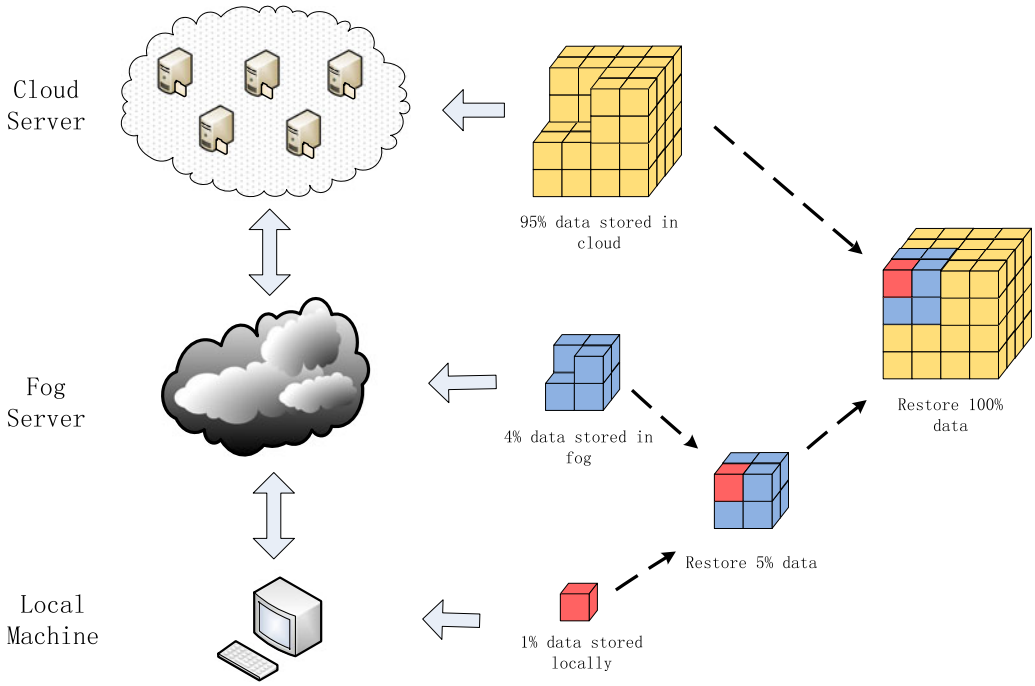


Fig. 2. Illustration of Three-Layer storage framework based on fog computing.

comes from user's machine. There will be about 4% data stored in the fog server and then upload the remainder data to the cloud server. The above operations are based on Hash-Solomon code. Hash-Solomon code is a kind of coding methods based on Reed-Solomon code. After being encoded by Hash-Solomon code, the data will be divided into *k* parts and generates *m* redundant data. Hash-Solomon code has such property, in these *k*+*m* parts of data, if someone has at least *k* parts, he can recover the complete data. In other word, nobody can recover the complete data with less than *k* parts of data. According to this property of Hash-Solomon code, in our scheme, we let no more than *k*-1 parts of data be stored in higher server which has larger storage capacity and let the remainder be stored in the lower server. In this way, the stealer cannot recover the complete data even if one of the three layers' data was stolen. Thus we can ensure the privacy of user's data. Then we consider the value of *k* and *m*. Assuming that we want to save *r*% data on the fog server. In the Hash-Solomon code, we have definitions as follows:

*Definition 1 Invalid Ratio:* the ratio of the number of failure data blocks to the number of data blocks which will be used in encoding. In other words, the ratio of the number of data blocks stored in lower server to the number of data blocks stored in the upper server. For example, the ratio of the number of data blocks stored in the local machine to the number of data blocks stored in the fog server. In the same way, the ratio of the number of data blocks stored in the fog server to the number of data blocks stored in the cloud server.

*Definition 2 Maximal Invalid Ratio:* the maximal invalid ratio is the ratio of the number of invalid data to the number of all data blocks when the upper server can just recover the complete data by the data blocks stored in them. If there was one more invalid data blocks, the upper server can't recover the complete data anymore.

In Hash-Solomon code, the *Maximal Invalid Ratio* can be expressed as $\frac{m}{k+m}$. For convenience, we just consider two layers situation. Assuming that there is $x$ MB data which is prepared to save. After encoding, there will be $\frac{k+m}{m} * x$ data. We prepare to save *r*% in the lower server.

In order to avoid the upper server recovers the data, the value of *k*, *m* and *r* must satisfy the relationship:

$$\frac{m}{k+m} \leq \frac{k+m}{k} * r \tag{1}$$

Through functional transformation, the relationship between *k m* and *r* can be expressed as formula (2). We can see that if the parameter *r* is determined, the parameter *k* can be expressed by *m*. So we can only consider the ratio and the number of data blocks when we use our scheme.

$$k = \frac{(m-2mr) + \sqrt{(2mr-m)^2 - 4m^2r^2}}{2r} \tag{2}$$

The parameter *k* is the number of blocks after data being divided, the parameter *m* is the number of redundant data blocks and the parameter *r* is the storage ratio of different servers. Besides, the fog server includes Computational Intelligence which can help the system with calculating the results of the values of *k* and *m*, because of the nodes in the fog server having its own computing power.

### C. Implementation Detail of Workflow

*1) Stored Procedure:* When user wants to store his file to the cloud server, the procedure is shown as Fig. 3. First of all, user's

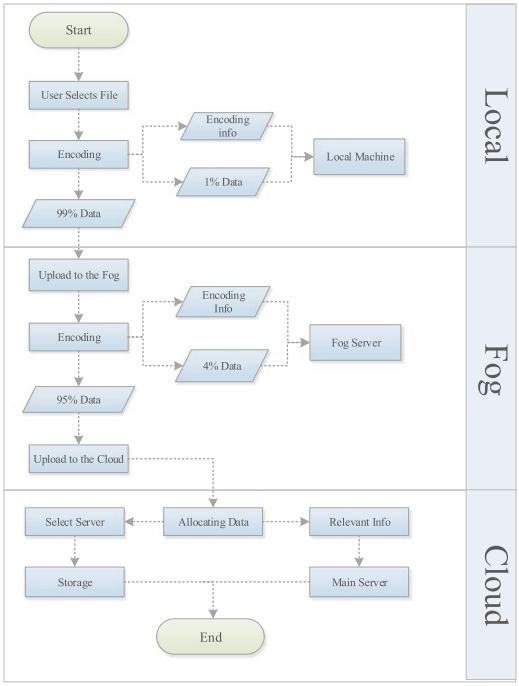


Fig. 3. Diagram of stored procedure.

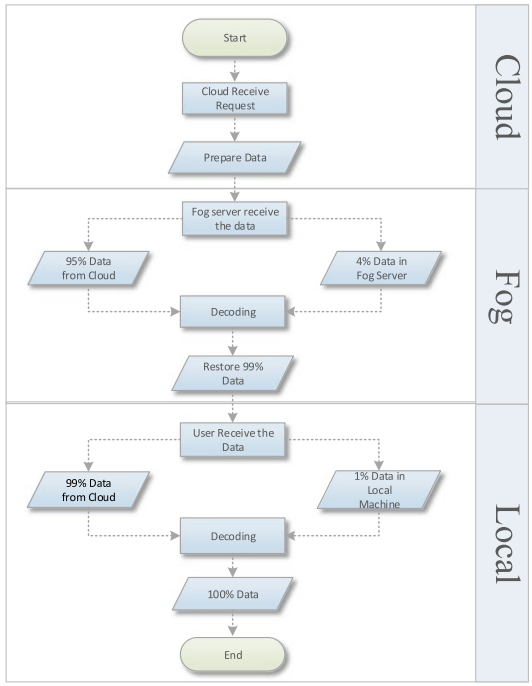


Fig. 4. Diagram of download procedure.

file will be encoded with Hash-Solomon code. And then, the file will be divided into several data blocks and the system will also feedback encoding information simultaneously. Assuming that 1% data blocks and the encoding information will be stored locally. The remainder 99% data blocks will be uploaded to the fog server. Secondly, after receiving the 99% data blocks from user's machine, these data blocks will be encoded with Hash-Solomon again. These data blocks will be divided into smaller data blocks and generates new encoding information. Similarly, assuming that 4% data blocks and encoding information will be stored in the fog server. The remainder 95% data blocks will be uploaded to the cloud server. Thirdly, after cloud server received the data blocks form fog side, these data blocks will be distributed by cloud manage system [32]. Finally, the storage procedure ends when all the related information be recorded in different servers.

*2) Download Procedure:* When user wants to download his file from the cloud server, the procedure is shown in Fig. 4. Firstly, cloud server receives user's request and then integrates the data in different distributed servers. After integration, cloud server sends the 95% data to the fog server. Secondly, the fog server receives the data from the cloud server. Combining with the 4% data blocks of fog server and the encoding information, we can recover 99% data. Then the fog server returns the 99% data to the user. Thirdly, the user receives the data from fog server. User can get the complete data by repeating the above steps.

### D. Theoretical Safety Analysis

This section will provide theoretical safety analysis of the structure proposed in our research and prove that the secure storage structure can really improve the capability of privacy protection.

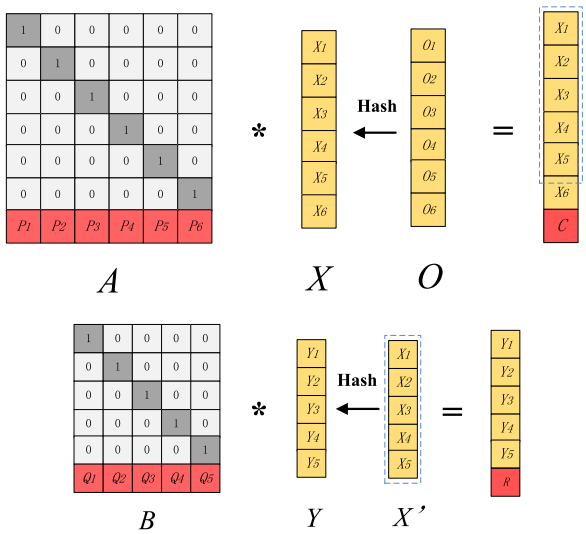


Fig. 5. Diagram of download procedure.

Based on the Reed-Solomon code algorithm, we propose a Hash-Solomon code algorithm. The Hash-Solomon encoding process is actually a matrix operation. As shown in Fig. 5, firstly we should do mapping transformation on the file which is prepared to be stored, so that each word of the file corresponds to a number in $GF(2^{\omega})$ [33]. After mapping transformation we get file matrix $O$. Secondly we do hash transform on matrix $O$ and get matrix $X$. Then we multiply the transformed matrix $X$ by the encoding matrix $A$. The multiplication will generate $k$ data blocks $X_1$ to $X_6$ and $m$ redundant data blocks $C$ ($k = 6$, $m$

TABLE I
CRACKING DIFFICULTY DEGREE

| Galois Field | m | k | Times of exhaustion |
|---|---|---|---|
| $GF(2^4)$ | 1 | 6 | $256^3$ |
| $GF(2^4)$ | 2 | 6 | $256^6$ |
| $GF(2^8)$ | 1 | 6 | $256^6$ |
| $GF(2^8)$ | 2 | 6 | $256^{12}$ |
| $GF(2^{16})$ | 1 | 6 | $256^{12}$ |
| $GF(2^{16})$ | 2 | 6 | $256^{14}$ |

$= 1$). In Fig. 5, we prepare to save $X_1$ to $X_5$ in the Cloud and Fog, and store $X_6$ and $C$ in the local machine. The next step is similar to the above operations, we do hash transform on $X'$ and get file matrix $Y$. Then we multiply the transformed matrix $Y$ by the encoding matrix $B$. At last, we store $Y_1$ to $Y_4$ in the cloud server and store $Y_5$ and $R$ in the fog server ($k = 5, m = 1$). The encoding matrix usually consists of an identity matrix and a Vandermonde matrix or a Cauchy matrix.

It is worth noting that Hash-Solomon code has the following properties: in the $k$+$m$ data blocks, if we have at least $k$ data blocks, we can recover the original data combining with the encoding matrix. But once the number of data blocks is less than $k$, it cannot be recovered. Using the above properties in our scheme, after each encoding, we store less than $k$ parts of data blocks in the higher server and store the remainder parts of data blocks in the lower server. With such reasonable allocation, the cloud server, the fog server and user's local machine independently store a certain percentage of data blocks. It is impossible to recover the original data with any single server's data. The TLS framework largely solves the leakage of user's privacy. Further considering a worse case, if the attacker is brilliant enough, he steals data blocks from two servers so that he owns more than $k$ parts of data blocks. Is the attacker able to recover the user's original data? Here is the encoding problem. Assuming that the attacker steals enough data, but if he doesn't have the information contained in the encoding matrix, he can hardly recover user's original data from the scattered data blocks. If he wants to crack the encoding matrix, the degree of difficulty is shown in the Table I.

As can be seen from the Table I, attacker can hardly crack the encoding matrix. In the real scenario, the values of $m$ and $k$ are usually very large, so it is impossible to crack the encoding matrix in theory. But using encoding technology cannot ensure the privacy of each data block especially for document file. For example, after a document is encoded, each part of data blocks still contains the information of the document. For some high privacy demanding documents, it is obviously not available. So we add a hash transform before encoding to disrupt the sequence of original data and save the relevant hash information in the local server. As shown in Fig. 6, the original code divides a sentence into different fragments according to original sequence. However, the hash code divides the sentence into different fragments according to random sequence. Thereby Hash-Solomon code improves the privacy protection and prevents the attacker from getting fragmentary information.

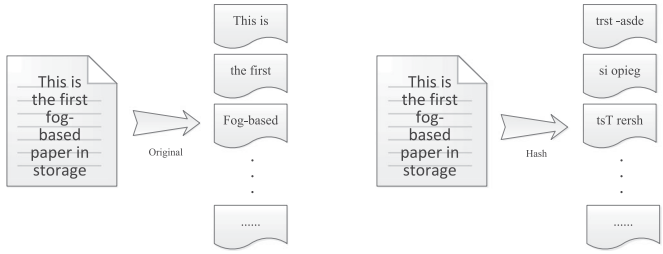


Fig. 6. Original transform vs. Hash Transform.

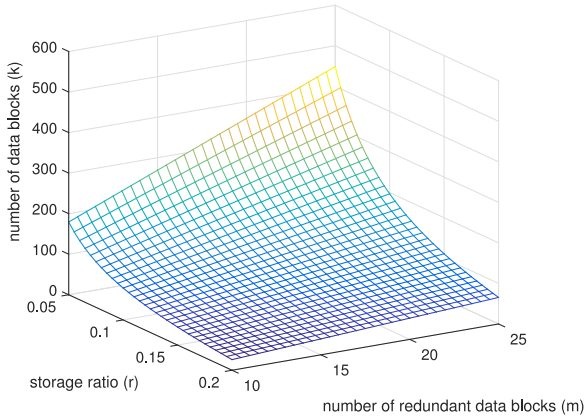


Fig. 7. Diagram of relationship of the number of data blocks (k), redundant data blocks (m) and storage ratio (r).

### E. Efficiency Analysis

In Section III-B, we have discussed the relationship of $k$ and $m$. As shown in the Fig. 7, we find that the ratio of $k$ and $m$ is decided once the storage ratio is decided. It means that if we set the storage ratio as 20%, $k = 3m$. Then we set $k = 3$, $m = 1$. In the real scenario, data blocks cannot be stored partly. In the above example, the lower server must store at least 2 blocks, so that the real storage ratio is 50%, which is far from the 20%. In order to reduce error, we can let $k$ or $m$ be a large number. However, with the increasing of $k$, the encoding and decoding efficiency will decrease, which will be proved by experiments in the next section. In this section, we will discuss how to balance the storage efficiency and the coding efficiency. At last, we propose a comprehensive index of the whole efficiency of the scheme.

The storage efficiency is an important index for a storage-related algorithm. A good system with high storage efficiency can save storage capacity as much as possible. Storage Industry Networking Association defines the storage efficiency as:

$$StorageEfficiency = \frac{DataSpace}{DataSpace + CheckSpace} \quad (3)$$

In our scheme, storage efficiency can be expressed as $E_s = \frac{k}{k+m}$. Then we can get the following formulas (4, 5).We can see that the storage efficiency will increase with the increment to the ratio of $k$ and $m$. From Fig. 7 we know that when the ratio of $k$ and $m$ increase, the number of data blocks ($k$) also increase,

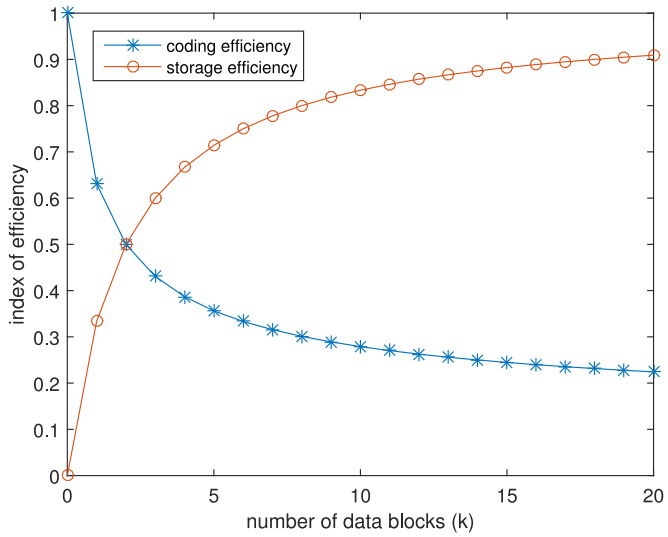


Fig. 8. Diagram of the influence of the number of data blocks (k) to the efficiency of storage and coding.

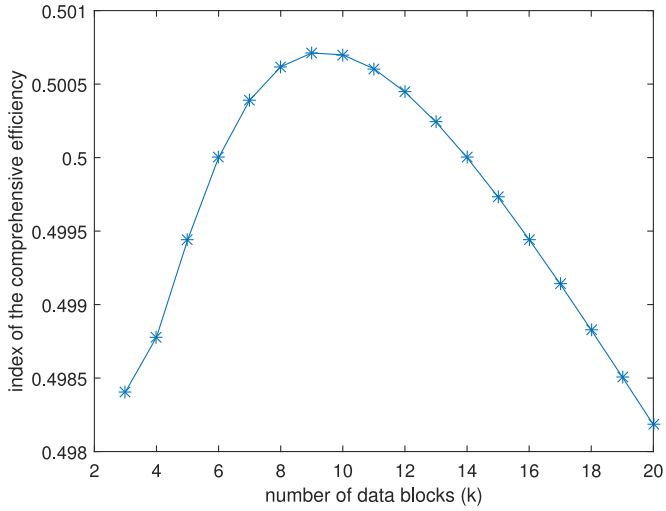


Fig. 9. Diagram of the influence of the number of data blocks (k) to the efficiency of storage and coding.

which influences the coding efficiency.

$$E_s = \frac{k}{k+m} = \frac{\frac{k}{m}}{\frac{k}{m}+1} \tag{4}$$

$$\lim_{\frac{k}{m}\to\infty} = \frac{\frac{k}{m}}{\frac{k}{m}+1} = 1 \tag{5}$$

The coding efficiency is related to the operation on Galois field. We consider the influence of different bits of coding which is related to the $\omega$ of the Galois field. The relationship of $\omega$, $k$ and $m$ satisfy the equation $2^\omega > k+m$. When $\omega$ increases, the consume of RAM increases. Therefore, we let the reciprocal of $\omega$ to present the coding efficiency and it can be expressed as

$$E_c = \frac{\ln(k+m)}{\ln 2} \tag{6}$$

Fig. 8 shows the change of storage efficiency and the coding efficiency when the number of $k$ increases. The value of $m$ is set to 2. Apparently, the tendency of storage efficiency is contrary to the tendency of coding. It means there must be a value of $k$ which can achieve a best efficiency of the whole system.

Therefore we should design a new index to take both of the storage efficiency and coding efficiency into consideration. The comprehensive efficiency of the scheme can be expressed as

$$E_w = C_1 \frac{\ln(k+m)}{\ln 2} + C_2 \frac{k}{k+m} \tag{7}$$

The parameter $C_1$ and $C_2$ are related to the storage ratio. For example, we set the value of $m$ to 2, then the value of $C_1$ is set as 0.6, the value of $C_2$ is set as 0.4. As shown in Fig. 9, the comprehensive efficiency of the scheme increases at first and decreases after it achieve the summit of the functional graph. We can consider the value of $k$ which corresponds to the summit is the most suitable value for the whole efficiency of the scheme.

TABLE II
EXPERIMENTS ENVIRONMENT

| Items | Parameter value |
|---|---|
| Operating system | Linux |
| Programming language | C |
| CPU | Intel Core i7 2.50 GHz |
| Memory | 8 GB |
| Hard Disk | 1TB |

## IV. EXPERIMENT AND ANALYSIS

In this section, we evaluate the performance and feasibility of the TLS framework based on fog computing model through a series of tests, including encoding, decoding and test of different sizes of data.

### A. Experimental Environment

All of the experiments in this paper were conducted by simulation and the environmental parameters are shown as Table II. There are three types files which are listed as flows: picture (.NEF, 24.3 MB), audio (.MP3, 84.2 MB) and video (.RMVB, 615 MB).

All the experiments in this paper use 'one more block' principle which means the lower server only saving $m+1$ data blocks. In this way, the scheme can ensure the privacy of data and reduce the storage pressure of the lower servers at the same time.

### B. Experiment Results

Fig. 10 shows the relationship between data storage in user's machine and the number of blocks while using different kinds of data. The parameter $m$ represents the number of redundant data blocks while the parameter $k$ represents the number of data blocks which we want the original data be divided into. Note that the value of $m$ is set as 2 in this part. As we can see, when

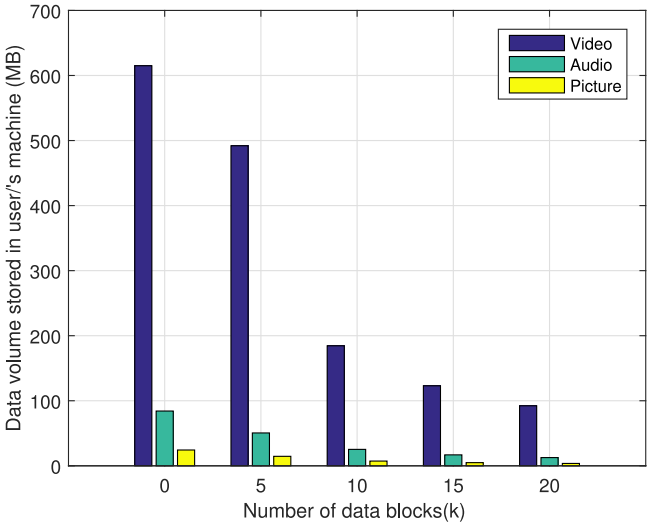


Fig. 10. The local storage volume of different files.

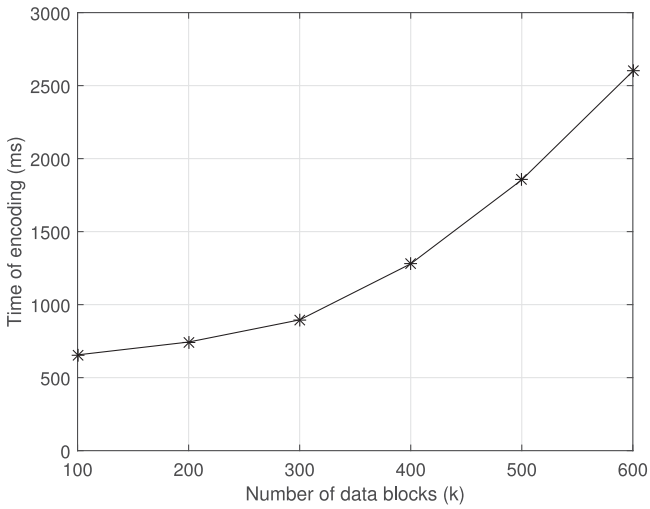


Fig. 11. Relationship between time of encoding and the number of k.

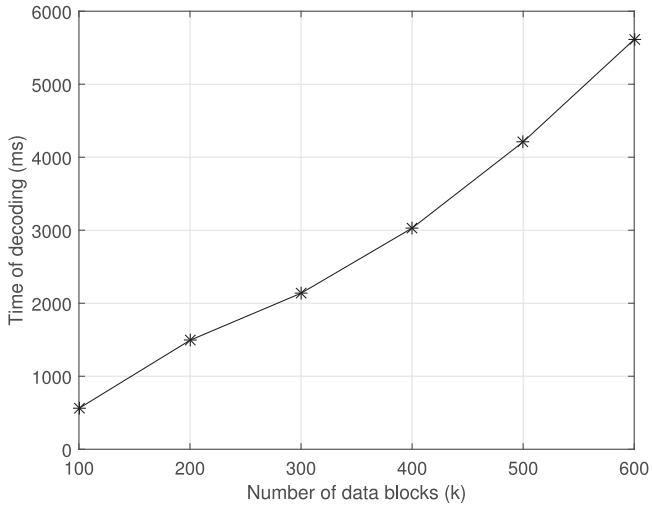


Fig. 12. Relationship between time of decoding and the number of k.

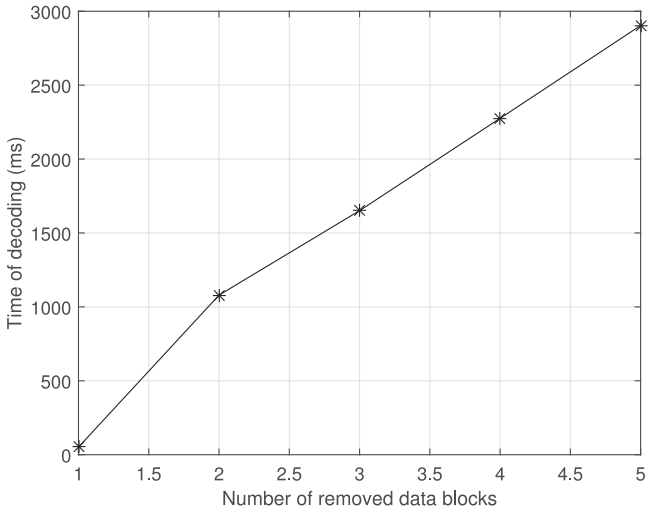


Fig. 13. Relationship between time of decoding and the number of removed data.

the number of data blocks $k$ increases, the data volume stored in user's local machine decreases. It means that the more the number of data blocks is, the smaller the local storage pressure is. On another hand, our method performs differently when using different volume of data. The larger the volume of the data is, the better effect our method performs in the experiment. Therefore, in the real scenario, it is of vital importance to increase the value of $k$ to alleviate user's storage pressure. As for small files, merging files before uploading is necessary.

Fig. 11 shows the tendency of encoding time with different number of data blocks. The value of $m$ is also set as 2. When the number of data blocks $k$ increases, the encoding time grows exponentially. Accordingly, in the real scenario, we should consider delay degree that user can endure and adjust the value of $k$ according to the user's machine performance dynamically.

The relationship between decoding time and number of data blocks is shown in Fig. 12. Both the value of $m$ and the value of removed data is set as 2. When the number of data blocks $k$ increases from 100 to 600, the decoding time increases at express speed. As we can see, the decoding process costs more time than the encoding process does, so we should pay more attention to enhance decoding efficiency in real scenario.

In the Fig. 13, we present the tendency of decoding time with different number of removed data from 1 to 5. The value of $k$ is set as 100 and the value of $m$ is set as 5. In the real scenario, the ratio of $m$ and $k$ should be very small to relieve the user's storage pressure. What's more, the number of removed data should be smaller than $m$, otherwise, system will be error-reporting. On another hand, the decoding time increases with the increment of the number of removed data, which means that we should download all of the data from the upper server as much as possible to maximize the decoding efficiency.

The Hash-Solomon code is the key to the whole efficiency of our scheme. Therefore, find a better coding matrix is of vital

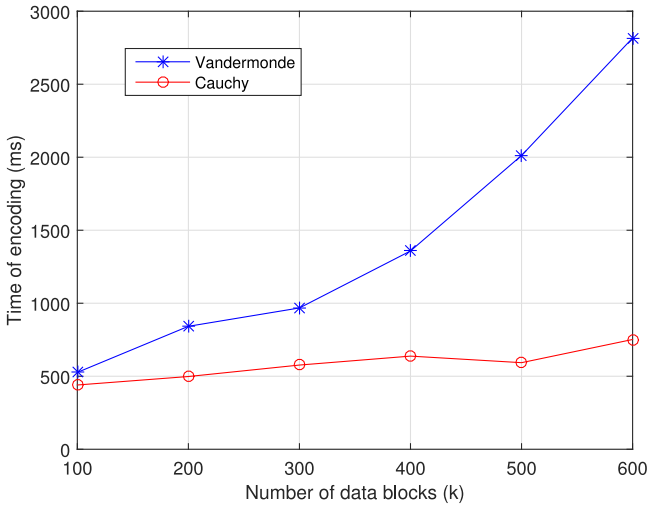


Fig. 14. Cauchy matrix vs. Vandermonde matrix.

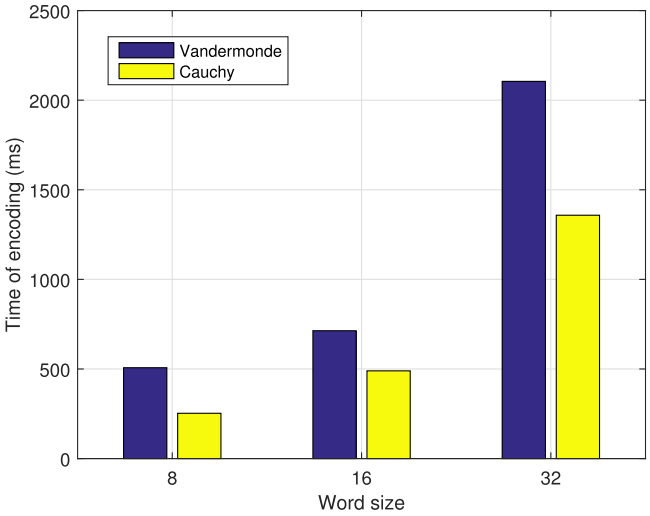


Fig. 15. Relationship between time of encoding and word size of Galois field.

importance. The code matrix can be chosen from Vandermonde matrix and Cauchy matrix. Different from Vandermonde matrix, Cauchy matrix uses AND operation and XOR logical operation. In Cauchy's way, coding efficiency improves. Besides, the complexity decrease from $O(n^3)$ to $O(n^2)$. As shown in the Fig. 14, we present the two tendencies of encoding time with different number of data blocks $k$ from 100 to 600. The value of $m$ is set as 2. We can see that the encoding time raises with the increase of the number of data blocks $k$, no matter Vandermonde or Cauchy. On another hand, the Cauchy matrix has better performance than Vandermonde matrix. The time cost of Cauchy always less than the Vandermonde. When the number of $k$ is very large, the cost of Vandermonde raises sharply while the cost of Cauchy increases slightly.

In the Section III, the coding efficiency is related to the $\omega$ in Galois field $GF(2^{\omega})$. As shown in Fig. 15, we present the encoding time with different values of $\omega$. Besides, we also consider the comparison of Vandermonde and Cauchy. As shown in the Fig. 15, we set the value of $\omega$ as 8, 16 and 32. As we can see, no matter Vandermonde or Cauchy, the cost of encoding time increases with the increase of $\omega$.

## V. Conclusion

The development of cloud computing brings us a lot of benefits. Cloud storage is a convenient technology which helps users to expand their storage capacity. However, cloud storage also causes a series of secure problems. When using cloud storage, users do not actually control the physical storage of their data and it results in the separation of ownership and management of data. In order to solve the problem of privacy protection in cloud storage, we propose a TLS framework based on fog computing model and design a Hash-Solomon algorithm. Through the theoretical safety analysis, the scheme is proved to be feasible. By allocating the ratio of data blocks stored in different servers reasonably, we can ensure the privacy of data in each server. On another hand, cracking the encoding matrix is impossible theoretically. Besides, using hash transformation can protect the fragmentary information. Through the experiment test, this scheme can efficiently complete encoding and decoding without influence of the cloud storage efficiency. Furthermore, we design a reasonable comprehensive efficiency index, in order to achieve the maximum efficiency, and we also find that the Cauchy matrix is more efficient in coding process.

## References

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," *Nat. Inst. Stand. Technol.*, vol. 53, no. 6, pp. 50–50, 2009.
[2] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless Commun. Mobile Comput.*, vol. 13, no. 18, pp. 1587–1611, 2013.
[3] J. Chase, R. Kaewpuang, W. Yonggang, and D. Niyato, "Joint virtual machine and bandwidth allocation in software defined network (sdn) and cloud computing environments," in *Proc. IEEE Int. Conf. Commun.*, 2014, pp. 2969–2974.
[4] H. Li, W. Sun, F. Li, and B. Wang, "Secure and privacy-preserving data storage service in public cloud," *J. Comput. Res. Develop.*, vol. 51, no. 7, pp. 1397–1409, 2014.
[5] Y. Li, T. Wang, G. Wang, J. Liang, and H. Chen, "Efficient data collection in sensor-cloud system with multiple mobile sinks," in *Proc. Adv. Serv. Comput., 10th Asia-Pac. Serv. Comput. Conf.*, 2016, pp. 130–143.
[6] L. Xiao, Q. Li, and J. Liu, "Survey on secure cloud storage," *J. Data Acquis. Process.*, vol. 31, no. 3, pp. 464–472, 2016.
[7] R. J. McEliece and D. V. Sarwate, "On sharing secrets and reed-solomon codes," *Commun. ACM*, vol. 24, no. 9, pp. 583–584, 1981.
[8] J. S. Plank, "T1: Erasure codes for storage applications," in *Proc. 4th USENIX Conf. File Storage Technol.*, 2005, pp. 1–74.
[9] R. Kulkarni, A. Forster, and G. Venayagamoorthy, "Computational intelligence in wireless sensor networks: A survey," *IEEE Commun. Surv. Tuts.*, vol. 13, no. 1, pp. 68–96, First Quarter 2011.
[10] Z. Xia, X. Wang, L. Zhang, Z. Qin, X. Sun, and K. Ren, "A privacy-preserving and copy-deterrence content-based image retrieval scheme in cloud computing," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 11, pp. 2594–2608, Nov. 2016.
[11] J. Shen, D. Liu, J. Shen, Q. Liu, and X. Sun, "A secure cloud-assisted urban data sharing framework for ubiquitous-cities," *Pervasive Mobile Comput.*, vol. 41, pp. 219–230, 2017.
[12] Z. Fu, F. Huang, K. Ren, J. Weng, and C. Wang, "Privacy-preserving smart semantic search based on conceptual graphs over encrypted outsourced data," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 8, pp. 1874–1884, Aug. 2017.
[13] J. Hou, C. Piao, and T. Fan, "Privacy preservation cloud storage architecture research," *J. Hebei Acad. Sci.*, vol. 30, no. 2, pp. 45–48, 2013.

[14] Q. Hou, Y. Wu, W. Zheng, and G. Yang, "A method on protection of user data privacy in cloud storage platform," *J. Comput. Res. Develop.*, vol. 48, no. 7, pp. 1146–1154, 2011.
[15] P. Barham *et al.*, "Xen and the art of virtualization," *ACM SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 164–177, 2003.
[16] G. Feng, "A data privacy protection scheme of cloud storage," vol. 14, no. 12, pp. 174–176, 2015.
[17] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 12, pp. 2706–2716, Dec. 2016.
[18] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 9, pp. 2546–2559, Sep. 2016.
[19] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 340–352, Feb. 2016.
[20] Z. Fu, F. Huang, X. Sun, A. Vasilakos, and C.-N. Yang, "Enabling semantic search based on conceptual graphs over encrypted outsourced data," *IEEE Trans. Serv. Comput.*. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TSC.2016.2622697
[21] G. Kulkarni, R. Waghmare, R. Palwe, V. Waykule, H. Bankar, and K. Koli, "Cloud storage architecture," in *Proc. 7th Int. Conf. Telecommun. Syst., Serv., Appl.*, 2012, pp. 76–81.
[22] C. Wang, S. S. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 362–375, Feb. 2013.
[23] J. Shen, J. Shen, X. Chen, X. Huang, and W. Susilo, "An efficient public auditing protocol with novel dynamic structure for cloud data," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 10, pp. 2402–2415, Oct. 2017.
[24] L. Wei *et al.*, "Security and privacy for storage and computation in cloud computing," *Inf. Sci.*, vol. 258, pp. 371–386, 2014.
[25] R. Atan, A. M. Talib, and M. A. A. Murad, "Formulating a security layer of cloud data storage framework based on multi agent system architecture," *GSTF J. Comput.*, vol. 1, no. 1, pp. 121–124, 2014.
[26] M. Z. A. Bhuiyan, T. Wang, T. Hayajneh, and G. M. Weiss, "Maintaining the balance between privacy and data integrity in internet of things," in *Proc. Int. Conf. Manage. Eng., Softw. Eng. Serv. Sci.*, 2017, pp. 177–182.
[27] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proc. 1st Edition MCC Workshop Mobile Cloud Comput*, 2012, pp. 13–16.
[28] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Comput. Netw.*, vol. 52, no. 12, pp. 2292–2330, 2008.
[29] T. Wang *et al.*, "Maximizing real-time streaming services based on a multi-servers networking framework," *Comput. Netw.*, vol. 93, pp. 199–212, 2015.
[30] T. Wang *et al.*, "Reliable wireless connections for fast-moving rail users based on a chained fog structure," *Inf. Sci.*, vol. 379, pp. 160–176, 2017.
[31] J. Zeng, T. Wang, Y. Lai, J. Liang, and H. Chen, "Data delivery from WSNs to cloud based on a fog structure," in *Proc. Int. Conf. Adv. Cloud Big Data*, 2016, pp. 104–109.
[32] M. D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, and A. Vakali, "Cloud computing: Distributed internet computing for it and scientific research," *IEEE Internet Comput.*, vol. 13, no. 5, pp. 10–13, Sep./Oct. 2009.
[33] R. Steinberg, "A geometric approach to the representations of the full linear group over a galois field," *Trans. Amer. Math. Soc.*, vol. 71, no. 2, pp. 274–282, 1951.

**Tian Wang** received the B.Sc. and M.Sc. degrees in computer science from Central South University, Changsha, China, in 2004 and 2007, respectively, and the Ph.D. degree from City University of Hong Kong, Hong Kong, in 2011. He is currently a Professor with the National Huaqiao University of China, Quanzhou, China. His research interests include wireless sensor networks, fog computing, and mobile computing.

**Jiyuan Zhou** received the B.S. degree from Tianjin Polytechnic University, Tianjin, China, in 2016. He is currently working toward the Master's degree from Huaqiao University, Quanzhou, China. His research interests include security in wireless networks, fog computing, and security in cloud storage

**Xinlei Chen** received the Bachelor's degree from Huaqiao University, Xiamen, China, in 2017. His current research interests include cloud computing and cloud storage.

**Guojun Wang** received the B.Sc. degree in geophysics in 1992, the M.Sc. degree in computer science in 1996, and the Ph.D. degree in computer science in 2002, all from Central South University, Changsha, China. He is currently the Pearl River Scholarship Distinguished Professor with Guangzhou University, Guangzhou, China. He was a Professor with Central South University, Changsha, China; a Visiting Scholar at Temple University and Florida Atlantic University, USA; a Visiting Researcher at the University of Aizu, Japan, and a Research Fellow at Hong Kong Polytechnic University. His research interests include cloud computing, trusted computing, and information security. He is a distinguished member of the CCF, and a member of the ACM and IEICE.

**Anfeng Liu** received the M.Sc. and Ph.D degrees from Central South University, Changsha, China, 2002 and 2005 respectively, both majored in computer science. He is a Professor with the School of Information Science and Engineering, Central South University. His major research interests are cyber-physical systems, service network, wireless sensor network. He is a Member (E200012141M) of China Computer Federation.

**Yang Liu** (M'14) received the B.E. degree in electrical engineering and its automation and the M.E. degree in control theory and control engineering from Harbin Engineering University, Harbin, China, in 2008 and 2010, respectively, and the Ph.D. degree in computer engineering from the Center for Advanced Computer Studies, University of Louisiana at Lafayette, Lafayette, LA, USA, in 2014. He is currently an Assistant Professor with Beijing University of Posts and Telecommunications. His current research interests include wireless networking and mobile computing. He is a member of the ACM.