Cluster Computing (2023) 26:3839–3854
https://doi.org/10.1007/s10586-022-03785-y

Enabling Secure Data sharing with data deduplication and sensitive information hiding in cloud-assisted Electronic Medical Systems

Zhiqiang Wang[1] · Wenjing Gao[1] · Ming Yang[2] · Rong Hao[1]

Received: 23 April 2022 / Revised: 5 October 2022 / Accepted: 12 October 2022 / Published online: 28 October 2022

Abstract
Data sharing is very important for medical researchers to do research on certain diseases in cloud-assisted electronic medical systems. Nonetheless, there are large amounts of duplicate data in shared electronic medical records, which incurs redundant storage. In addition, data sharing of electronic medical records might expose the sensitive information of patients. In order to address above problems, we propose a secure data sharing scheme with data deduplication and sensitive information hiding in cloud-assisted electronic medical systems in this paper. In order to protect the sensitive information privacy and enhance the deduplication efficiency, we replace the patient's sensitive information of electronic medical records by wildcards before encrypting the whole electronic medical records. The authorized researcher can decrypt and obtain the electronic medical records under the condition that the sensitive information of shared electronic medical records is hidden. Moreover, we clarify the diagnose information of the electronic medical records into different types according to the duplicate ratio. The authorized researchers can selectively download data according to the duplicate ratio of diagnostic information. Our proposed scheme can resist brute-force attacks and single-point-of-failure attack. The experimental results show our proposed scheme is more efficient than the existing schemes.

Keywords Data sharing · Secure deduplication · Sensitive information hiding · COVID-19 · Electronic medical record · Data integrity

1 Introduction

In the era of data explosion, individuals and enterprises need to store huge amounts of data. For example, IDC thinks that the global datasphere will reach 175 zettabytes by 2025 [1]. Faced with heavy storage burden, users would like to upload the data to the cloud server to release them from storing and managing such large-scale data. Nonetheless, it incurs a lot of duplicate data stored in cloud server because a lot of identical data may be uploaded to the cloud server by different users. The study shows that about 75% of data is duplicate in standard application system [2] and the proportion of identical data even reaches 90% in backup and archival storage system [3]. In order to reduce redundant data in cloud server, deduplication technology comes into being. Deduplication technology means that the cloud server only stores one duplicate for several identical data. This technology attracts wide attention since it can save a lot of financial cost of the cloud server and the user.

Data deduplication has a wide range of applications in various fields, such as cloud-assisted electronic health systems. Compared to traditional medical record management systems, cloud-assisted electronic health systems are more efficient, accurate and reliable in managing electronic medical records [4, 5]. In addition, cloud-assisted electronic health systems also play an important role in resolving judgements and disputes in medical malpractice [6, 7]. As we all know, the type of the diagnostic information such as symptoms and drugs in electronic medical records is limited. For example, there are only about 100

✉ Rong Hao
hr@qdu.edu.cn

[1] College of Computer Science and Technology, Qingdao University, Qingdao 266071, China

[2] Shandong Provincial Key Laboratory of Computer Networks, Shandong Computer Science Center, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China

kinds of antibiotics in existence [8]. Accordingly, there exist a lot of the same data in electronic medical records. The study shows that data deduplication saves about 65% storage space in electronic health systems [9]. Data sharing of electronic medical records is very important for medical researchers to do research on certain diseases. For example, the COVID-19 electronic medical record contains some patient's common symptoms like fever and dizziness. Sharing such electronic medical records is very helpful for researchers to study how to find the potential COVID-19 patients. Nevertheless, data sharing of electronic medical records might bring privacy-exposure problems. Generally, the electronic medical record is composed of two parts. The first part contains the sensitive information, such as patient's name, age, ID number and phone number. The second part is the diagnostic information prescribed by the doctor, including the patient's symptoms, the type of illness, the dose of medication and so on. Actually, only the diagnostic information is valuable to researchers, while researchers should not know patient's sensitive information by data sharing services [10, 11]. Therefore, it is very important to achieve data sharing under the condition that data deduplication is performed efficiently and the sensitive information of shared electronic medical records is hidden.

1.1 Contribution

To cope with this problem, we design a secure data sharing scheme for cloud-assisted electronic medical systems in this paper. To protect the privacy and enhance the deduplication efficiency, we replace the patient's sensitive information of electronic medical records by wildcards before encrypting whole electronic medical records. After that, the encrypted electronic medical records are uploaded to the cloud server so that it could not know anything useful about the electronic medical records. Any authorized researcher can decrypt and obtain the electronic medical records under the condition that the sensitive information of shared electronic medical records is hidden. Since the sensitive information is uniformly blinded with wildcards, the duplicate ratio of blinded sensitive information becomes higher. As a result, our scheme greatly improves the deduplication efficiency remarkably. Moreover, we clarify the diagnose information of the electronic medical records into three types according to duplicate ratios: high duplicate ratio, intermediate duplicate ratio and low duplicate ratio. The authorized researchers can selectively download data according to duplicate ratio of diagnostic information. Furthermore, the security of the key might be a bottleneck of the system. If the key server is compromised, the key may be leaked. In order to improve the security of the key, we employ proactive secret sharing technique [12] to resist brute-force attacks and single-point-of-failure attack. Security analysis shows that the proposed scheme ensures the confidentiality of shared electronic medical records. On the one hand, the sensitive information in the electronic medical records is confidential to researchers. Meanwhile, the integrity of the electronic medical records can be guaranteed. We also conduct experiments to evaluate the performance of the proposed scheme according to deduplication efficiency, storage efficiency, computational costs and computation delay.

1.1.1 Organization

The rest organization of the paper is as follows. In Sect. 2, we introduce the related work. In the next section, we show some necessary preliminaries. Section 4 gives the system model and the security model. We describe our secure data sharing scheme in Sect. 5 and give the security analysis in Sect. 6. Performance evaluation is shown in Sect. 7. In the last section, we conclude this paper.

2 Related work

With the security awareness increasing, more and more users would like to encrypt their data before the data are uploaded to the cloud server. Unfortunately, even if the identical data is encrypted, different users can produce different ciphertexts since they use different keys to encrypt the identical data. It leads to the cloud server unable to perform deduplication according to ciphertext of data. To cope with the problem of deduplication over encrypted data and improve the storage efficiency [13], convergent encryption (CE) has been proposed [14]. It requires that the encryption key is derived from the data itself. Bellare et al. [15] formalized a new primitive named as message-locked encryption (MLE) and proved that the MLE scheme is secure. Moreover, many data deduplication schemes based on convergent encryption and message-locked encryption have been proposed [16–19]. Unfortunately, these MLE-based and CE-based schemes cannot be against brute-force attacks [20]. Bellare et al. [21] proposed the server-aided deduplication scheme called DupLESS which introduces a fully trusted key server to produce MLE keys. After that, many deduplication schemes have been proposed to resist brute-force attacks [22–25], where the key server is fully trusted and will not be attacked by any adversary. Thereby, the completely trusted key server becomes single point of failure. More specifically, it is infeasible to introduce a completely trusted key server. It is hard to be equipped with such a fully trusted key server in reality. If the key server is broken down, the secret stored in the key server is exposed. In other words, attackers only need to compromise one key server if they want to obtain the

server-side secret. To resist single-point-of-failure attack, Zhang et al. [26] and Duan et al. [27] proposed to store the secret by multiple key servers using a threshold secret sharing [28]. Zhang et al. [12] improved the scheme of multiple key servers and proposed to replace some new key servers periodically. Let the new key servers store new secret shares while share the same server-side secret. It improves the security of server-side secret. As mentioned in [15], there exists a duplicate faking attack which causes legitimate users unable to obtain the correct data. More specifically, if attackers upload the tag of m but upload the ciphertext of m^* to the cloud server, it will make the tag inconsistent with the ciphertext m^*. As a result, the subsequent users who upload the tag of m will download the ciphertext of error data m^*. To solve this problem, tag consistency is considered to ensure data integrity [18, 29, 30]. Specifically, users verify tag consistency after they decrypt the returned ciphertext. If the verification fails, users cannot determine the cause of the failure. It may be because the uploaded data is fake, or the cloud server has destroyed the data. Li et al. [31] and Yang et al. [32] proposed that the tag is generated directly from the ciphertext, while the cloud server checks tag consistency in the phase of data upload. As mentioned above, convergent encryption (CE) cannot be against brute-force attacks for predictable messages. Hence, data popularity is considered to preserve the privacy of the data [16]. More specifically, a data block is considered "popular" when it is owned by more than the popularity threshold number of users, otherwise it is considered "unpopular". Data blocks with different popularity are protected under encryption mechanisms with different security levels. Puzio et al. [33] proposed PerfectDedup scheme which takes advantage of the characteristics of perfect hash to ensure data confidentiality. In addition, Li et al. [34] introduced the concept of transparent integrity auditing which can keep the cloud server from misbehaving. In order to audit the integrity of user's data without downloading them, Shao et al. [35] propose an efficient TPA-based auditing scheme for secure cloud storage.

As we all know, the electronic medical record is composed of patient's sensitive information and diagnostic information. In general, the duplicate ratio of the sensitive information is low, while the duplicate ratio of the diagnostic information is high. Based on this feature of the electronic medical record, Zhang et al. [26] designed a deduplication scheme that only performs deduplication for diagnostic information. Nonetheless, the sensitive information is all stored in the cloud server which increases the storage costs of the cloud server. Furthermore, this scheme doesn't support data sharing. Shen et al. [36] designed a scheme for data sharing with sensitive information hiding. Although the sensitive information is hidden for researchers, the sensitive information is stored repeatedly. Because this scheme does not support deduplication, the storage efficiency of this scheme is not high.

As far as we know, all above schemes cannot achieve secure data sharing supporting efficient deduplication and sensitive information hiding. In this paper, we explore how to achieve secure data sharing with sensitive information hiding, meanwhile we try to improve the deduplication efficiency.

3 Preliminaries

In this section, we introduce the basic knowledge needed in this paper, including MLE, bilinear map, discrete logarithm problem and computational Diffie-Hellman problem.

3.1 Message-locked encryption

Message-locked encryption (MLE) is a special symmetric encryption, which is widely adopted in deduplication. The encryption key and decryption key are computed from messages themselves. The same message will correspond the same key and the same ciphertext no matter who runs this encryption method. MLE can be expressed as a tuple MLE = (P, K, E, D) containing four algorithms:

$P \leftarrow \$\mathrm{P}$: It is parameter generation algorithm. It outputs a public parameter P which is published to all users.
$K \leftarrow \mathrm{K}(P, M)$: It is key generation algorithm. It inputs the public parameter P and the message M, and outputs the MLE key K. The same message M produces the same MLE key K.
$C \leftarrow \mathrm{E}(P, K, M)$: It is encryption algorithm. It inputs the public parameter P, the MLE key K and the message M, and outputs the ciphertext C.
$M \leftarrow \mathrm{D}(P, K, C)$: It is decryption algorithm. It inputs the public parameter P, the MLE key K and the ciphertext C, and outputs the plaintext M.

Any attacker cannot distinguish between ciphertext generated by MLE and random string of unpredictable information except with negligible possibility [15]. A special manifestation of MLE is convergent encryption (CE), in which the key is the hash value of the message.

3.2 Bilinear map

Let G and G_T be an addictive cycle group and a multiplicative cycle group of large prime order p, respectively. Let P be a generator of G. Bilinear Map is a map $e : G \times G \rightarrow G_T$, and its properties are as follows:

(1) Bilinearity: for all $a, b \in Z_p^*$, and $A, B \in G$, we have $e(aA, bB) = e(A, B)^{ab}$.

(2) Nondegeneracy: for all $A, B \in G$, and $A \neq B, e(A, B) \neq 1$.
(3) Computability: it is easy to calculate the map e.

3.3 Discrete logarithm (DL) problem

Let G be an addictive cycle group of large prime order p and P be its generator. For an unknown $x \in Z_p^*$, given xP as input, the aim is to output x. The DL assumption is true only if no algorithm can output x in polynomial time.

3.4 Computational Diffie–Hellman (CDH) problem

Let G be an addictive cycle group of large prime order p and P be its generator. For unknown $x, y \in Z_p^*$, given (P, xP, yP) as input, the aim is to output xyP. The CDH assumption is true only if no algorithm can output xyP in polynomial time.

4 System model and secure model

4.1 System model

The system model involves four entities: hospital, cloud server, key servers and researchers, as shown in Fig. 1.

- Hospital: The hospital is a fully trusted entity. The hospital produces huge amounts of electronic medical records every day that need to be moved to the cloud server for sharing with researchers. To ensure the security of the electronic medical records, the hospital needs to interact with key servers to produce the MLE key which is used to encrypt these electronic medical records. To protect the electronic medical record, the hospital will not collude with any attacker. If a researcher wants to obtain the electronic medical records, he/she firstly needs to get authorization from the hospital.
- Cloud server: The cloud server owns huge storage space and powerful computing ability. Meanwhile, the cloud server has the functions of identifying identity and performing deduplication. It also calculates and updates the statistical information of electronic medical records in real time. Besides, the hospital shares the electronic medical records with researchers through the cloud server under the condition of the privacy of patients being protected.
- Key servers: Key servers are independent of the hospital, researchers and the cloud server. They use (t, n)-threshold secret sharing to share a constant server-side secret, which is used to compute the MLE key. Each key server stores a share of the server-side secret. The hospital needs to generate the MLE key by interacting with the n key servers. These n key servers

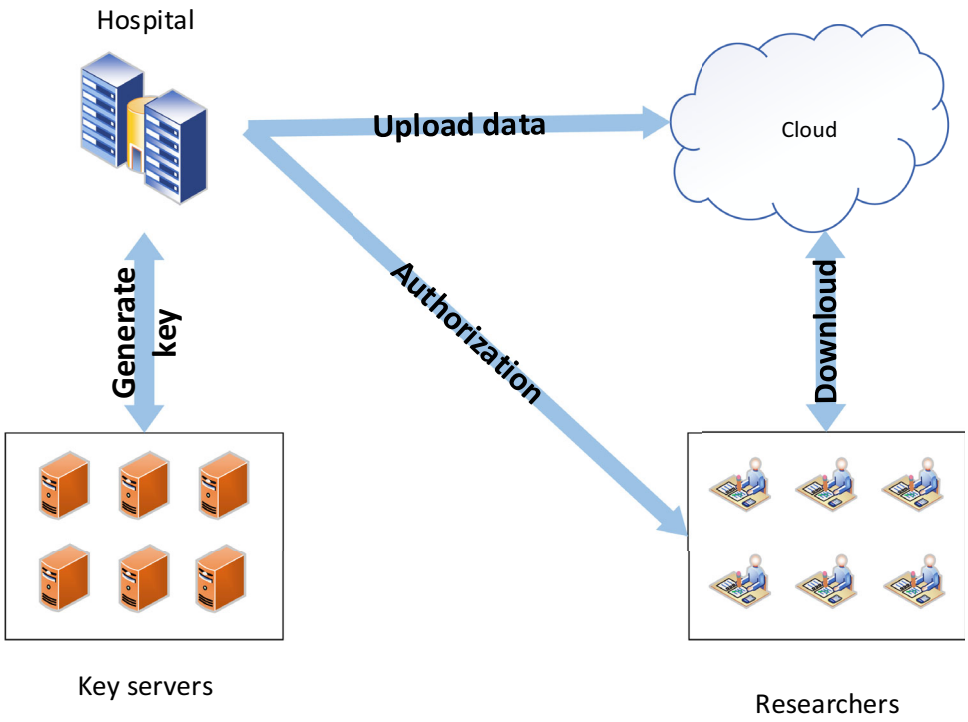

Fig. 1 System model

are not fully trusted, which means that one or more of them may be compromised.
- Researchers: Researchers need to obtain authorization from the hospital and then download electronic medical records from the cloud server for their research.

4.2 Adversary model

We mainly consider two kinds of adversaries: internal adversaries and external adversaries in the adversary model.

4.2.1 Internal adversaries

- The cloud server: Similar to the existing literature for cloud computing security [29, 37, 38], we assume that the cloud server will perform its task honestly for the sake of its reputation, but it is curious about the electronic medical records uploaded by the hospital and tries to obtain the plaintext of the electronic medical records.
- Researchers: We assume that authorized researchers will not collude with the hospital, the cloud server, unauthorized researchers and adversaries. Authorized researchers can download electronic medical records from the cloud server and obtain disease-related information in electronic medical records. Meanwhile, researchers are curious about sensitive information of patients, such as the patient's name and phone number.
- The compromised key servers: Adversaries may extract the secret shares from the compromised key servers and then launch brute-force attacks to guess the plaintext of the electronic medical records. Once the server-side secret is exposed, the security of electronic medical records will be difficult to guarantee.
- The hospital: When the hospital uploads data to the cloud server, data error may happen. For example, after uploading a tag, the hospital uploads the incomplete data or other non-corresponding data. It will lead to the inconsistency between the data and the tag. If the cloud server stores error data, it will not be able to rectify the data in the future since the cloud server will perform deduplication and only store one copy. As a result, the error data will affect the reliability of data seriously and further mislead the researchers.

4.2.2 External adversaries

We assume that external adversaries can obtain valuable information by monitor the communication between various entities. Based on this information, they try to recover the plaintext of the electronic medical records.

4.3 Design goals

In order to resist the above adversaries, we list the goals the proposed scheme should achieve.

- Data confidentiality: Data confidentiality is a fundamental requirement of the deduplication scheme. Due to the particularity of electronic medical records, the cloud server and unauthorized researchers cannot obtain the plaintext of electronic medical records. Moreover, even authorized researchers cannot obtain the sensitive information of patients in electronic medical records.
- Resistance to brute-force attacks and single-point-of-failure attack: Even if brute-force attacks are launched, any adversary cannot obtain valuable information. Moreover, if an adversary attacks a key server and extracts the server-side secret share from this key server, he/she also cannot recover the MLE key.
- Data integrity: Both the cloud server and authorized researchers can check the integrity of data. Moreover, if the data downloaded by the authorized researchers is error, the researchers can determine if the error data is from duplicate faking attacks or destroyed by the cloud server.
- Downloading selectivity: The authorized researchers can download electronic medical records selectively according to duplicate ratio.
- Efficiency: On the premise of ensuring security, the efficiency is our goal, including deduplication efficiency, computing efficiency and storage efficiency.

5 The proposed scheme

5.1 Overview

Electronic medical records contain disease-related information which is very helpful for researchers to do research on the disease. Hence, it is necessary for hospital to share electronic medical records with researchers. However, electronic medical records also contain patient's sensitive information that should not be exposed to researchers. In order to hide the patient's sensitive information, we design a secure data sharing scheme. Before uploading the electronic medical record, the hospital firstly blinds the sensitive information of the electronic medical record with wildcards. Then the hospital interacts with the key servers to generate the MLE key. The hospital divides the electronic medical record into multiple data blocks, and then encrypts them with the corresponding MLE keys to obtain the ciphertexts. Compared with file-level deduplication, chunk-level deduplication has higher deduplication

efficiency. The hospital derives a tag from the ciphertext of a data block, which can be used for duplicated data detection and data integrity verification. The hospital sends the tag to the cloud server. The cloud server checks whether the data block is duplicated with this tag. If the data block is not duplicated, it means that the hospital needs to upload the data block. In this way, anybody cannot obtain patient's sensitive information from the electronic medical records downloaded from the cloud server. After receiving the ciphertext of the data block, the cloud server makes deduplication operation to save storage space. Since the sensitive information of electronic medical records is uniformly blinded with wildcards, the duplicate ratio of the sensitive information is higher in our scheme. In addition to performing deduplication, the cloud server calculates and updates the duplicate ratio of the diagnose information of electronic medical records. We divide the diagnose information of electronic medical records into three categories according to the duplicate ratio: high duplicate ratio, intermediate duplicate ratio and low duplicate ratio. In this way, the authorized researchers can download electronic medical records selectively according to duplicate ratio. For example, the COVID-19 electronic medical records contain multiple clinical symptoms. Some symptoms like fever and cough are very common but some symptoms like hypoglycemia only appear in a few electronic medical records. Symptoms with different duplicate ratio have different research values for researchers. If researchers want to research the common symptoms of COVID-19, they can choose to only download the electronic medical records with high duplicate ratio. On the one hand, it reduces the interference of the electronic medical records with low duplicate ratio to researchers. On the other hand, it reduces communication costs since researchers do not need to download all electronic medical records.

5.2 Construction of our scheme

5.2.1 System setup

a) The system chooses an additive cycle group G and a multiplicative cycle group G_T of prime order p. P is a generator of G, and $e : G \times G \rightarrow G_T$ is a bilinear map. The system publishes parameters (G, G_T, p, P, e).
b) The hospital HP selects an anti-collision hash function $H : \{0,1\}^* \rightarrow G$.
c) The hospital HP randomly chooses a master key msk.
d) The hospital HP chooses a symmetric encryption algorithm $E(\bullet)$ and a public-key encryption algorithm $Enc(\bullet)$.

5.2.2 Sensitive information hiding

a) After generating an electronic medical record F, the hospital HP firstly replaces patient's sensitive information in the electronic medical record F with wildcards "*". We show an example for electronic medical record in Fig. 2. We denote the new file after this process as F^*.
b) The hospital HP divides F^* into multiple fixed size blocks $(m_1^*, m_2^*, \ldots, m_l^*)$, where l represents the number of data blocks.

5.2.3 Key servers initialization

(a) Assume $\{KS_1, KS_2, \ldots, KS_n\}$ is the set of n mutually independent key servers. We divide the total time into multiple fixed length intervals called epochs.
(b) For $k = 1, 2, \ldots, n$, KS_k randomly selects t elements $a_{k,0}, a_{k,1}, a_{k,2}, \ldots, a_{k,t-1} \in Z_p^*$, and constructs a polynomial $F_k(x) = a_{k,0} + a_{k,1}x + a_{k,2}x^2 + \ldots + a_{k,t-1}x^{t-1} modp$.
(c) KS_k calculates $F_k(j)$ and sends $F_k(j)$ to KS_j secretly, for $j = 1, 2, 3, \ldots, n and j \neq k$. KS_k calculates commit set $\{a_{k,0}P, a_{k,1}P, a_{k,2}P, \ldots, a_{k,t-1}P\}$ and publishes them to all other key servers.
(d) After receiving $F_k(j)$, KS_j needs to verify the validity of $F_k(j)$ by checking whether $F_k(j)P = \sum_{\xi=0}^{t-1} k^{\xi} \cdot a_{k,\xi}P$ holds. If the equation holds, KS_j accepts $F_k(j)$, otherwise rejects $F_j(k)$.
(e) When receiving $(n-1)$ correct $F_k(j)$ from $KS_k (k = 1, 2, 3, \ldots, n; k \neq j)$, KS_j calculates its secret share $s_j = F_1(j) + F_2(j) + F_3(j) + \ldots + F_n(j)$.
(f) Let $s = a_{1,0} + a_{2,0} + a_{3,0} + \ldots + a_{n,0}$. According to Lagrange interpolation, we know $s = \sum_{\varsigma=1}^{t} \omega_{\varsigma} \cdot s_{\varsigma}$, where $\omega_{\varsigma} = \prod_{1 \leq \eta \leq t, \eta \neq \varsigma} \frac{\eta}{\eta - \varsigma}$.
(g) KS_j calculates the commit $Q_j = s_j \bullet P$ for share s_j and the public value $Q = \sum_{\epsilon=1}^{n} a_{\epsilon,0} \bullet P$.
(h) Without loss of generality, we assume that the key servers in the $\mathcal{X}$-th epoch are $KS^{(\mathcal{X})} = \left\{KS_1^{(\mathcal{X})}, KS_2^{(\mathcal{X})}, KS_3^{(\mathcal{X})}, \ldots, KS_n^{(\mathcal{X})}\right\}$, and the key servers in the $(\mathcal{X}+1)$-th epoch are $KS^{(\mathcal{X}+1)} = \{KS_1^{(\mathcal{X}+1)}, KS_2^{(\mathcal{X}+1)}, KS_3^{(\mathcal{X}+1)}, \ldots, KS_n^{(\mathcal{X}+1)}\}$. Firstly, t honest and reliable key servers are selected from $KS^{(\mathcal{X})}$, expressed as $\{KS_{i1}^{(\mathcal{X})}, KS_{i2}^{(\mathcal{X})}, KS_{i3}^{(\mathcal{X})}, \ldots, KS_{it}^{(\mathcal{X})}\}$. The corresponding secret shares are $\{s_{i1}^{(\mathcal{X})}, s_{i2}^{(\mathcal{X})}, s_{i3}^{(\mathcal{X})}, \ldots, s_{it}^{(\mathcal{X})}\}$. For $\alpha = 1, 2, 3, \ldots, t$, $Q_{i\alpha}^{(\mathcal{X})} = s_{i\alpha}^{(\mathcal{X})} \cdot P$ has been published.

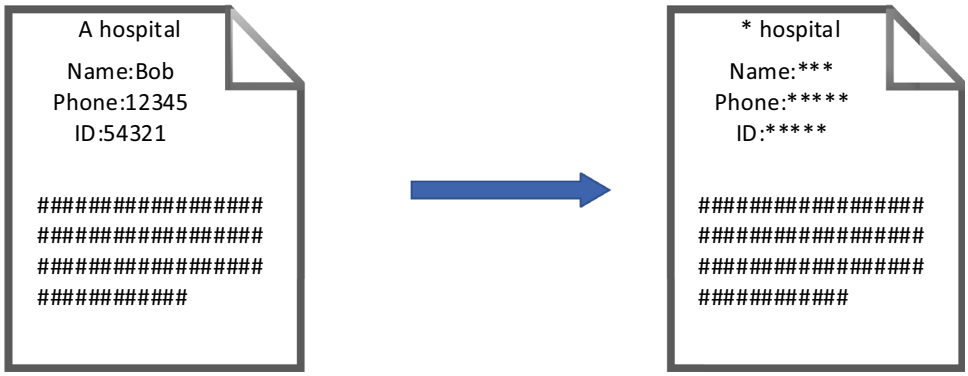

Fig. 2 An example of the processed electronic medical record

(i) $KS_{i\alpha}^{(\mathcal{X})}$ randomly selects $(t-1)$ elements $b_{i\alpha,1}, b_{i\alpha,2}, b_{i\alpha,3}, \ldots, b_{i\alpha,t-1}$, and constructs a function $g_{i\alpha}^{(\mathcal{X})}(x) = s_{i\alpha}^{\mathcal{X}} + b_{i\alpha,1} \cdot x + b_{i\alpha,2} \cdot x^2 + \ldots + b_{i\alpha,t-1} \cdot x^{t-1} modp$. Here, $KS_{i\alpha}^{(\mathcal{X})}$ computes and publishes commits $b_{i\alpha,1} \cdot P, b_{i\alpha,2} \cdot P, b_{i\alpha,3} \cdot P, \ldots, and b_{i\alpha,t-1} \cdot P$, and computes $s_{i\alpha,\beta}^{(\mathcal{X})} = g_{i\alpha}^{(\mathcal{X})}(\beta)$ for $\beta = 1, 2, 3, \ldots, n$.

(j) $KS_{i\alpha}^{(\mathcal{X})}$ sends $s_{i\alpha,\beta}^{(\mathcal{X})}$ to $KS_{\beta}^{(\mathcal{X}+1)}$ through a secure channel. Then $KS_{\beta}^{(\mathcal{X}+1)}$ checks the validity of $s_{i\alpha,\beta}^{(\mathcal{X})}$ by verifying whether $s_{i\alpha,\beta}^{(\mathcal{X})} \cdot P = s_{i\alpha}^{(\mathcal{X})} \cdot P + \sum_{\gamma=1}^{t-1} \beta^{\gamma} \cdot b_{i\alpha,\gamma} \cdot P$ holds. If the checking failed, $KS_{\beta}^{(\mathcal{X}+1)}$ aborts; otherwise, $KS_{\beta}^{(\mathcal{X}+1)}$ sends "Accept" to all other key servers in the $(\mathcal{X}+1)$-th epoch.

(k) After getting all "Accept" from other key servers, $KS_{\beta}^{(\mathcal{X}+1)}$ has received all correct shares $s_{i1,\beta}^{(\mathcal{X})}, s_{i2,\beta}^{(\mathcal{X})}, s_{i3,\beta}^{(\mathcal{X})}, \ldots, s_{it,\beta}^{(\mathcal{X})}$. $KS_{\beta}^{(\mathcal{X}+1)}$ computes its secret share $s_{\beta}^{(\mathcal{X}+1)} = \sum_{\varsigma=1}^{t} \omega_{i\varsigma} \cdot s_{i\varsigma,\beta}^{(\mathcal{X})}$, where $\omega_{i\varsigma} = \prod_{\substack{1 \le \eta \le t \\ \eta \ne i\varsigma}} \frac{\eta}{\eta - i\varsigma}$. Finally, $KS_{\beta}^{(\mathcal{X}+1)}$ computes and publishes public commit $Q_{\beta}^{(\mathcal{X}+1)} = s_{\beta}^{(\mathcal{X}+1)} \cdot P$.

Remark 1 Although key servers have been replaced by new key servers and the secret shares stored in the new key servers have changed in the end of each epoch, the server-side secret s shared by the new key servers has not changed. As we know, in the $\mathcal{X}$-th epoch, $s = \sum_{\varsigma=1}^{t} \omega_{\varsigma} \cdot s_{\varsigma}^{(\mathcal{X})} = \sum_{\varsigma=1}^{t} \omega_{\varsigma} \cdot \sum_{\beta=1}^{t} \omega'_{\beta} \cdot s_{\varsigma,\beta}^{(\mathcal{X})} = \sum_{\beta=1}^{t} \sum_{\varsigma=1}^{t} \omega_{\varsigma} \cdot \omega'_{\beta} \cdot s_{\varsigma,\beta}^{(\mathcal{X})} = \sum_{\beta=1}^{t} \omega'_{\beta} \cdot s_{\beta}^{(\mathcal{X}+1)}$, where ω_{ς} and ω'_{β} are Lagrange coefficients. In the $(\mathcal{X}+1)$-th epoch, $s' = \sum_{\beta=1}^{t} \omega'_{\beta} \cdot s_{\beta}^{(\mathcal{X}+1)}$. So, $s = s'$, which means that the server-side secret s has not changed.

5.2.4 MLE key generation

(a) The hospital HP randomly selects an element $r \in Z_p^*$, and computes $m'_i = r \cdot H(m_i^*)$ for $i = 1, 2, 3, \ldots, n$. Then HP sends m'_i to all key servers.

(b) For $k = 1, 2, 3, \ldots, n$, KS_k computes $\sigma_i^{(k)} = s_k \cdot m'_i$ and returns $\sigma_i^{(k)}$ to HP.

(c) HP checks the validity of $\sigma_i^{(k)}$ by verifying whether $e\left(\sigma_i^{(k)}, P\right) = e(m'_i, Q_k)$ holds. If the equation doesn't hold, HP rejects $\sigma_i^{(k)}$.

(d) After receiving t effective $\{\sigma_i^{(i1)}, \sigma_i^{(i2)}, \sigma_i^{(i3)}, \ldots, \sigma_i^{(it)}\}$, where $i1, i2, i3, \ldots, it \in [1, n]$, HP computes $\sigma_i = r^{-1} \cdot \sum_{\delta=i1}^{it} \omega_{\delta} \cdot \sigma_i^{\delta} = s \cdot H(m_i^*)$.

(e) HP computes the MLE key of the data block m_i^* as $K_i = H(\sigma_i)$.

Remark 2 Each hospital will select a different element r to compute different m'_i. Thus, the key server cannot determine whether the data block m_i^* is identical or not according to m'_i. Since $\sigma_i = s \cdot H(m_i^*)$, σ_i is only related to the server-side secret s and the data block m_i^*. As we know, the server-side secret s is constant. If the data block m_i^* is identical, each hospital will produce the same σ_i. Since $K_i = H(\sigma_i)$, each hospital will obtain the same MLE key K_i. Each hospital encrypts the same data block and computes the data tag with the same MLE key to obtain the same data ciphertext and the same data tag. Therefore, the cloud server can perform data deduplication on the data uploaded by different hospitals.

5.2.5 Data upload and deduplication

The sensitive information in electronic medical records has high duplicate ratio after it is uniformly blinded with wildcards. Hence, the category of sensitive information is worthless for researchers. In order to exclude the influence of the category of sensitive information, we design

different upload processes for sensitive information and diagnostic information.

5.2.5.1 Sensitive information

(a) After getting MLE keys, *HP* firstly encrypts the data block m_i^* for $i = 1, 2, 3, \ldots, n$ as $C_i = E(K_i, m_i^*)$. Then *HP* computes $\tau_i = H(C_i)$ as the tag of the data block m_i^*. Finally, *HP* sends these τ_i to the cloud server *CS*.

(b) *CS* maintains a set $\{\tau\}$ for data blocks of sensitive information. After receiving the tag τ_i, *CS* checks whether τ_i exists in the set $\{\tau\}$. If *CS* finds the tag τ_i, then aborts. Otherwise, goes to the next step.

(c) *HP* calculates the ciphertext $CK_i = E(msk, K_i)$ of the MLE key, and then uploads (C_i, CK_i) to *CS*.

(d) After receiving (C_i, CK_i), *CS* verifies the integrity of C_i through checking $H(C_i) = \tau_i$. If the verification passes, *CS* stores (C_i, CK_i), otherwise *CS* aborts.

5.2.5.2 Diagnostic information

(a) After getting MLE keys, *HP* firstly encrypts the data block m_i^* for $i = 1, 2, 3, \ldots, n$ as $C_i = E(K_i, m_i^*)$. Then *HP* computes $\tau_i = H(C_i)$ as the tag of the data block m_i^*. Finally, *HP* sends these τ_i to the cloud server *CS*.

(b) *CS* maintains a tuple $T = (\tau, \rho, \lambda, \varphi)$ for each unique data block of diagnostic information, where τ is the tag of the data block, ρ is the number of duplicates of the data block, λ is the duplicate ratio of the data block in all files, and φ is the category of the data block. The data blocks are divided into three categories: low duplicate ratio, intermediate duplicate ratio and high duplicate ratio which are represented by -1, 0, 1 respectively. *CS* maintains a value f to record the number of files stored currently. *CS* sets two thresholds t' and t'' to distinguish the categories of data blocks.

(c) After receiving the tag τ_i, *CS* checks whether τ_i exists in the tuple set $\{T\}$. If *CS* doesn't find τ_i, *HP* needs to perform the next step d). otherwise, the step d) g) and h) will be skipped.

(d) *CS* constructs a tuple $T_i = (\tau_i, \rho_i, \lambda_i, \varphi_i)$ for τ_i and sets $\tau_i = \rho_i = \lambda_i = 0, \varphi_i = -1$.

(e) For each received tag τ_i, *CS* sets $\rho_i = \rho_i + 1$. After receiving all the tags of a file, *CS* sets $f = f + 1$ and calculates the duplicate ratio of each data block $\lambda_i = \frac{\rho_i}{f}$.

(f) *CS* updates the categories based on the duplicate ratio of data blocks. If $\lambda_i < t'$, the cloud server sets $\varphi_i = -1$; if $t' \le \lambda_i < t''$, *CS* sets $\varphi_i = 0$; if $\lambda_i \ge t''$, *CS* sets $\varphi_i = 1$.

(g) *HP* calculates the ciphertext of the MLE key $CK_i = E(msk, K_i)$, and then uploads (C_i, CK_i) to *CS*.

(h) After receiving (C_i, CK_i), *CS* checks the integrity of C_i through verifying whether $H(C_i) = \tau_i$ holds. If the verification passes, *CS* stores (C_i, CK_i), otherwise *CS* aborts.

5.2.6 Authorization and data download

The hospital *HP* randomly selects an element $x \in Z_p^*$, calculates and publishesxP. The researcher *R* randomly selects an element$y \in Z_p^*$, calculates and publishes yP.

(a) *R* interacts with *HP* to obtain authorization to access the electronic medical records. *HP* calculates $HPR = msk \cdot x \bullet (yP)$ and then sends $(HPR, \{\tau\})$ to *R*.

(b) *R* calculates $msk = HPR \bullet (y \bullet xP)^{-1}$ and then sends $\{\tau\}$ and $\{\varphi\}$ to *CS* where φ is data category. For example, if *R* wants to download the electronic medical records with the category of very common, *R* sets $\varphi = 1$ and sends φ to *CS*.

(c) After receiving $\{\tau\}$ and $\{\varphi\}$, *CS* firstly finds out the tags $\{\tau\}$ that match the categories $\{\varphi\}$ according to the tuple T. Then *CS* picks out the electronic medical records that contain the tags $\{\tau\}$. Finally, *CS* returns the ciphertexts $\{(C, CK)\}$ of the data blocks that make up these electronic medical records to *R*.

(d) *R* verifies the integrity of *C* by checking $H(C) = \tau$. If the verification fails, *R* aborts. Otherwise, *R* calculates $K = D(msk, CK)$ and $m^* = D(K, C)$.

6 Security analysis

We analyze the security of our proposed scheme from the aspects including resistance to brute-force attacks and single-point-of-failure attack, data confidentiality and data integrity.

6.1 Resistance to Brute-force attacks and single-point-of-failure attack

First, we analyze that attacker cannot launch brute-force attacks to obtain the plaintext information of data. Specifically, for a given ciphertext of data block C_i, the attacker wants to determine which ciphertext of data block C_k^* corresponds to the ciphertext C_i. More specifically, for each candidate data block m_k^*, the attacker calculates its corresponding ciphertext C_k, and then compares C_k with C_i to determine which data block m_k^* is the plaintext of C_i. However, as shown in Sect. 5.2.5, if the attacker wants to encrypt the data block m_k^*, he/she must obtain the MLE key

K_i, where $K_i = H(\sigma_i)$. Hence, the attacker must obtain the secret σ_i stored in the key servers. Since the key servers are independent and the attacker cannot collude with the key servers, the attacker cannot obtain the server-side secret σ_i. So, the attacker cannot obtain the MLE key K_i and calculate the ciphertexts of the candidate data blocks. Therefore, the attacker cannot launch brute-force attacks.

Next, we analyze that our scheme can remove the single-point-of-failure problem. In order to obtain σ_i, the attacker attacks one key server, and then finds the secret share from the information stored in the compromised key server. As shown in Sect. 3.2.3, the secret σ_i is stored in n key servers in the form of (t, n)-threshold blind tag. The server-side secret σ_i can be recovered only after at least t secret shares are obtained. If the number of compromised key servers does not reach t, the attacker cannot obtain enough information from these compromised key servers to recover σ_i. In addition, our scheme divides the whole time into segments of fixed duration, called epochs. At the end of each epoch, we will select a batch of new key servers to store the server-side secret σ_i, where the server-side secret σ_i has not changed but the secret shares stored in new key servers have changed. In other words, at the end of each epoch, all secret shares stored by the compromise key servers acquired by the attacker will expire, which further enhances the security of the server-side secret σ_i. Therefore, our scheme can remove single-point-of-failure problem.

6.2 Data confidentiality

In this section, we show that no adversary, including the Cloud Server Provider (CSP), unauthorized researchers and external adversary, can get the plaintext of the data stored in the cloud server.

Theorem 1 *For the CSP, the proposed scheme satisfies semantic security as long as the Discrete Logarithm (DL) assumption holds.*

Proof We define a polynomial-time adversary $\mathcal{A}$ to simulate the corrupted semi-trusted CSP.

Assume the adversary $\mathcal{A}$ can obtain the plaintext from the ciphertext C with advantage $\epsilon(\kappa)(\kappa$ is a secure parameter). We construct an algorithm$\mathcal{B}$, which can break the DL assumption with the same advantage as the adversary$\mathcal{A}$. The follows are the details of the security game:

Init: Let (G, G_T, p, P, e) be a tuple published by the system, where p is the prime order of G and G_T, and P is the generator of G. $\mathcal{B}$ randomly chooses a server-side secret $s \in \mathbb{Z}_p^*$ and computes the public value $Q = s \cdot P \in G$. $\mathcal{B}$ generates a subgroup $G_s = \{s_1, s_2, \ldots\}$ in G, where all elements satisfy the equation $Q = s_i \cdot P (i = 1, 2, \ldots)$ and $s \notin G_s$. Given the parameters y, where $y = s$ or $y \in G_s$, $\mathcal{B}$ gives the parameters (G, G_T, p, P, e, Q, y) to $\mathcal{A}$.

Challenge $\mathcal{A}$ constructs two data blocks $m_0, m_1 \in G$, and then sends them to $\mathcal{B}$. $\mathcal{B}$ selects a uniformly random bit $b \in \{0, 1\}$, and then computes the encryption key $K = H(s \cdot H(m_b))$. Finally, $\mathcal{B}$ outputs the ciphertext $C_b = E(K, m_b)$.

Guess $\mathcal{A}$ generates a prediction b' of b. If $b' = b$, $\mathcal{B}$ returns 1 to denote that $y = s$; otherwise, $\mathcal{B}$ returns 0 to denote that y is a random element from the subgroup G_p.

Based on the assumption, $\mathcal{A}$ can obtain server-side secret s from the public value $Q = s \cdot P$ with advantage $\epsilon(\kappa)$ when $y = s$, and then $\mathcal{A}$ can obtain the encryption key $K = H(s \cdot H(m_b))$. Finally, $\mathcal{A}$ can obtain m_b with the encryption key K by computing $m_b = D(K, C_b)$. Hence, we know $\Pr[\mathcal{A}(\mathrm{b}' = \mathrm{b})] = \frac{1}{2} + \epsilon(\kappa)$. Since $\mathcal{B}$ returns 1 only when the prediction b' of $\mathcal{A}$ is equal to b, we know $\Pr[\mathcal{B}(G, G_T, P, Q, y) = 1 : y = s] = \Pr[\mathcal{A}(b' = b)] = \frac{1}{2} + \epsilon(\kappa)$.

When y is a random element from G_s, y is evenly distributed in G_s and is unaffected by b. Therefore, we have $\Pr[\mathcal{A}(b' = b)] = \frac{1}{2}$, which indicates that $\Pr[\mathcal{B}(G, G_T, P, Q, y) = 0 : y \in G_s] = \Pr[\mathcal{A}(b' = b)] = \frac{1}{2}$. So we can obtain that $DL - Adv_{\mathcal{B}} = |\Pr[\mathcal{B}(G, G_T, P, Q, y) = 1 : y = s] - \Pr[\mathcal{B}(G, G_T, P, Q, y) = 0 : y \in G_s]| = \left|\frac{1}{2} - \left(\frac{1}{2} + \epsilon(\kappa)\right)\right| = \epsilon(\kappa)$, which implies the advantage that polynomial-time algorithm $\mathcal{B}$ determines whether $y = s$ is $\epsilon(\kappa)$. Based on the DL assumption, we can find that the advantage $\epsilon(\kappa)$ is negligible.

In our scheme, the hospital *HP* authorizes the researchers through the master key *msk*. Hence, the only way for unauthorized researchers to access electronic medical records is to extract *msk* from $HPR = msk \cdot x \cdot y \cdot P$. Next, we prove that our scheme is semantically secure for unauthorized researchers.

Theorem 2 *For unauthorized researchers, the proposed scheme satisfies semantic security as long as the Computational Diffie-Hellman (CDH) assumption holds.*

Proof We define a polynomial-time adversary $\mathcal{A}$ to simulate the corrupted unauthorized researchers. Assume the adversary $\mathcal{A}$ can obtain the master key *msk* from the ciphertext *HPR* with advantage $\epsilon(\kappa)$. We construct an algorithm $\mathcal{B}$, which can break the CDH assumption with

the same advantage as the adversary $\mathcal{A}$. The follows are the details of the security game:

Init Generate the parameters $(P, x \cdot P, y \cdot P, W)$, where P is the generator of G, unknown $x, y \in \mathbb{Z}_p^*$, $x \cdot P, y \cdot P \in G_T and W \in G_T$. $\mathcal{B}$ gives the public parameters $(G, G_T, p, P, e, x \cdot P, y \cdot P)$ to $\mathcal{A}$.

Challenge $\mathcal{A}$ constructs two data blocks $m_0, m_1 \in G$, and then sends them to $\mathcal{B}$. $\mathcal{B}$ selects a uniformly random bit $b \in \{0, 1\}$, and returns the ciphertext $HPR = m_b \cdot W$.

Guess: $\mathcal{A}$ generates a prediction b' of b. If $b' = b$, $\mathcal{B}$ returns 1 to denote that $W = x \cdot y \cdot P$; otherwise, $\mathcal{B}$ returens 0 to denote that W is a random element from G_T.

Based on the assumption, $\mathcal{A}$ can obtain W from the tuple $(P, x \cdot P, y \cdot P)$ with advantage $\epsilon(\kappa)$ if $W = x \cdot y \cdot P$. Then $\mathcal{A}$ can obtain m_b with W by computing $m_b = HPR \cdot W^{-1}$. Hence, we have $\Pr[\mathcal{A}(b' = b)] = \frac{1}{2} + \epsilon(\kappa)$. Since $\mathcal{B}$ returns 1 only when the prediction b' of $\mathcal{A}$ is equal to b, we know $\Pr[\mathcal{B}(G, G_T, P, x \cdot P, y \cdot P, W) = 1] =$
$\Pr[\mathcal{A}(b' = b)] = \frac{1}{2} + \epsilon(\kappa)$.

When W is a random element from G_T, $m_b \cdot W$ is evenly distributed in G_T and is unaffected by b from $\mathcal{A}$'s view. Therefore, we have $\Pr[\mathcal{A}(b' = b)] = \frac{1}{2}$, which indicates that $\Pr[\mathcal{B}(G, G_T, P, x \cdot P, y \cdot P, W) = 0] = \frac{1}{2}$. So we can obtain that $CDH - Adv_{\mathcal{B}} = |\Pr[\mathcal{B}(G, G_T, P, x \cdot P, y \cdot P, W) = 1] -$
$\Pr[\mathcal{B}(G, G_T, P, x \cdot P, y \cdot P, W) = 0]| = |\frac{1}{2} +$
$\epsilon(\kappa) - \frac{1}{2}| = \epsilon(\kappa)$, which implies that the advantage of polynomial-time algorithm $\mathcal{B}$ to determine whether $W = x \cdot y \cdot P$ is $\epsilon(\kappa)$. Based on the CDH assumption described in Sect. 3.4, we can find that the advantage $\epsilon(\kappa)$ is negligible.

Theorem 3 *For external adversaries, the proposed scheme satisfies semantic security as long as the DL and CDH assumptions hold.*

Proof As we all know, external adversaries have less relevant information than internal adversaries. Hence, there is no doubt that external adversaries are unable to get the plaintext of outsourced data.

6.3 Data integrity

We will prove that our scheme can ensure data integrity, i.e., our scheme can ensure the integrity of both the uploaded data and the downloaded data. Specifically, assume that the hospital *HP* uploads the tag τ_i of data block m_i^*, and the cloud server does not find duplicate and requires *HP* to upload data, while *HP* intentionally or unintentionally uploads the ciphertext C_k of other data block m_k^*. If the cloud server does not check data integrity, it will store the error data C_k. When *HP* uploads the tag τ_i of data block m_i^* again, the cloud server will find duplicate and will not require *HP* to upload data. So C_k stored in the cloud server can never be corrected. When the researcher R would like to download the data C_i, he/she actually downloads the error data C_k. As a result, the researcher cannot decrypt m_i^* from C_k. Similarly, assume that *HP* uploads the data block C_i correctly. When the researcher R wants to download the ciphertext C_i, the cloud server returns the ciphertext C_k of other data blocks m_k^* to R. The researcher cannot decrypt m_i^* from C_k. Fortunately, our scheme can detect incomplete data. As shown in Sect. 3.2.5, before uploading the data block m_i^*, *HP* uploads its tag τ_i to the cloud server, where $\tau_i = H(C_i)$. When the cloud server receives the data block C_k, the cloud server checks whether $\tau_i = H(C_k)$ holds. If $\tau_i \neq H(C_k)$, then $C_k \neq C_i$. It indicates that the data is incomplete. Similarly, in the process of data sharing, *HP* sends the tag τ_i to the researcher R who checks whether $\tau_i = H(C_k)$ holds after downloading C_k from the cloud server. If $\tau_i \neq H(C_k)$, the data is incomplete.

7 Performance evaluation

To demonstrate the efficiency of our scheme, we conduct experiments using a real-world dataset [39]. This dataset contains about 300 electronic medical records which are divided into 6300 data blocks. Note that the electronic medical records of the dataset exclude all patient's sensitive information. We pad the patient's sensitive information to these records to form intact electronic medical records. The total sensitive information accounts for about 38% of whole electronic medical records. We do experiments in java running on a desktop computer with Windows OS system, a 2.10 GHz Inter Core i5 CPU and 8 GB memory. The security level is set to 64 bits. In order to intuitively show the efficiency of our scheme, we compare our scheme with Zhang et al.'s scheme [26] and Bellare et al.'s scheme [21] to evaluate the performance of these schemes in terms of efficiency of deduplication, storage costs, computational costs and computation delay. It should be noted that we ignore related evaluation about the phase of data download and data sharing since deduplication only happens in the phase of data upload.

7.1 Deduplication efficiency

Firstly, we compare our scheme with DupLESS scheme [21] and HealthDep scheme [26] with regard to the deduplication efficiency. As shown in Fig. 3, the deduplication efficiency of our scheme is the best in these three schemes, followed by DupLESS scheme, and the worst by HealthDep scheme. In the HealthDep scheme, because of the low duplicate ratio of patient's sensitive information in

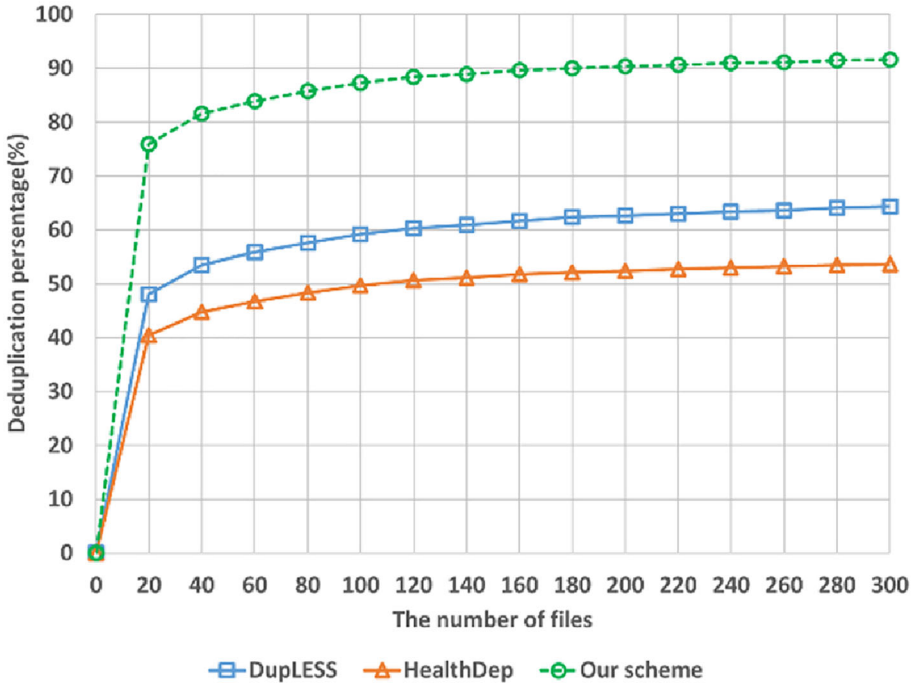

Fig. 3 efficiency of deduplication

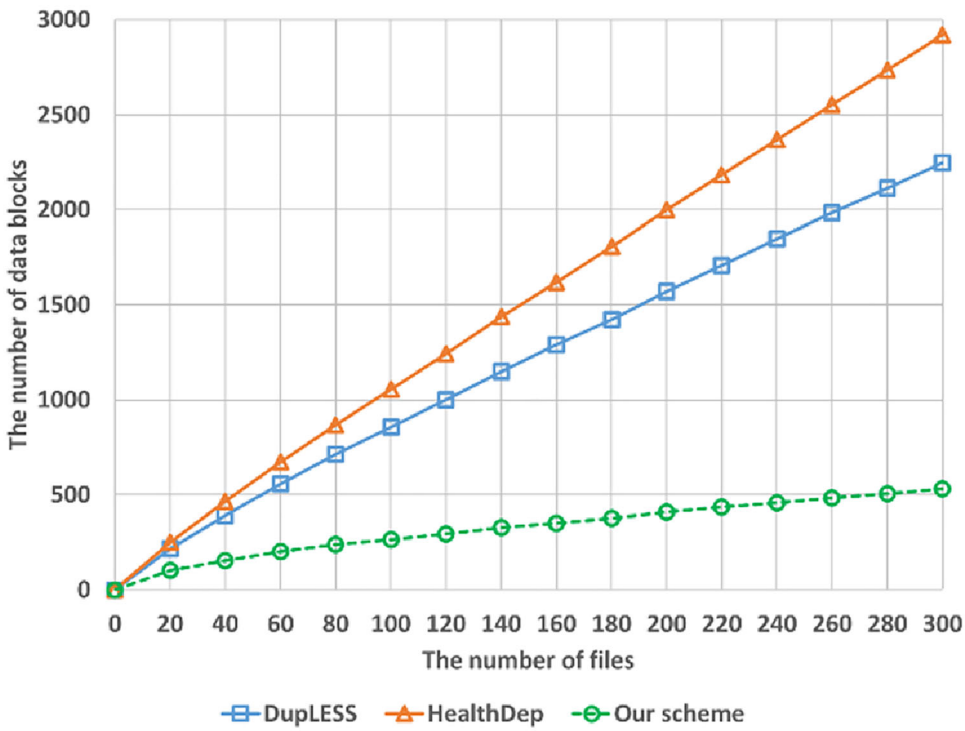

Fig. 4 Comparison of storage costs during the data upload phase

electronic medical record, the cloud server only performs deduplication on others except patient's sensitive information. The duplicate data in the sensitive information is not deleted, so the deduplication efficiency of HealthDep is the lowest. The DupLESS scheme does not consider protecting patients' sensitive information. So it views sensitive information and other information as equal, and performs deduplication directly on whole electronic medical record. The duplicate data in the sensitive information is deleted, thus the deduplication efficiency of DupLESS scheme is slightly higher than that of HealthDep scheme. In our scheme, before uploading electronic medical records, the

hospital firstly replaces patients' sensitive information with wildcards. After the sensitive information is converted to wildcards with the same length, the cloud server performs deduplication. Therefore, the duplicate ratio of sensitive information remarkably increases so that the deduplication efficiency also greatly increases.

7.2 Storage efficiency

Figure 4 shows the comparison results of storage efficiency. The more data blocks the cloud server stores, the lower the storage efficiency is. It can be seen clearly from Fig. 4 that the storage efficiency of cloud server is influenced by deduplication efficiency and the number of files. As shown in Fig. 4, the storage efficiency of our scheme is the best in these three schemes. Moreover, as the number of files increases, the storage efficiency gap among these three schemes becomes larger and larger. In these three schemes, the deduplication efficiency of our scheme is the highest, so the storage efficiency of our scheme is the highest. HealthDep scheme and our scheme all introduce n key servers to share a constant server-side secret like the schemes [12]. This secret sharing scheme can improve the security of the server-side secret, but it also needs additional storage space to store n secret shares. Fortunately, this additional storage space is insignificant. On the one hand, the secret share is derived from the server-side secret s. Thus, the size of the secret share is much smaller than that of the data. On the other hand, n secret shares are stored in the key servers instead of the hospital and the cloud server. In other words, the secret sharing scheme does not increase the storage overhead of the hospital and the cloud server.

7.3 Computational costs

We provide a comparison of computational costs in Fig. 5. We randomly select 10, 150 and 300 files from the dataset to do experiments. When the total number of files is relatively small, the computational time of these three schemes is shown in Fig. 5a. When the number of files is 0, the computational time of DupLESS scheme is the shortest, followed by HealthDep scheme and our scheme. The main reason is that HealthDep scheme and our scheme use multiple key servers to protect the server-side secret. In the process of system setup, n key servers interact each other to generate secret shares which consumes some time. In contrast, DupLESS scheme employs single key server which saves time for generating secret shares. As shown in Fig. 5b, when the total number of files increases, the computational costs of DupLESS scheme is the highest, followed by HealthDep scheme and our scheme. The main factor is that HealthDep scheme and our scheme adopt the

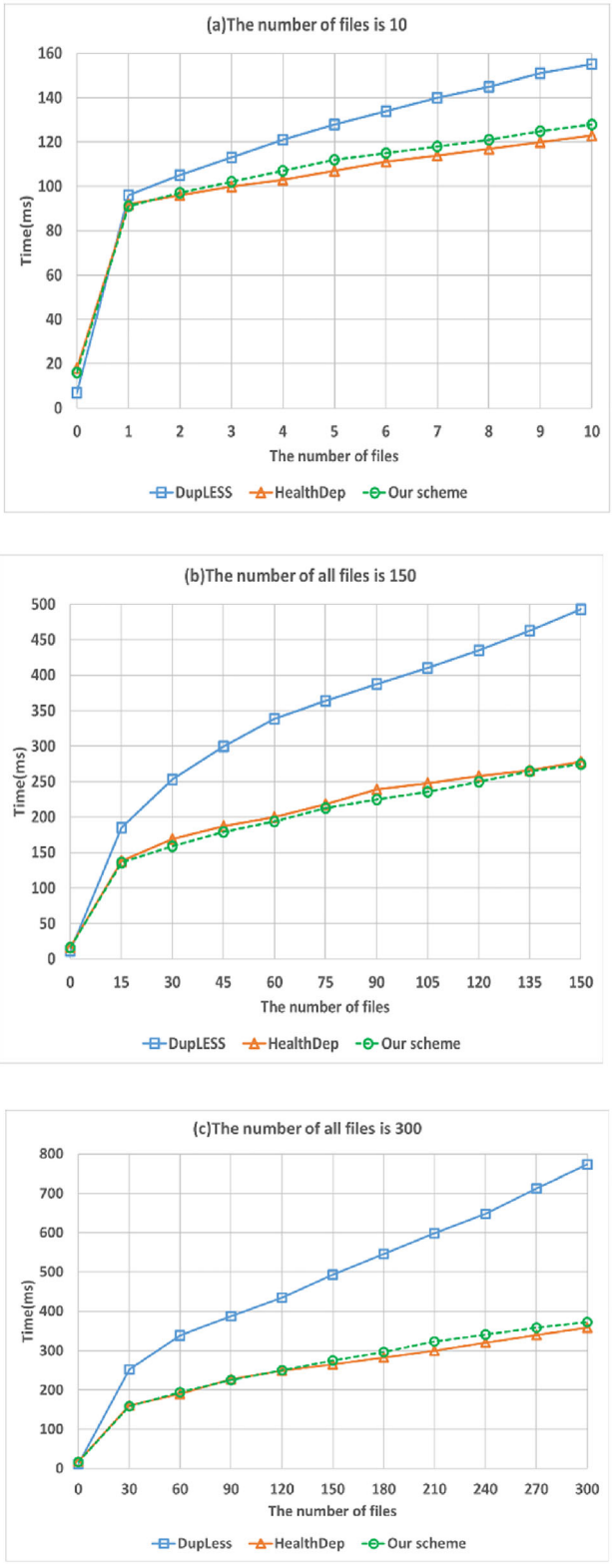

Fig. 5 Comparison of computational costs during the data upload phase

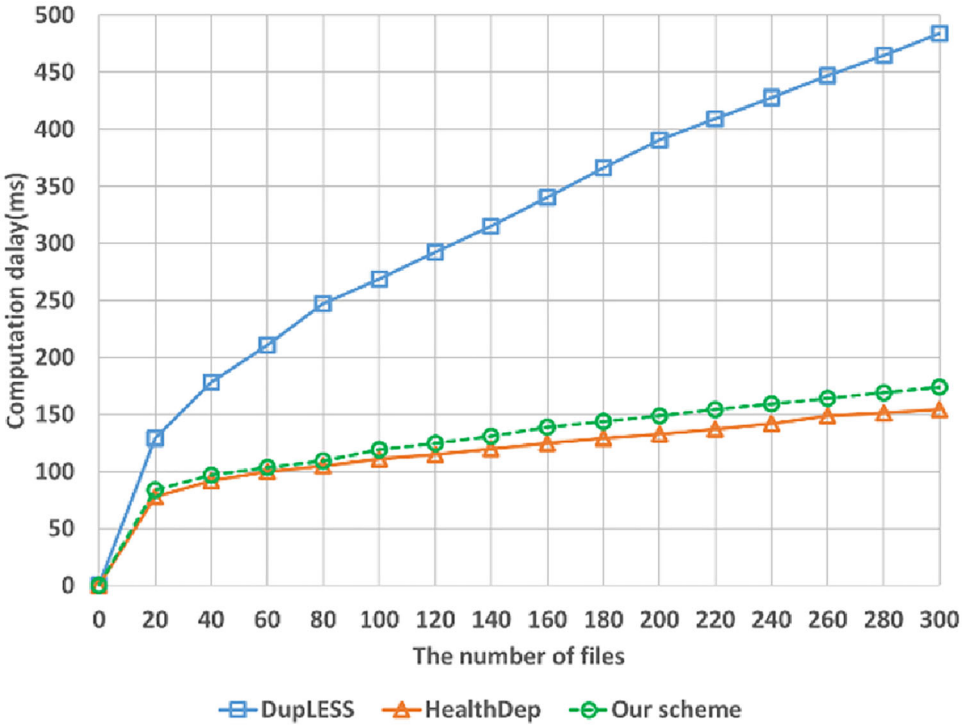

Fig. 6 Comparison of computation delay during the data upload phase

way of multiple key server interaction to ensure the security of the server-side secret. But DupLESS scheme uses only single key server. To ensure the security of the server-side secret, DupLESS scheme adopts RSA-OPRF protocol which contains some modular exponentiations of cycle multiplicative group. Compared with Hash algorithm adopted by HealthDep scheme and our scheme, RSA-OPRF protocol consumes more time. Therefore, the computational costs of DupLESS scheme are the highest. In Fig. 5b, we can see that the computational time of HealthDep scheme is similar to that of our scheme. In Fig. 5a and c, the computational time of our scheme is slightly longer than that of HealthDep scheme. The main reason is that our scheme takes extra time to blind the sensitive information compared with HealthDep scheme. Although the computational cost of HealthDep scheme is slightly better than that of our scheme, HealthDep scheme needs to store more data blocks. In contrast, our scheme needs reasonable computational cost while ensuring storage efficiency.

7.4 Computation delay

In the Fig. 5a and c, we observe that the computational time of our scheme is slightly higher than that of HealthDep scheme. In order to explore the influencing factors, we compare the time for users to generate MLE keys in these three schemes in Fig. 6. Note that we define the time for users to generate MLE keys as the computation delay. As shown in the Fig. 6, the computation delay of DupLESS scheme is the highest, followed by our scheme, and the lowest by HealthDep scheme. Just as we analyzed in Sect. 7.3, DupLESS scheme spends a lot of time in key generation, so the computation delay of DupLESS scheme is much higher than that of HealthDep scheme and our scheme. In addition, the main reason why the computation delay of HealthDep scheme is lower than that of our scheme is that the sensitive information is handled in different ways. In the HealthDep scheme, the hospital randomly selects a key which is used to encrypt the sensitive information. In our scheme, the hospital replaces the sensitive information with wildcards firstly, then interacts with the key servers to produce the corresponding MLE keys. Besides the time of blinding sensitive information, our scheme also needs to spend time interacting with the key servers. Although HealthDep scheme can reduce computation delay, it also reduces the deduplication efficiency and storage efficiency of cloud server. In contrast, our scheme achieves low computation delay while ensuring high deduplication efficiency.

In conclusion, compared with schemes [21, 26], our scheme has shown good efficiency in terms of data deduplication, storage costs, computational costs and computation delay.

8 Conclusion

In this paper, we propose a secure data sharing scheme with data deduplication and sensitive information hiding. In our scheme, multiple key servers are adopted to resist brute-force attacks and single-point-of-failure attack. We replace sensitive information of electronic medical record with wildcards to ensure the privacy of the sensitive information which also improves deduplication efficiency. We analyze the characteristics of electronic medical records of the same disease and divide data blocks into different categories based on their duplicate ratios, which helps researchers choose the data according to duplicate ratios. Performance evaluation shows that our scheme is indeed efficient according to data deduplication, storage costs, computational costs and computation delay.

Acknowledgements This research is supported by the National Natural Science Foundation of China (62172245, 62102211), Major Scientific and Technological Innovation project of Shandong Province (2022CXGC020102, 2020CXGC010114), Shandong Provincial Natural Science Foundation, China (ZR2021QF018) and Key Research and Development Project of Qingdao(21-1-2-21-XX).

Author contribution ZW: contributed to the idea of the study and wrote the manuscript. WG: contributed to the idea of the study and experiments. MY: helped perform the security analysis of the study. RH: contributed to the idea of the study and polish the manuscript.

Funding Funding was provided by National Natural Science Foundation of China (Grant Nos.: 62172245, 62102211)

Data availability The data set used in this research will be available and will be found from the websites in references.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This study does not violate and does not involve moral and ethical statement.

Informed consent I have read and comprehended the journal's contents, and I agree to all of the terms and conditions stated. All authors were aware of the publication of the paper and agreed to its publication.

References

1. Reinsel, D., Gantz, J., Rydning, J.: The Digitization of the World from Edge to Core. International Data Corporation, Framingham (2018)
2. Reinsel, G., Gantz, J.: The digital universe decade-are you ready. IDC White Paper (2010)
3. Biggar, H.: Experiencing data de-duplication: Improving efficiency and reducing capacity requirements. The Enterprise Strategy Group (2007)
4. Casola, V., Castiglione, A., Choo, K.-K.R., Esposito, C.: Healthcare-related data in the cloud: challenges and opportunities. IEEE Cloud Comput. **3**(6), 10–14 (2016)
5. Gao, X., Yu, J., Chang, Y., Wang, H., Fan, J.: Checking only when it is necessary: enabling integrity auditing based on the keyword with sensitive information privacy for encrypted cloud data. IEEE Trans Dependable Secure Compu (2021). https://doi.org/10.1109/TDSC.2021.3106780
6. Sun, J., Zhu, X., Zhang, C., Fang, Y.: HCPP: cryptography based secure EHR system for patient privacy and emergency healthcare. In: 31st International Conference on Distributed Computing Systems, pp 373–382 (2011)
7. Studdert, D.M., Mello, M.M., Gawande, A.A., Gandhi, T.K., Kachalia, A., Yoon, C., Puopolo, A.L., Brennan, T.A.: Claims, errors, and compensation payments in medical malpractice litigation. N. Engl. J. Med. **354**(19), 2024–2033 (2006)
8. List of antibiotics (2018). https://en.wikipedia.org/wiki/List_of_antibiotics.
9. Meyer, D.T., Bolosky, W.J.: A study of practical deduplication. ACM Trans. Storage **7**(4), 1–20 (2012)
10. Liu, Y., Yu, J., Fan, J., Vijayakumar, P., Chang, V.: Achieving privacy-preserving DSSE for intelligent IoT healthcare system. IEEE Trans. Industr. Inf. **18**(3), 2010–2020 (2021)
11. Ge, X., Yu, J., Hao, R., Lv, H.: Verifiable keyword search supporting sensitive information hiding for the cloud-based healthcare sharing system. IEEE Trans. Ind. Inf. **18**, 5573–5583 (2021)
12. Zhang, Y., Xu, C., Cheng, N., Shen, X.: Secure encrypted data deduplication for cloud storage against compromised key servers. In: IEEE Global Communications Conference, pp. 1–6 (2019)
13. Shakarami, A., Ghobaei-Arani, M., Shahidinejad, A., Masdari, M., Shakarami, H.: Data replication schemes in cloud computing: a survey. Clust. Comput., 2545–2579 (2021)
14. Douceur, J.R., Adya, A., Bolosky, W.J., Simon, P., Theimer, M.: Reclaiming space from duplicate files in a serverless distributed file system. In: Proceedings 22nd International Conference on Distributed Computing Systems, pp. 617–624 (2002)
15. Bellare, M., Keelveedhi, S., Ristenpart, T.: Message-locked encryption and secure deduplication. In: 32nd Annual IACR Eurocrypt International Conference on the Theory and Applications of Cryptographic Techniques, pp. 296–312, (2013)
16. Stanek, J., Sorniotti, A., Androulaki, E., Kencl, L.: A secure data deduplication scheme for cloud storage. In: 18th International Conference on Financial Cryptography and Data Security, pp. 99–118 (2014)
17. Liu, X., Sun, W., Lou, W., Pei, Q., Zhang, Y.: One-tag checker: Message-locked integrity auditing on encrypted cloud deduplication storage. In: IEEE INFOCOM 2017-IEEE Conference on Computer Communications, pp. 1–9 (2017)
18. Chen, R., Mu, Y., Yang, G., Guo, F.: BL-MLE: block-level message-locked encryption for secure large file deduplication. IEEE Trans. Inf. Forensics Secur. **10**(12), 2643–2652 (2015)
19. Anderson, P., Zhang, L.: Fast and secure laptop backups with encrypted de-duplication. In: Proceedings of the 24th International Conference on Large Installation System Administration, pp. 1–8 (2010)
20. Miao, M., Wang, J., Li, H., Chen, X.: Secure multi-server-aided data deduplication in cloud computing. Pervasive Mob. Comput. **24**, 129–137 (2015)
21. Bellare, M., Keelveedhi, S., Ristenpart, T.: DupLESS: Server-aided encryption for deduplicated storage. In: 22nd USENIX Security Symposium, pp. 179–194 (2013)
22. Shin, Y., Koo, D., Yun, J., Hur, J.: Decentralized server-aided encryption for secure deduplication in cloud storage. IEEE Trans. Serv. Comput. **13**(6), 1021–1033 (2020)

23. Yan, Z., Zhang, L., Wenxiu, D., Zheng, Q.: Heterogeneous data storage management with deduplication in cloud computing. IEEE Trans. Big Data **5**(3), 393–407 (2017)
24. Hur, J., Koo, D., Shin, Y., Kang, K.: Secure data deduplication with dynamic ownership management in cloud storage. IEEE Trans. Knowl. Data Eng. **28**(11), 3113–3125 (2016)
25. Li, J., Li, J., Xie, D., Cai, Z.: Secure auditing and deduplicating data in cloud. IEEE Trans. Comput. **65**(8), 2386–2396 (2015)
26. Zhang, Y., Xu, C., Li, H., Yang, K., Zhou, J., Lin, X.: Healthdep: an efficient and secure deduplication scheme for cloud-assisted ehealth systems. IEEE Trans. Ind. Inf. **14**(9), 4101–4112 (2018)
27. Duan, Y.: Distributed key generation for encrypted deduplication: Achieving the strongest privacy. In: Proceedings of the 6th Edition of the ACM Workshop on Cloud Computing Security, pp. 57–68 (2014)
28. Vo, D.L., Zhang, F., Kim, K.: A new threshold blind signature scheme from pairings. In: The 2003 Symposium on Cryptography and Information Security, pp. 699–702 (2003)
29. Jiang, S., Jiang, T., Wang, L.: Secure and efficient cloud data deduplication with ownership management. IEEE Trans. Serv. Comput. **13**(6), 1152–1165 (2020)
30. Koo, D., Hur, J.: Privacy-preserving deduplication of encrypted data with dynamic ownership management in fog computing. Futur. Gener. Comput. Syst. **78**, 739–752 (2018)
31. Li, J., Chen, X., Huang, X., Tang, S., Xiang, Y., Hassan, M.M., Alelaiwi, A.: Secure distributed deduplication systems with improved reliability. IEEE Trans. Comput. **64**(12), 3569–3579 (2015)
32. Yang, X., Lu, R., Shao, J., Tang, X., Ghorbani, A.: Achieving efficient secure deduplication with user-defined access control in cloud. In: IEEE Transactions on Dependable and Secure Computing (2020)
33. Puzio, P., Molva, R., Onen, M., Loureiro, S.: PerfectDedup: secure data deduplication. In: Data Privacy Management, and Security Assurance: 10th International Workshop, DPM 2015, and 4th International Workshop, QASA 2015, Vienna, Austria, September 21–22, 2015, pp. 150–166 (2016)
34. Li, S., Xu, C., Zhang, Y., Du, Y., Chen, K.: Blockchain-based transparent integrity auditing and encrypted deduplication for cloud storage. IEEE Trans Serv. Comput. (2022). https://doi.org/10.1109/TSC.2022.3144430
35. Shao, B., Ji, Y.: Efficient TPA-based auditing scheme for secure cloud storage. Cluster Comput., pp. 1989–2000 (2021)
36. Shen, W., Qin, J., Yu, J., Hao, R., Hu, J.: Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage. IEEE Trans. Inf. Forens. Secur. **14**(2), 331–346 (2018)
37. Ge, X., Yu, J., Zhang, H., Bai, J., Fan, J., Xiong, N.: SPPS: a search pattern privacy system for approximate shortest distance query of encrypted graphs in IIoT. IEEE Trans. Syst. Man Cybernet: Syst. **52**(1), 136–150 (2022)
38. Cong, L., Yu, J., Ge, X.: Enabling efficient privacy-preserving subgraph isomorphic query over graphs. Futur. Gener. Comput. Syst. **132**, 1–10 (2022)
39. Heart failure clinical records Data Set. (2020). http://archive.ics.uci.edu/ml/datasets/Heart+failure+clinical+records.

Zhiqiang Wang received the B.E. degree in Management from Shandong Normal University in 2020. He is pursuing a master degree in computer technology at Qingdao University. His research interests include cloud computing security and encrypted data deduplication.

Wenjing Gao received the B.E. degree in Information Security from Qingdao University in 2018. She is currently pursuing the Ph.D. degree in software engineering at Qingdao University. Her research interests include cloud computing security and secure outsourcing computation.

Ming Yang received the B.S. and M.S. degrees from the School of Information Science and Engineering, Shandong University, in 2004 and 2007, and the Ph.D. degree from the School of Electronic Engineering, Beijing University of Posts and Telecommunications, in 2010. He is currently an Assistant Professor with Qilu University of Technology (Shandong Academy of Sciences), Shandong Computer Science Center (National Supercomputer Center in Jinan), Shandong Provincial Key Laboratory of Computer Networks. His research interests include cloud computing security, big data security and network security.

Rong Hao is an associated professor of the College of Computer Science and Technology at Qingdao University. She is research interests include cloud computing security, digital signature, and network security. She has published over 50 academic papers. She is the reviewer of more than 10 international academic journals.