Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

# A secure data sharing scheme with cheating detection based on Chaum-Pedersen protocol for cloud storage*

Xin WANG[†1,2,3], Bo YANG[†‡1,3], Zhe XIA[4], Hong-xia HOU[1,3]

[1]*School of Computer Science, Shaanxi Normal University, Xi'an 710119, China*

[2]*School of Electronic Information and Artificial Intelligence, Shaanxi University of Science and Technology, Xi'an 710021, China*

[3]*State Key Laboratory of Information Security, Chinese Academy of Sciences, Beijing 100093, China*

[4]*School of Computer Science and Technology, Wuhan University of Technology, Wuhan 430070, China*

[†]E-mail: wangxin@sust.edu.cn; byang@snnu.edu.cn

Received Jan. 23, 2018; Revision accepted Mar. 6, 2019; Crosschecked June 11, 2019

**Abstract:** With the development of cloud computing technology, data can be outsourced to the cloud and conveniently shared among users. However, in many circumstances, users may have concerns about the reliability and integrity of their data. It is crucial to provide data sharing services that satisfy these security requirements. We introduce a reliable and secure data sharing scheme, using the threshold secret sharing technique and the Chaum-Pedersen zero-knowledge proof. The proposed scheme is not only effective and flexible, but also able to achieve the semantic security property. Moreover, our scheme is capable of ensuring accountability of users' decryption keys as well as cheater identification if some users behave dishonestly. The efficiency analysis shows that the proposed scheme has a better performance in terms of computational cost, compared with the related work. It is particularly suitable for application to protect users' medical insurance data over the cloud.

**Key words:** Data sharing; Chaum-Pedersen proof; Cheating detection; Cloud storage
https://doi.org/10.1631/FITEE.1800066 **CLC number:** TP309.2

## 1 Introduction

With the development of cloud computing technology, the cloud has demonstrated superiority in areas of computation and storage. Although there is much convenience provided by the cloud, security issues in the cloud have recently attracted more and more attention from both academia and industry. Considering that the cloud service can be publicly accessed by all users, the data owner will lose physical control of the data outsourced in the cloud. However, most existing research focuses on only how to guarantee the confidentiality of data, while little research considers incorporating accountability and auditing mechanisms into secure cloud storage. One approach uses the encryption method, in which the data stored in the cloud were encrypted using cryptographic primitives. Alhat et al. (2014) and Liang et al. (2014) discussed the security modes for the desirable protocols. Another approach focuses on

‡ Corresponding author

* Project supported by the National Key R&D Program of China (No. 2017YFB0802000), the National Natural Science Foundation of China (Nos. 61772326 and 61572303), the Research Fund for International Young Scientists, China (No. 61750110528), National Cryptography Development Fund for the 13th Five-Year Plan, China (No. MMJJ20170216), the Foundation of State Key Laboratory of Information Security, China (No. 2017-MS-03), the Fundamental Research Funds for the Central Universities, China (No. GK201702004), the Scientific Research Program Funded by Shaanxi Provincial Education Department, China (No. 16JK1109), the Provincial Natural Science Foundation Research Project of Shaanxi, China (No. 2017JQ6029), and the Doctoral Scientific Fund Project of Shaanxi University of Science & Technology, China (No. BJ11-12)

ORCID: Xin WANG, http://orcid.org/0000-0003-1904-7821

encryption keys. Kale and Vaidya (2016) and Shen et al. (2017) provided an efficient and secure method for key management. The former approach employs the key aggregate method from the system layer, and the latter one is based on the key agreement protocols. Moreover, Yang et al. (2015) combined cryptography with statistical analysis so that multiple paradigms of the data can be provided. Li et al. (2014) and Mohammed et al. (2014) used differential privacy to protect data security in the cloud. This method adds some noise into the data. The negative aspect is that although the overall data are still useful, the individual data are no longer accurate. Therefore, the applications of this approach have been restricted. It should be used only in circumstances where the accuracy of individual data is not required. Apart from the above works, security analysis is crucial for secure data storage in the cloud. Since the Rivest-Shamir-Adleman (RSA) algorithm has been widely used in the cloud to protect data, Yang et al. (2017) evaluated the security of data storage using RSA and introduced a parallel block algorithm using strip, cyclic, and improved strip to enhance the performance in the general number field sieve algorithm. In addition, to handle a complex large amount of data efficiently, an orthogonal tensor singular value method for higher-order data decomposition has been analyzed in cyber security (Feng et al., 2018).

To design provably secure and scalable protocols for data sharing in the cloud, fine-grained and flexible cryptography tools are required. Yu et al. (2010) first presented a data sharing method in the cloud by key-policy attribute-based encryption (ABE). To support multiple owners, Liu et al. (2013) used group signature and broadcast encryption to design an anonymous data sharing scheme. Liang et al. (2014) and Liu et al. (2014) independently presented a revocable data sharing scheme by employing revocable key-policy ABE schemes with the proxy re-encryption technique. However, the scheme introduced in Liu et al. (2014) has high computational overhead as it is designed based on a bilinear group. There exists a security flaw because the root secret key can be learnt by the cloud. Furthermore, it was pointed out that when the attributes as well as the required components are handled by ABE, the design is inefficient to be used in real-world applications (Liu et al., 2014). Xu et al. (2018) combined the complete subtree method and time tag to frequently update user membership for large user groups. Moreover, a secure, private, and scalable data sharing protocol based on ciphertext-policy style was presented in Dong et al. (2014).

Despite its great convenience and benefits, data storage in the cloud faces many new security challenges. For example, some special requirements in practical applications have been overlooked. To illustrate this issue, we consider the following scenario (Fig. 1). As a special field, medical treatment needs to access patients' personal health information. With a wide deployment of medical treatment through the Internet, the problem of patient privacy is particularly prominent in saving medical costs, improving patient experience, and optimizing resource allocation. Meanwhile, to realize information exchange between social insurance centers and designated medical institutions, as well as the exchange of clinical test results, electronic medical records, financial information, and other data in different information systems between medical institutions, it is necessary to ensure that these exchanges are carried out without exposing patients' private personal information. If the patients' electronic medical records, health records, consultation information, image data, and financial information are improperly contacted, stored, intercepted, and tampered with in the process of information transmission, it will cause serious information security problems. In particular, sometimes the disclosure of health information or financial information of important persons not only violates individual privacy and enterprises' interests, but also has a major impact on public confidence, or even influences international situations. Therefore, how to enhance the ability to protect the privacy and security of patients' health data has become an important issue of medical care through the Internet. In the scenario of cloud-based medical care, a patient (data owner) stores the above-mentioned important personal information, such as electronic medical records, health records, consultation information, and financial information, in ciphertext in the cloud, separates the access rights of the file into multiple copies, and assigns the rights to different types of groups, such as family group, friend group, medical staff group, and financial information management group, where each group is composed of multiple users. When a patient

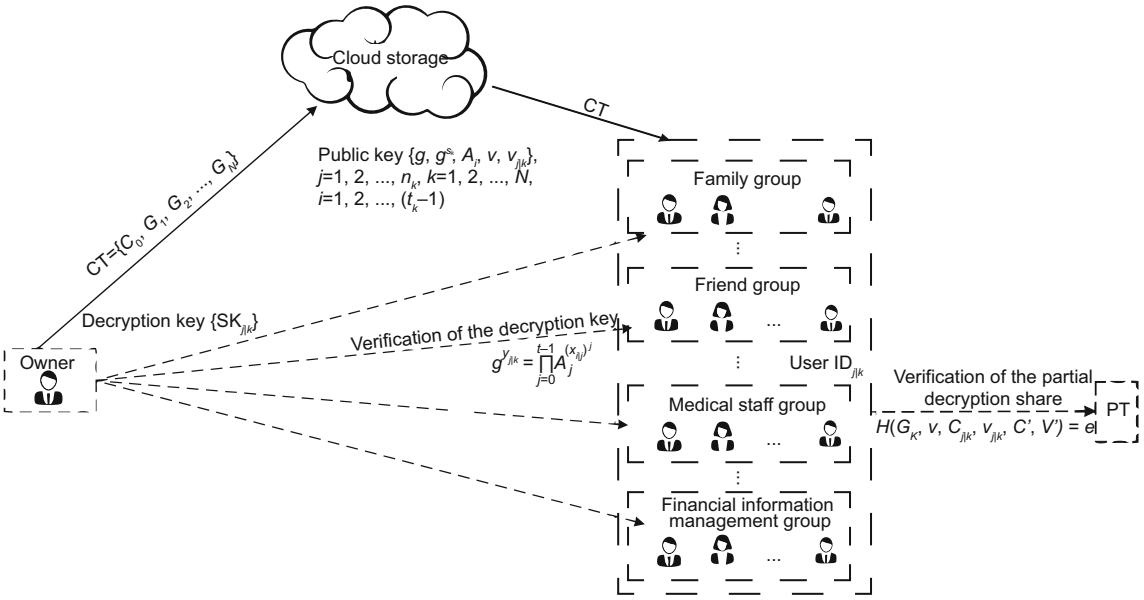


**Fig. 1 Framework for privacy-preserving data sharing service in the cloud**

(data owner) encounters an emergency or accidental death such that the evidence of the information mentioned above needs to be extracted, the information can be recovered by a group of entities even if the patient cannot provide it. In this process, to maintain the fairness of information extraction, it is crucial to allow the access rights to be invalidate for a few users, prevent deceivers from violating personal interests, and prevent dishonest users from providing false rights.

In Fig. 1, since the medical insurance information is sensitive, to protect patients' private personal data, the data owner authorizes who is allowed to access the data before they are outsourced to the cloud. Therefore, those users with the access right are carefully selected and authorized by the owner. Specifically, for the benifit of the valid users and the stability of the whole system, these users are not easily or frequently revoked. To guarantee fairness in data extraction, the master secret is divided into several parts and assigned to several user groups so that the decryption can be supervised under a scientific control mechanism. Although ABE is a fine-grained access control mechanism, the authorized users are determined by their attributes. However, the management of these users using the attributes will become intricate and complicated. To simplify the management of users, we choose to use the secret sharing technique. To ensure that both the data owner and the users perform correctly according to the protocol and to achieve supervision between these users, we apply symmetric encryption and verifiable secret sharing. Although our method is less efficient than the ones that use differential privacy, the individual data in our proposed scheme remain accurate, so the data can be analyzed directly. Hence, it is more versatile and it has potential to find more applications. When decrypting the ciphertext in the above scenario, the users should reveal their partial decryption keys. So, the correctness of the partial decryption keys is crucial for the decryption process. In other words, the identification of fake keys is very important for decryption. In existing works, the Reed-Solomon (RS) coding technique (Obana and Tsuchida, 2014; Hoshino and Obana, 2016) is often used to identify dishonest users. However, RS decoding needs to assume that the number of malicious participants is smaller than one-third of all participants, and this method does not need to rely on computational assumptions. Instead, we use the non-interactive Chaum-Pedersen zero-knowledge proof to identify the dishonest users who have presented false keys in the reconstruction phase. The advantage of our method is that it can identify every malicious participant.

In this study, we propose a flexible and semantic security data sharing scheme for the cloud environment. This scheme provides the following benefits

with respect to both security and efficiency: (1) The cloud server can assist record search using data file tags, but it cannot learn any meaningful information about owner's data or owner's personal sensitive information. (2) The users who can access the data file are authorized by the data owner, and they can verify the decryption keys sent by the owner. Even if some partial decryption keys from the authorized users are incorrect, the system can still function properly without affecting the reliability of data. (3) Dishonest users who present fake decryption keys can be identified in advance, without leaking the decryption keys for the honest users. Hence, the ciphertext can be safely and correctly decrypted under the supervision of these groups of users.

## 2 Preliminaries

### 2.1 Secret-sharing schemes

Secret-sharing schemes (SSSs) (Shamir, 1979) divide a secret into multiple parts. Each part is called a secret share and the secret can be recovered when accumulating the required number of these secret shares. Linear $(t, n)$ secret sharing is an important primitive in threshold cryptography (Shamir, 1979).

To share the secret $s \in Z_p$, the dealer $\mathcal{D}$ randomly selects a polynomial $f(x) = a_0 + a_1 x + \cdots + a_{t-1}x^{t-1}$ over $Z_p$ of degree at most $t-1$, where $a_0 = s$. Then $\mathcal{D}$ sends the shares $y_i = f(x_i)$ $(i = 1, 2, \ldots, n)$ to each shareholder through private channels, where $x$ is the public value associated with the shareholders.

To reconstruct the secret, any subset $\mathcal{A}$ of these shareholders can reconstruct the secret using polynomial interpolation, if $|\mathcal{A}| \geq t$, $s = \sum\limits_{i \in \mathcal{A}} y_i \prod\limits_{\substack{j \in \mathcal{A} \\ j \neq i}} \frac{x_j}{x_j - x_i}$. For simplicity, in the following content, we denote $L_i(x) = \prod\limits_{\substack{j=1 \\ j \neq i}}^{k} \frac{x - x_j}{x_i - x_j}$.

### 2.2 Threshold encryption cryptosystem

Threshold secret sharing is used to design the encryption scheme in our protocol. A threshold encryption cryptosystem consists of the following five algorithms:

1. The key generation algorithm (KeyGen) takes as input a security parameter $\kappa$, the number of the decryption parties $n$ $(n \geq 1)$, the threshold number $t$ $(1 \leq t \leq n)$, and a random string $x$. It outputs a public key pk, a set of secret key shares $\{y_1, y_2, \ldots, y_n\}$, and a number of verification keys $\{v_1, v_2, \ldots, v_n\}$.

2. The encryption algorithm (Enc) takes as input the public key pk, a random string $x$, and a plaintext $\mathcal{M}$. It outputs a ciphertext CT.

3. The partial decryption algorithm (PartDec) takes as input the public key pk, a ciphertext CT, an index $i$ $(1 \leq i \leq n)$, and the corresponding secret key share $y_i$. It outputs a decryption share $c_i$ and a proof $p_i$ that proves the validity of partial decryption.

4. The verification algorithm (Veri) takes as input a ciphertext CT, an index $i$ $(1 \leq i \leq n)$, the verification keys $\{v_1, v_2, \ldots, v_n\}$, the decryption share $c_i$, and its proof $p_i$. It outputs "1" if the proof is valid, and otherwise stops.

5. The combing algorithm (Comb) takes as input the public key pk and any subset of $t$ valid decryption shares. It outputs the plaintext $\mathcal{M}$.

### 2.3 Bloom filter

To protect the owner's personal information, we use the Bloom filter to hide the information, including home address, email address, job, and age. The Bloom filter is a kind of random data storage structure. It consists of multiple functions $\mathrm{BF}(x) = (\mathrm{bh}_1(x), \mathrm{bh}_2(x), \ldots, \mathrm{bh}_k(x))$. The Bloom filter is used to hide the value of the attribute and the partial information of the attribute (Lai et al., 2012). In the proposed scheme, the Bloom filter is used to anonymously store the data file and the data file is performed as the verification for the searched tags by verifying the output $(\mathrm{bh}_1(x), \mathrm{bh}_2(x), \ldots, \mathrm{bh}_k(x))$ to match the input $x$. Denote the owner's data file as $\mathrm{Value}_{\mathrm{owner}} = \{\mathrm{HomeAdd}_{\mathrm{owner}}, \mathrm{EmailAdd}_{\mathrm{owner}}, \mathrm{job}_{\mathrm{owner}}, \mathrm{age}_{\mathrm{owner}}\}$, and $\mathrm{Tag}_{\mathrm{owner}} = \mathrm{Hash}(\mathrm{Value}_{\mathrm{owner}})$, where the Hash function is a public parameter. The owner's data file after being processed by the Bloom filter is $\mathrm{BF}_{\mathrm{owner}} = \mathrm{BF}(\mathrm{Tag}_{\mathrm{owner}}) = (\mathrm{bh}_1(\mathrm{Tag}_{\mathrm{owner}}), \mathrm{bh}_2(\mathrm{Tag}_{\mathrm{owner}}), \ldots, \mathrm{bh}_k(\mathrm{Tag}_{\mathrm{owner}}))$.

The Bloom filter has the very attractive characteristic of concise space and convenient query (Bloom, 1970) .

### 2.4 Proof of equality of discrete logarithms

The Chaum-Pedersen proof protocol can be used to prove the equality of discrete logarithms (Chaum and Pedersen, 1992). Let $p$ and $q$ be two large primes such that $q\,|(p-1)$. We denote $G_q$ as the subgroup of $Z_p^*$ with order $q$. Let $g$ and $h$ be two generators of $G_q$. We can prove that the values $y \equiv g^x \pmod p$ and $t \equiv h^x \pmod p$ have the same exponent value $x$ without revealing it. The proof works as follows:

1. The prover $\mathcal{P}$ randomly chooses a value $r \in Z_q$, and then sends $U \equiv g^r \pmod p$ and $V \equiv h^r \pmod p$ to the verifier $\mathcal{V}$.

2. $\mathcal{V}$ sends a random challenge $e \in Z_q$ back to $\mathcal{P}$.

3. $\mathcal{P}$ computes $z = r + xe \bmod p$ and sends $z$ to $\mathcal{V}$.

4. $\mathcal{V}$ accepts the proof if $g^z \equiv Uy^e \pmod p$ and $h^z \equiv Vt^e \pmod p$; otherwise, $\mathcal{V}$ rejects the proof.

The correctness of the above protocol is obvious. The above protocol can be made non-interactive using Fiat-Shamir heuristics.

Special soundness holds because for two accepting conversations with the same first move $(U, V, e_1, z_1)$ and $(U, V, e_2, z_2)$, where $e_1 \neq e_2$, the witness $x$ that satisfies $y \equiv g^x \pmod p$ and $t \equiv h^x \pmod p$ can be extracted as $x \equiv (z_1 - z_2)/(e_1 - e_2) \bmod p$. Honest verifier zero-knowledge holds because for any random values $e \in Z_q$ and $z \in Z_q$, the fabricated tuple $(g^z y^{-e}, h^z t^{-e}, e, z)$ will be an acceptable conversation, and its distribution is perfectly indistinguishable from a real proof.

## 3 System model and security definitions

### 3.1 System model

The system model involves four main participants: data owner, data consumers, cloud server, and public key generator (PKG). PKG builds and maintains the public key infrastructure. The data owner (the patient) generates his/her ciphertext and stores it in the cloud. He/She also generates the shares for the decryption key and secretly delivers the share to every authorized consumer. When the consumer receives the decryption key share, he/she first verifies the validity of his/her share and then keeps it securely. When the owner's data file is found by the arbitration institution, the consumers and the trusted third party ask for the ciphertext of data from the cloud by providing $\mathrm{Tag}_{\mathrm{owner}}$. After the verification of $\mathrm{Tag}_{\mathrm{owner}}$ by a Bloom filter, the cloud sends back the ciphertext to these consumers and the trusted arbitration institution. Before these users cooperatively decrypt the ciphertext, the trusted arbitration institution first checks the correctness of the decryption key share using the Chaum-Pedersen protocol to identify every malicious user who has provided a fake key share. Finally, the plaintext of the data is obtained through the cooperation of these honest cloud consumers. This framework for secure flexible reliable data sharing for the cloud on a medical scenario is shown in Fig. 1.

### 3.2 Security definitions

To provide a rigorous security analysis for our proposed protocol, we use the following security definitions:

**Definition 1** (Correctness) If there exist $t$ honest decryption parties, a threshold encryption cryptosystem can decrypt a ciphertext and output the correct plaintext, even in the presence of some adversary who has full control of $(t-1)$ corrupt decryption parties.

**Definition 2** (Threshold semantic security) The definition was first defined by Fouque et al. (2000), and it is an extension of the semantic security definition for non-threshold encryption. Consider the following series games:

G1: Adversary $\mathcal{A}$ chooses $(t-1)$ decryption parties to corrupt. $\mathcal{A}$ can force them to surrender their private information, and $\mathcal{A}$ has full control of their behaviors for the rest of the game.

G2: The trusted dealer $\mathcal{D}$ runs the KeyGen algorithm to generate the keys. The public key pk and all verification keys $\{v_1, v_2, \ldots, v_n\}$ are broadcasted, and each decryption party receives his/her secret share. Adversary $\mathcal{A}$ learns the secret key shares held by the corrupted parties.

G3: Adversary $\mathcal{A}$ has access to a partial decryption oracle. For example, $\mathcal{A}$ can encrypt a message $m$ and input its ciphertext CT into the oracle. Then the oracle returns $n$ decryption shares of CT, along with proofs of their validity. $\mathcal{A}$ can use this oracle as many times as he/she likes.

G4: Adversary $\mathcal{A}$ issues two messages $m_0$ and $m_1$ in the message space, and sends them to an

encryption oracle. This oracle randomly selects a bit $b$, encrypts the message $m_b$, and returns its ciphertext CT to $\mathcal{A}$.

G5: Again, adversary $\mathcal{A}$ uses the partial decryption oracle as many times as he/she likes. The requirement is that $\mathcal{A}$ cannot use CT to query the partial decryption oracle.

G6: Adversary $\mathcal{A}$ outputs a bit $b'$.

The adversary's advantage is defined to be the absolute difference between 1/2 and the probability that $b = b'$. A threshold encryption is said to be threshold semantically secure if for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$, his/her advantage of running the game is negligible.

## 4 The proposed scheme

We propose an effective, reliable, and integrated data sharing scheme about Internet medical care to ensure semantic security and effective usage of owner's data on cloud storage. The patient's sensitive information (such as name, age, job, and home address, unrelated to the important data) is hidden by a Bloom filter and the important data files (such as electronic medical records, health records, consultation information, and financial information) are encrypted and stored in the cloud.

To achieve the secret information decrypted under mutual supervision between groups, decryption key sharing is used to distribute the right in cloud decryption users. In detail, these authorized users are divided into several groups, such as family group, friend group, medical staff group, and financial information management group, and they are enrolled with the protocol by their identity.

Each data file is determined by the set of these group secrets, and for every group $U_k$ there has been appointed one secret decryption key $s_k$, and then the data file can be successfully decrypted when these group secrets are correctly recovered. Each user is allocated a decryption key share $\mathrm{SK}_{j|k}$ of the group secret key $s_k$ by employing Shamir's threshold secret sharing.

### 4.1 System initialization

1. The public key generator chooses a group $\mathcal{G}_1$ of prime order $p$, an independent generator $g$ of $\mathcal{G}_1$, and a collision-resistant hash function $H$.

2. Then it takes a grouping function $\varphi(\cdot)$ and divides these users $U$ into $N$ different groups, such as doctors and nurses, relatives and friends, and legal officers, by user's identity, which are denoted as $U_1, U_2, \ldots, U_N$ satisfying $U = U_1 \bigcup U_2 \bigcup \ldots \bigcup U_N$. Here, $U$ denotes the set of users who want to share an owner's data file.

Suppose the user ID is partitioned into $U_{\varphi(\mathrm{ID})}$, where $\varphi(\cdot)$ is defined as $\varphi(\mathrm{ID}) : \mathrm{ID} \mapsto \{1, 2, \ldots, N\}$; i.e., user ID is assigned to one of group 1 to $N$. If $\varphi(\mathrm{ID}) = k$, where $k \in \{1, 2, \ldots, N\}$, then user group $U_{\varphi(\mathrm{ID})}$ is also denoted as $U_k$ for short, namely the group ID.

3. $U_k$ is a partitioned group and the number of the users in group $U_k$ is $n_k$. The data file owner selects a random $s_k$ and random polynomial $f_k(x) = a_{k,0} + a_{k,1}x + \ldots + a_{t_{k-1},1}x^{t_k-1}$ over $Z_p$ of degree at most $t_k - 1$, where $a_{k,0} = s_k$ for every group $U_k$ for $k = 1, 2, \ldots, N$.

### 4.2 Key generation

User $\mathrm{ID}_i$ is a user of the universal user set, which is partitioned into group $U_k$ by grouping function $\varphi(\cdot)$, where $|U_k| = n_k$. To sign the user $\mathrm{ID}_i$ in group $U_k$, if user $\mathrm{ID}_i$ is the $j^{\mathrm{th}}$ in group $U_k$, then it is denoted as $\mathrm{ID}_{i \to j|U_k}$, $\mathrm{ID}_{j|k}$ for short.

The data owner computes $y_{j|k} = f_k(x_{j|k})$ using group polynomial $f_k(x)$ for each user $\mathrm{ID}_{j|k}$ in group $U_k$, where $j = 1, 2, \ldots, n_k$. Here, $\{x_{1|k}, x_{2|k}, \ldots, x_{n_k|k}\}$ are public values associated with $\{\mathrm{ID}_{j|k}\}$, where $j = 1, 2, \ldots, n_k$, $k = 1, 2, \ldots, N$.

The data owner first computes $A_{i|k} = g^{a_{k,i}}$, $i = 0, 1, \ldots, t_k - 1$. We denote the verification key $v = g$. Now the other verification keys are calculated as $v_{j|k} = v^{y_{j|k}}$, $j = 1, 2, \ldots, n_k$. The values $(g, g^{s_k}, A_{i|k}, v, v_{j|k})$ are public, where $i = 1, 2, \ldots, t_k - 1$, $j = 1, 2, \ldots, n_k$, and $k = 1, 2, \ldots, N$.

Then, the data owner sends the secret key shares $\{\mathrm{SK}_{j|k}\} = \{y_{j|k}\}$ to the corresponding decryption parties $\{\mathrm{ID}_{j|k}\}$ by a private channel, where $j = 1, 2, \ldots, n_k$.

After user $\mathrm{ID}_{j|k}$ receives the secret share $\mathrm{SK}_{j|k}$, he/she first verifies whether his/her received share key $\mathrm{SK}_{j|k}$ is valid as $g^{y_{j|k}} = \prod_{j=0}^{t-1} A_j^{(x_{i|k})^j}$. After secret share $\mathrm{SK}_{j|k}$ is verified, the user takes $\mathrm{SK}_{j|k}$ as his/her secret share.

### 4.3 Data file generation

$\mathcal{M}$ is the data to be encrypted, the data owner takes a random $r_k \in Z_p$ and random exponents $\{s_k\}$, and the encrypted data file is published as $\mathrm{CP} = \{G_1, G_2, \ldots, G_k, \ldots, G_N, C_0\}$, where $G_k = g^{r_k}$, $C_0 = \mathcal{M} \cdot g^{\sum_{k=1}^{N} r_k \cdot s_k}$, $k = 1, 2, \ldots, N$.

If we denote the owner's information as $(\mathrm{Value}_{\mathrm{owner}})$, then the tag of the file is $\mathrm{Tag}_{\mathrm{owner}} = H(\mathrm{Value}_{\mathrm{owner}})$. Then the tag of retrieval and matching can be constructed as $\mathrm{BF}_{\mathrm{owner}} = \mathrm{BF}(\mathrm{Tag}_{\mathrm{owner}})$ by a Bloom filter.

The owner uploads his/her data file CP anonymously to the cloud server. Each stored data file is a format as $\mathrm{ID}||\mathrm{Tag}_{\mathrm{owner}}||\mathrm{CP}$.

### 4.4 Partial decryption algorithm

$A_k$ is an authority set of group $U_k$ and $\{A_k \,|A_k \subseteq U_k\,, k \in \{1, 2, \ldots, N\}\}$ are the union of the $N$ authority sets for $N$ groups.

These authorized users of the $N$ sets $A_k$ compute $\mathrm{Tag}_{\mathrm{owner}} = H(\mathrm{Value}_{\mathrm{owner}})$ of the data file that they want to decrypt and then send it to the cloud server.

The cloud server receives $\mathrm{Tag}_{\mathrm{owner}}$ and verifies $\mathrm{BF}_{\mathrm{owner}} = \mathrm{BF}(\mathrm{Tag}_{\mathrm{owner}})$, where $\mathrm{Tag}_{\mathrm{owner}}$ is provided by these users. If it is satisfied, the ciphertext CT is sent back to the users.

Given the ciphertext CT, each decryption user partially decrypts it. The decryption user $\mathrm{ID}_{j|k}$ uses his/her secret share key $y_{j|k}$ to compute the partial decryption share $C_{j|k} = G_k^{y_{j|k}}$. He/She also generates a non-interactive proof $p_{j|k}$ to prove that $C_{j|k}$ and $v_{j|k}$ have been raised to the same power $y_{j|k}$.

### 4.5 Decryption of data file

Authorized users $A_k$ receive the corresponding data file $G_1, G_2, \ldots, G_N, C_0$, which are sent by the cloud server. All the authority users from those authority sets $A_k$ check the equation $e = H\left(G_k, v, C_{j|k}\,, v_{j|k}\,, G_k^z \cdot C_{j|k}^{-e}, v^z \cdot v_{j|k}^{-e}\right)$ using the verification keys $\{v_{1|k}\,, v_{2|k}, \ldots, v_{n_k|k}\}$. If the proof $p_{j|k}$ is valid, the partial decryption share $C_{j|k} = G_k^{y_{j|k}}$ is a correct decryption share of $G_k$.

Then if there is no dishonest shareholders in all of the authority users, these authority users recover the data $\mathcal{M}$ using the partial decryption share $C_{j|k} = G_k^{y_{j|k}}$ as

$$\begin{aligned}\mathcal{M} &= \frac{C_0}{\prod_{k=1}^{N}\left(\prod_{j=1}^{t_k-1}\left(C_{j|k}\right)^{L_{j|k}}\right)} = \frac{C_0}{\prod_{k=1}^{N}\left(g^{r_k \cdot y_{j|k} \cdot L_{j|k}}\right)} \\ &= \frac{C_0}{\prod_{k=1}^{N}\left(g^{r_k \cdot s_k}\right)} = \frac{C_0}{g^{\sum_{k=1}^{N} r_k \cdot s_k}}. \end{aligned} \tag{1}$$

In addition, the decryption can be outsourced to the cloud. In this situation, these authorized users first generate the transformation key $\mathrm{TKCloud}_k$ in groups for the cloud before they outsource the decryption and then obtain group decryption $\mathrm{GK}_k$ $(k = 1, 2, \ldots, N)$. To generate $\mathrm{TKCloud}_k$ and $\mathrm{GK}_k$, the authorized user group $A_k$ chooses a random value $z_k \in Z_p$ and computes the transformation key $\mathrm{TKCloud}_k$ as $\{\mathrm{TKCloud}_k\} = \left\{\mathrm{SK}_{1|k}\,, \mathrm{SK}_{2|k}, \ldots, \mathrm{SK}_{n_k|k}\right\}$, and outputs the group decryption key $\mathrm{GK}_k = z_k$, where $k = 1, 2, \ldots, N$. We allow these authorized users themselves to generate the transformation key in the group. This is more flexible. Then they send the transformation key $\mathrm{TKCloud}_k$ to the cloud server for outsourced decryption. The cloud computes the following equation using $\mathrm{TKCloud}_k$:

$$\begin{aligned}\tilde{C}_0 &= \prod_{j=1}^{t_k-1}\left(\mathrm{TKCloud}_{j|k}\right)^{L_{j|k}} \\ &= \prod_{j=1}^{t_k-1}\left(g^{r_k \cdot y_{j|k}}\right)^{\frac{1}{z_k} \cdot L_{j|k}} \\ &= g^{r_k \cdot s_k \cdot \frac{1}{z_k}}, \end{aligned} \tag{2}$$

and then sends $\tilde{C}_0$ to the user groups $A_k$. After receiving $\tilde{C}_0$, all the user groups $A_k$'s compute message $\mathcal{M}$ as $C_0/\tilde{C}_0 = C_0/\prod_{k=1}^{N}\left(g^{r_k \cdot s_k \cdot \frac{1}{z_k}}\right)^{z_k} = \mathcal{M}$.

## 5 Security analysis

### 5.1 Provable security

In the proposed scheme, we use symmetric encryption to hide the message data and use secret sharing based on the threshold ElGamal scheme to share the session key within users in every group. When the security of threshold secret sharing reduces to the security of the ElGamal scheme, the modification does not disclose any information from

the ciphertext, and the proposed scheme is semantically secure. The detailed proof of threshold secret sharing is given as follows:

1. Correctness

We first show that the above protocol achieves the correctness property. If there exist $t_k$ honest decryption parties for each group $U_k$, the correct plaintext will be recovered even in the presence of $(t_k-1)$ corrupt decryption parties. If these $t_k$ decryption users are honest, there will exist at least $t_k$ valid decryption shares. Then the blind factor of plaintext $\mathcal{M}$ is recovered for

$$\begin{aligned}&\prod_{k=1}^{N}\left(\prod_{j=1}^{t_k-1}\left(C_{j|k}\right)^{L_{j|k}}\right)\\&=\prod_{k=1}^{N}\left(\prod_{j=1}^{t_k-1}\left(G_k^{y_{j|k}}\right)^{L_{j|k}}\right)=\prod_{k=1}^{N}\left(\prod_{j=1}^{t_k-1}\left(g^{r_k\cdot y_{j|k}}\right)^{L_{j|k}}\right)\\&=\prod_{k=1}^{N}\left(g^{\sum\limits_{j-1}^{t_k-1}r_k\cdot y_{j|k}\cdot L_{j|k}}\right)=\prod_{k=1}^{N}\left(g^{r_k\cdot s_k}\right)=g^{\sum\limits_{k=1}^{N}r_k\cdot s_k}.\end{aligned}\tag{3}$$

Hence, the combing algorithm will return the correct plaintext $\mathcal{M}$.

2. Threshold semantic security

We use reduction to show that the proposed protocol achieves threshold semantic security. Assume that there exists a PPT adversary $\mathcal{A}$ who can break the threshold semantic security of the proposed protocol with some non-negligible probability. Then we prove that using $\mathcal{A}$ as a subroutine, an attacker $\mathcal{B}$ can be constructed in polynomial time that breaks the semantic security of the original ElGamal encryption. To invoke $\mathcal{A}$ as a subroutine, attacker $\mathcal{B}$ must simulate all information that $\mathcal{A}$ views in the threshold protocol. $\mathcal{A}$ should not distinguish between a simulated conversation and a real run of the protocol.

In the protocol the secret key to encrypt the data $\mathcal{M}$ is divided into the sum of some shares and each share is associated to every group. This division is information security. Then we need to prove only the semantic security for every group $U_k$. Without loss of generality, we denote $U_k$ as $U$ in the following proof if there is no ambiguity.

**Theorem 1** In the random oracle model, the proposed scheme satisfies the semantical security of the threshold scheme against static adversaries under the condition that the original ElGamal scheme is semantically secure.

To break semantic security of the original ElGamal encryption scheme, attacker $\mathcal{B}$ runs a game with the challenger. First, $\mathcal{B}$ is given the public key $(g, g^s)$, and $g$ is a generator of group $\mathcal{G}_1$ where $s \in G_1$ is random. Then $\mathcal{B}$ chooses two messages $m_0$ and $m_1$ from the plaintext space, and sends them to the challenger who then randomly chooses a bit $b$ and returns the encryption of $m_b$ to $\mathcal{B}$. $\mathcal{B}$ guesses which message has been encrypted. Now, we show in the following that to invoke $\mathcal{A}$ as a subroutine, $\mathcal{B}$ can simulate $\mathcal{A}$'s view in a series of games G1–G6.

G1: Adversary $\mathcal{A}$ chooses $(t-1)$ decryption parties for the group to corrupt. Without loss of generality, we denote these parties as $P_1, P_2, \ldots, P_{t-1}$.

G2: $\mathcal{B}$ randomly selects $(t-1)$ values $y_1, y_2, \ldots, y_{t-1}$, and $\mathcal{B}$ denotes $v$ as $g$. For $j = 1, 2, \ldots, t-1$, $\mathcal{B}$ computes $v_j = v^{y_j}$. The other verification keys $v_j$ $(j = t, t+1, \ldots, n)$ can be calculated as follows:

We define a $(t-1)$-degree polynomial $f(x) = s + a_1 x + \ldots + a_{t-1}x^{t-1}$, and then have

$$\begin{cases}y_1 = s + a_1x_1 + a_2x_1^2 + \ldots + a_{t-1}x_1^{t-1},\\ y_2 = s + a_1x_2 + a_2x_2^2 + \ldots + a_{t-1}x_2^{t-1},\\ \quad\vdots\\ y_{t-1}{=}s + a_1x_{t-1}{+}a_2x_{t-1}^2{+}\ldots{+}a_{t-1}x_{t-1}^{t-1}.\end{cases}\tag{4}$$

We represent Eq. (4) as

$$\begin{bmatrix}s\\ y_1\\ \vdots\\ y_{t-1}\end{bmatrix}=\begin{bmatrix}1 & 0 & \cdots & 0\\ 1 & x_1 & \cdots & x_1^{t-1}\\ \vdots & \vdots & & \vdots\\ 1 & x_{t-1} & \cdots & x_{t-1}^{t-1}\end{bmatrix}\cdot\begin{bmatrix}s\\ a_1\\ \vdots\\ a_{t-1}\end{bmatrix}.\tag{5}$$

From Eq. (5), it can be seen that matrix

$$\begin{bmatrix}x_1 & \cdots & x_1^{t-1}\\ \vdots & & \vdots\\ x_{t-1} & \cdots & x_{t-1}^{t-1}\end{bmatrix}\tag{6}$$

is a Vandermonde matrix, and then

$$\begin{bmatrix}1 & 0 & \cdots & 0\\ 1 & x_1 & \cdots & x_1^{t-1}\\ \vdots & \vdots & & \vdots\\ 1 & x_{t-1} & \cdots & x_{t-1}^{t-1}\end{bmatrix}\tag{7}$$

has an inverse matrix. The inverse matrix is denoted

as

$$\begin{bmatrix} b_{11} & \cdots & b_{1t} \\ b_{21} & \cdots & b_{2t} \\ \vdots & & \vdots \\ b_{t1} & \cdots & b_{tt} \end{bmatrix}. \tag{8}$$

We have

$$\begin{bmatrix} b_{11} & \cdots & b_{1t} \\ b_{21} & \cdots & b_{2t} \\ \vdots & & \vdots \\ b_{t1} & \cdots & b_{tt} \end{bmatrix} \cdot \begin{bmatrix} s \\ y_1 \\ \vdots \\ y_{t-1} \end{bmatrix} = \begin{bmatrix} s \\ a_1 \\ \vdots \\ a_{t-1} \end{bmatrix}. \tag{9}$$

Then from Eq. (9) we have

$$\begin{cases} a_1 = b_{21} \cdot s + b_{22} \cdot y_1 + \ldots + b_{2t} \cdot y_{t-1}, \\ a_2 = b_{31} \cdot s + b_{32} \cdot y_1 + \ldots + b_{3t} \cdot y_{t-1}, \\ \quad \vdots \\ a_{t-1} = b_{t1} \cdot s + b_{t2} \cdot y_1 + \ldots + b_{tt} \cdot y_{t-1}, \end{cases} \tag{10}$$

and from Eq. (10), we have

$$\begin{cases} v^{a_1} = v^{b_{21} \cdot s} v^{b_{22} \cdot y_1} \ldots v^{b_{2t} \cdot y_{t-1}}, \\ v^{a_2} = v^{b_{31} \cdot s} v^{b_{32} \cdot y_1} \ldots v^{b_{3t} \cdot y_{t-1}}, \\ \quad \vdots \\ v^{a_{t-1}} = v^{b_{t1} \cdot s} v^{b_{t2} \cdot y_1} \ldots v^{b_{tt} \cdot y_{t-1}}. \end{cases} \tag{11}$$

Therefore, from Eq. (11) for $j = t, t+1, \ldots, n$, we can compute

$$v_j = v^{y_j} = v^{f(x_j)} = v^s v^{a_1 \cdot x_j} v^{a_2 \cdot x_j^2} \ldots v^{a_{t-1} \cdot x_j^{t-1}}. \tag{12}$$

Finally, attacker $\mathcal{B}$ sends the polynomial $f(x) = s + a_1 x + a_2 x^2 + \ldots + a_{t-1} x^{t-1}$, the public key $(g, y = g^s)$, $(t-1)$ secret key shares $\{y_1, y_2, \ldots, y_{t-1}\}$, and the verification keys $\{v_1, v_2 \ldots, v_n\}$ to adversary $\mathcal{A}$.

G3: $\mathcal{B}$ simulates the partial decryption oracle and answers $\mathcal{A}$'s decryption queries. If $\mathcal{A}$ encrypts a message $\mathcal{M}$ and asks $\mathcal{B}$ to decrypt its ciphertext $\mathrm{CT} = (G = g^r, C_0 = \mathcal{M} \cdot g^{r \cdot s})$, $\mathcal{B}$ will compute $C_j = G^{y_j}$ for $j = 1, 2, \ldots, t-1$, $G^s = C_0/\mathcal{M}$, and the other decryption shares $C_j$ can be calculated as $C_j = G^{y_j} = G^{f(x_j)} = G^s G^{a_1 \cdot x_j} \ldots G^{a_{t-1} \cdot x_j^{t-1}}$, $j = t, t+1, \ldots, n$. Next, $\mathcal{B}$ needs to generate proofs for these decryption shares. If $\mathcal{B}$ knows $y_j$, the proof $p_i$ is generated in the standard way as described in Section 4.5. Moreover, in the random oracle model, attacker $\mathcal{B}$ has full control of the hash function, and $\mathcal{B}$ can answer a hash query using any value of his/her choice, as long as he/she returns a consistent output if the same input is queried multiple times. Hence, $\mathcal{B}$ can fabricate the other proofs where he/she has no knowledge of their secret shares. This is done by defining the value of the random oracle at $H\left(G, v, C_j, v_j, G^z \cdot C_j^{-e}, v^z \cdot v_j^{-e}\right)$ to be $e$. Now, $\mathcal{B}$ returns $n$ decryption shares $(C_1, C_2, \ldots, C_n)$ along with their proofs $(p_1, p_2, \ldots, p_n)$ to $\mathcal{A}$.

G4: In this step, $\mathcal{B}$ first waits for $\mathcal{A}$ to select two messages $m_0$ and $m_1$ from the plaintext space. After receiving these two values, $\mathcal{B}$ forwards them to the challenger. The challenger then randomly selects a bit $b$, encrypts $m_b$ using the original ElGamal encryption scheme, and returns its ciphertext CT to $\mathcal{A}$. Now, $\mathcal{B}$ sends CT to $\mathcal{A}$.

G5: This step is similar to step G3. The additional requirement is that $\mathcal{A}$ is not allowed to use CT to query the partial decryption oracle.

G6: $\mathcal{A}$ outputs a bit $b'$, and $\mathcal{B}$ forwards $b'$ to the challenger. It is clear that the above simulation can be carried out in polynomial time. The remaining task is to prove that a simulated conversation is indistinguishable from a real run of the protocol. Because no view has been simulated in steps G1 and G6, and step G5 just repeats G3, we need only to show that adversary $\mathcal{A}$ cannot distinguish between the simulated view and the real view from steps G2–G4.

Indistinguishability in G2: In the step, the same $g$, $g^s$, and $f(x)$ are used. These values are randomly chosen. Hence, the secret polynomial $f(x)$ and public keys $g$ and $g^s$ exactly follow the same distribution as in the real protocol. In the simulation, the secret key shares $y_j$ are randomly distributed in $G_1$. Then these cannot be distinguished from real ones. The simulated verification keys $\{v_1, v_2 \ldots, v_n\}$ are randomly distributed in the cyclic group $\mathcal{G}_1$, and they also cannot be distinguished from real ones. Hence, the simulated view in this step is statistically indistinguishable from a real run of the protocol.

Indistinguishability in G3: In this step, the simulated decryption shares $(C_1, C_2, \ldots, C_n)$ follow the same distribution as those in the real protocol. Both are randomly distributed in the cyclic group $\mathcal{G}_1$. Moreover, the simulated proofs $(p_1, p_2, \ldots, p_n)$ follow the same distribution as those in the real protocol. Hence, the simulated view in this step is

indistinguishable.

Indistinguishability in G4: In this step, the ciphertext CT is randomly distributed in group $\mathcal{G}_1$. Hence, they are indistinguishable.

### 5.2 Security analysis

In the proposed scheme, we use encryption and the threshold secret share technique to protect the security of the data. To be more suitable for the medical scene, we deploy some other security properties.

1. Privacy of the owner's information and supervision of groups

When the data owner stores the data over the cloud to enjoy the convenience of the cloud service, it is unnecessary to expose his/her information unrelated to the data to the cloud. Then the owner can take an anonymous style when the ciphertext is uploaded to the cloud. Here we choose the Bloom filter to hide his/her information for the retrieval and matching of the data file's tag. Then the privacy of the data owner is completely protected.

In the proposed scheme, the secret is divided into each group to hold each group accountable, if and only if all $N$ groups present the correct secret shares $s_k$, and the plaintext is correctly decrypted. Any user who presents a fake share in the authorized user set would cause the group key $s_k$ error, which leads to a false decryption process.

2. Verifiability of the share key distribution

In our proposed scheme, to prevent fraud on the data owner, the cloud user can verify the correctness of the share key distributed by the owner.

Supposing that user $\mathrm{ID}_j$ accepts the secret key share $\mathrm{SK}_{j|k}$ from the data owner, there is a unique $y_{j|k}$ such that $g^{y_{j|k}} = \prod_{j=0}^{t-1} (A_j)^{(x_{i|k})^j}$. The effectiveness of this verification is obvious.

3. Cheating detection

The RS coding technique is a usual method to identify the cheater. The existing methods are not based on computational assumptions, but they need to assume that the number of malicious participants is smaller than 1/3 of the total number. Our method is based on computational assumptions through the zero-knowledge proof technology and can identify every malicious participant.

The proof is employed here which shows prevention of the cheaters changing the secret key share $\mathrm{SK}_{j|_k}$. This is used to test the validity of the secret key shares, because the Chaum-Pedersen protocol has the ability to verify the same index $y_{j|k}$ in partial share $C_{j|k}$ and verification $v_{j|k}$.

The proof works as follows: Take the collision-resistant hash function $H$. User $\mathrm{ID}_{j|k}$ randomly selects a value $\delta$, and computes $C' = C_{j|k}^{\delta}$ and $v' = v_{j|k}^{\delta}$. Then the proof $p_{j|k}$ is a pair $\left(z_{j|k}, e_{j|k}\right)$, where $z_{j|k} = y_{j|k} \cdot e_{j|k} + \delta$ and $e_{j|k} = H\left(G_k, v, C_{j|k}, v_{j|k}, C', v'\right)$. $\mathrm{ID}_{j|k}$ broadcasts the decryption share $C_{j|k}$ as well as its proof $p_{j|k}$, where $j = 1, 2, \ldots, n_k$ and $k = 1, 2, \ldots, N$.

From the above analysis, we can see that our scheme does not need to restrict the number of malicious participants. Hence, it is suitable for more practical applications.

## 6 Performance analysis

The basic operations such as file creation or deletion and user addition or revocation are processed similarly as in all these schemes. Here, the discussion is omitted.

In these data sharing schemes on ABE (Yu et al., 2010; Dong et al., 2014; Liang et al., 2014; Liu et al., 2014; Xu et al., 2018), there was a flaw in the design of Liu et al. (2014) as the root secret key is known to the cloud for the re-encryption of the original ciphertext. Liang et al. (2014) designed the composite order groups, which makes the scheme less efficient. Here we compare our scheme with some classical and efficient data sharing schemes in Yu et al. (2010), Dong et al. (2014), and Xu et al. (2018). The schemes in Yu et al. (2010) and Xu et al. (2018) are based on key-policy ABE, while the scheme in Dong et al. (2014) is based on ciphertext-policy ABE (CP-ABE). In our scheme, $t_k$ can also be seen as the attribute involved in the group $U_k$. Data confidentiality is achieved in all these schemes since the data file is stored as ciphertext for the cloud server, and the cloud servers are not able to learn the plaintext of any data file.

The decryption keys of data are not known by the cloud server in any scheme in Yu et al. (2010), Dong et al. (2014), and Xu et al. (2018), as well as in our scheme, although the proxy re-encryption key is given to the cloud in Yu et al. (2010) and Xu et al. (2018). The comparison analysis of the security properties between our scheme and the schemes in Yu et al. (2010), Dong et al. (2014), and Xu et al.

(2018) is summarized in Table 1.

### 6.1 Computation complexity

In this subsection, we compare the computation overhead of our scheme with that of Yu et al. (2010), Dong et al. (2014), and Xu et al. (2018)'s schemes, considering key generation, encryption, and decryption. For a fair comparison, we consider a general situation in which all users have participated in the decryption stage. In Yu et al. (2010), Dong et al. (2014), and Xu et al. (2018), the main computational costs involved in encryption and decryption algorithms are pairing $e(g,g)$ and scalar multiplication. In our scheme, there is no involvement in a pairing operation.

The ciphertext of the proposed scheme is $\mathrm{CP}=\left\{G_1=g^{r_1},G_2=g^{r_2},\ldots,G_N=g^{r_N},C_0=\mathcal{M}\cdot g^{\sum_{k=1}^{N} r_k\cdot s_k}\right\}$. During encryption, the data owner takes all encryption operations and needs to do $N$ scalar multiplications for $C_0$ and $G_k$, separately. Thus, the owner should take $2N$ scalar multiplications in total for encryption. Then the computation complexity of encryption is $O(N\cdot t_S)$. In the decryption phase, to recover the ciphertext, the user needs at most $|U|$ scalar multiplications to calculate $\prod_{k=1}^{N}\left(\prod_{j=1}^{t_k-1}\left(C_{j|k}\right)^{L_{j|k}}\right)=\prod_{k=1}^{N}\left(g^{r_k\cdot y_{j|k}\cdot L_{j|k}}\right)=\prod_{k=1}^{N}\left(g^{r_k\cdot s_k}\right)=g^{\sum_{k=1}^{N} r_k\cdot s_k}$. So, the computation complexity of decryption is $O(|U|\cdot t_{\mathrm{S}})$. In the key generation phase, it is only polynomial time to calculate $y_{j|k}=f_k\left(x_{j|k}\right)$ and at most $O(|U|)$ scalar multiplications to calculate public key $\left\{g,g^{s_k},A_{i|k},v,v_{j|k}\right\}$, where $i=1,2,\ldots,t_k-1$, $j=1,2,\ldots,n_k$, and $k=1,2,\ldots,N$. Thus, the computational complexity of key generation is $O(|U|\cdot t_{\mathrm{S}})$.

In Yu et al. (2010), the data owner needs to do one scalar multiplication and one pairing to calculate $\tilde{E}=M\cdot e(g,g)^{ys}$, and one scalar multiplication for $E_i=g^{t_i\cdot s}$ to generate a ciphertext. Therefore, the computation complexity of encryption is $O(|I_{\mathrm{u}}|\cdot t_{\mathrm{P}}+2|I_{\mathrm{u}}|\cdot t_{\mathrm{S}})$. To recover the ciphertext, it has to compute $e(E_i,\mathrm{sk}_i)=e(g,g)^{p_i(0)s}$ for each leaf node first. Then the blind factor $Y^s=e(g,g)^{ys}$ is recovered aggregating these pairing results in the bottom-up manner if and only if attributes $I$ satisfy access tree $T$. So, the time complexity for decryption is $O(\max(|I_{\mathrm{m}}|,|I_{\mathrm{u}}|)\cdot t_{\mathrm{P}})$. The time complexity for key generation $\mathrm{SK}=\left\{\mathrm{sk}_i\,\middle|\,\mathrm{sk}_i=g^{p_i(0)/t_i},i\in I_{\mathrm{m}}\right\}$ is $O(|I_{\mathrm{m}}|\cdot t_S)$.

In Dong et al. (2014), the data owner needs to do two pairing multiplications to calculate $C_{1,x}=e(g_1,g_1)^{\lambda_x}e(g_1,g_1)^{\alpha_{\rho(x)}r_x}$, one scalar multiplication for $C_{2,x}=g_1^{r_x}$, and two for $C_{3,x}=g_1^{\beta_{\rho(x)}r_x}g_1^{\omega_x}$. Therefore, the computation complexity of encryption is $O(2|I_{\mathrm{m}}|\cdot t_{\mathrm{P}}+3|I_{\mathrm{m}}|\cdot t_{\mathrm{S}})$. To recover the ciphertext, the user needs another $2|I_{\mathrm{m}}|$ scalar multiplications at most to calculate $\prod_x\left\{C_{1,x}\cdot e(H(\mathrm{ID}_u),C_{3,x})/e\left(\mathrm{sk}_{\rho(x),u},C_{2,x}\right)\right\}$, so the time complexity of decryption is $O(2|I_{\mathrm{m}}|\cdot t_{\mathrm{P}}+|I_{\mathrm{m}}|\cdot t_{\mathrm{S}})$. The time complexity for key generation is $O(|I_{\mathrm{u}}|\cdot t_{\mathrm{S}})$.

In Xu et al. (2018), the data owner needs to do one pairing multiplication and $(|V|+|I_{\mathrm{u}}|+3)$ scalar multiplications to calculate $\mathrm{ct}=\{S,t,C,C_1,\{C_i^{(0)}\},C^{(1)},\{C_j^{(1)}\}\}$, where ct denotes a ciphertext, $C=me(g_1,g_2)^s$, $C_1=g^s$, $\{C_i^{(0)}\}=\{T(i)^s|p(i)\in S\}$, $C^{(1)}=\{U'^S\}$, and $\{C_j^{(1)}|j\in V\}=\{U_i^S|i=1,2,\ldots,n+1\}$ with $n$ the maximum size of the attribute set used in encryption. Therefore, the computation complexity of encryption is $O(1\cdot t_{\mathrm{P}}+(|V|+|I_{\mathrm{u}}|+3)\cdot t_{\mathrm{S}})$. To recover the ciphertext, the user first needs to recover $\prod_{p(i)\in S}\left(\frac{e(K_i^{(0)},C_1)}{e(K_i^{(1)},C_i^{(0)})}\right)^{\omega_i}$ with computation complexity $O(2|I_{\mathrm{u}}|\cdot t_{\mathrm{P}}+|I_{\mathrm{u}}|\cdot t_{\mathrm{S}})$, then needs two pairings to calculate $\frac{e(K^{(0)},C_1)}{e(K^{(1)},C_t)}$, and finally needs one pairing

**Table 1 Security properties**

| Scheme | Anonymous storage | Identifiable cheater | Verification of key share | Revocation of large-scale users |
|---|---|---|---|---|
| Yu et al. (2010) | No | No | Yes | No |
| Dong et al. (2014) | No | No | No | No |
| Xu et al. (2018) | No | No | No | Yes |
| Our scheme | Yes | Yes | Yes | No |

to obtain message $m$. So, the time complexity of decryption is $O((2|I_u|+3)t_P+|I_u|\cdot t_S)$. The time complexity for key generation $\mathrm{sk}_{\mathrm{id},x}=\{K_i^{(0)}=g_2^{M_i u}T(i)^{r_i}, K_i^{(1)}=g^{r_i}\}$ is $O(2|U|\cdot t_S)$, where $i=1,2,\ldots,d$ and $d$ is the row number of the access matrix.

The computational complexities of our scheme and the schemes proposed by Yu et al. (2010) , Dong et al. (2014), and Xu et al. (2018) are given in Table 2. From the above, we can see that the computation complexity is related to the total number of the cloud users in our proposed scheme and the number of attributes that all the users have mastered. In general, every user would have several attributes, and then the complexity of our scheme is slightly lower than that of other schemes.

## 6.2 Experimental results

The evaluation was conducted through experimentally evaluating the time cost of the proposed scheme among Yu et al. (2010), Dong et al. (2014), and Xu et al. (2018) on a computer with Windows 7 Intel i5-4590S-3.00 GHz CPU and 4 GB RAM.

### 6.2.1 Overload of the key generation algorithm

The key generation algorithm is to compute the power multipication in all four schemes. To simplify the comparison, we took all cloud users in one group. Then overloads of key generation of these schemes are shown in Fig. 2. In Xu et al. (2018), the decryption keys are based on an attribute related to the time that is frequently updated. So, the design of the key is complex, and the overload of key generation will be very large. It can also be seen from Fig. 2 that the overload in our scheme is much less than that in Xu et al. (2018) and Dong et al. (2014) and a little more than that in Yu et al. (2010).

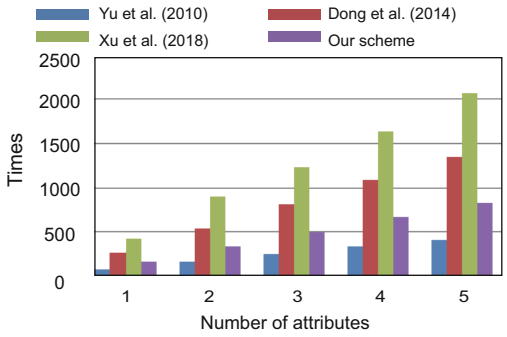


**Fig. 2 Comparison of key generation performance**

References to color refer to the online version of this figure

### 6.2.2 Overload of the encryption algorithm

The encryption costs of our scheme and other schemes with the number of attributes from 10 to 50 are given in Fig. 3. From Fig. 3, we can see that the encryption cost increases linearly with the number of attributes in the three schemes, and our scheme has less cost than the others. In our scheme, the encryption is to calculate $(G_1,G_2,\ldots,G_N,C_0)$, which is based on the number of the groups, while in Yu et al. (2010), Dong et al. (2014), and Xu et al. (2018), the calculation of ciphertext $C$ is based on attributes of the ciphertext. In addition, the ciphertexts in Xu et al. (2018) were designed by time regarding the proxy re-encryption of ciphertext by the cloud, which involves more cost in computing the ciphertext.

### 6.2.3 Overload of the decryption algorithm

In our scheme, the decryption is to compute $\prod_{k=1}^{N}\left(\prod_{j=1}^{t_k-1}\left(C_{j|k}\right)^{L_{j|k}}\right)$. The cost of decryption depends on mainly the numbers of groups and users in each group. When all users are in one group in our scheme, the overheads of decryption with the attributes of access structure up to 50 are given in Fig. 4, where the number of users in our scheme is

**Table 2 Comparison of computation complexity**

| Scheme | Encryption (data owner) | Decryption (user) | Key generation |
|---|---|---|---|
| Yu et al. (2010) | $O(\lvert I_u\rvert\cdot t_P+2\lvert I_u\rvert\cdot t_S)$ | $O(\max(\lvert I_m\rvert,\lvert I_u\rvert)\cdot t_P)$ | $O(\lvert I_m\rvert\cdot t_S)$ |
| Dong et al. (2014) | $O(2\lvert I_m\rvert\cdot t_P+3\lvert I_m\rvert\cdot t_S)$ | $O(2\lvert I_m\rvert\cdot t_P+\lvert I_m\rvert\cdot t_S)$ | $O(\lvert I_u\rvert\cdot t_S)$ |
| Xu et al. (2018) | $O(1\cdot t_P+(\lvert V\rvert+\lvert I_u\rvert+3)\cdot t_S)$ | $O((2\lvert I_u\rvert+3)t_P+\lvert I_u\rvert\cdot t_S)$ | $O(2\lvert U\rvert\cdot t_S)$ |
| Our scheme | $O(N\cdot t_S)$ | $O(\lvert U\rvert\cdot t_S)$ | $O(\lvert U\rvert\cdot t_S)$ |

$I_u$: attributes of the user; $I_m$: attributes of the access structure; $V$: size of the user identities in binary form; $U$: set of universal users; $N$: number of the partitioned groups in our scheme; $t_P$: pairing time; $t_S$: scalar multiplication time

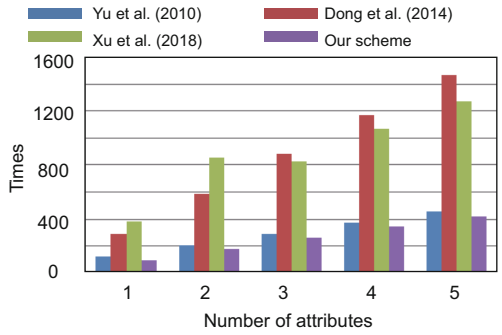


**Fig. 3 Comparison of encryption performance**

References to color refer to the online version of this figure

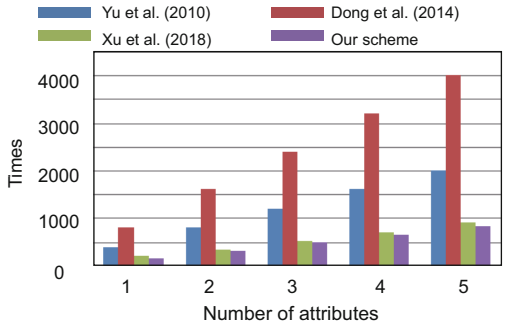


**Fig. 4 Comparison of decryption performance**

References to color refer to the online version of this figure

the same as the number of attributes in Dong et al. (2014). As only the scheme in Dong et al. (2014) is based on CP-ABE, the cost of computation to decrypt is related to the overload of attributes in the access matrix, so the overload is very large. From Fig. 4, it can be seen that the overhead in our scheme is almost the same as that in Xu et al. (2018) and less than that in Yu et al. (2010).

### 6.3 Communication cost

In these schemes, the communication cost is caused mainly by the encrypted data and key dissemination. The encrypted data sent by the data owner to the cloud are $\{G_1 = g^{r_1}, G_2 = g^{r_2}, \ldots, G_N = g^{r_N}, C_0 = \mathcal{M} \cdot g^{\sum_{k=1}^{N} r_k \cdot s_k}\}$, which require $(N+1)\log|\mathcal{G}_1|$ bits. The value $y_{j|k} = f_k\left(x_{j|k}\right)$ for every $j \in U$ requires $(N+1)\log|\mathcal{G}_1|$. Thus, the communication cost is given by $(N+1)\log|\mathcal{G}_1|+3|U|\cdot\log|\mathcal{G}_1|+$data.

The comparison of communication overhead between the proposed scheme and the other three schemes is given in Table 3.

**Table 3 Comparison of communication cost**

| Scheme | Communication cost |
|---|---|
| Yu et al. (2010) | $\lvert I\rvert + 2\log\lvert I\rvert + (\lvert I\rvert + 1)\log\lvert\mathcal{G}_1\rvert + \log\lvert\mathcal{G}_2\rvert +$ data |
| Dong et al. (2014) | $\lvert I\rvert^2 + \log\lvert I\rvert + (2\lvert I\rvert + 1)\log\lvert\mathcal{G}_1\rvert + (\lvert I\rvert + 1)\log\lvert\mathcal{G}_2\rvert +$ data |
| Xu et al. (2018) | $(2d + 3 + \lvert I\rvert + \lvert U\rvert)\log\lvert\mathcal{G}_1\rvert + \log\lvert\mathcal{G}_2\rvert +$ data |
| Our scheme | $(N+1)\log\lvert\mathcal{G}_1\rvert + 3\lvert U\rvert\cdot\log\lvert\mathcal{G}_1\rvert +$data |

## 7 Applications to complex medical scenarios on Internet

Health care data cover the life of human beings, and they contain the convergence and aggregation of various data. These data include medical record information, medical insurance information, health log, genetic, medical experiments, and scientific research data. These data thus cover an extensive range. Personal medical data should meet the requirement of personal privacy protection. Medical experimental data, scientific research data, and insurance information not only have aspects of the privacy of data subjects and medicine industry trends, but also affect national security. Therefore, in the development process and application of medical care data, it is necessary to provide a targeted compliance guarantee for data origin authentication and the type of medical data.

In the Internet-based personal health environment, how to safely manage personal health care data is a great challenge. When a person stores his/her medical insurance data in the cloud for future occasional needs (sudden death or insurance claims), it is crucial that who is allowed to access the data should be carefully considered. In this process, due to the diversity and complexity of medical data, not only the confidentiality of data, but also the actual scenario needs to be considered in the data storage process. As shown in Fig. 1, we give a protection scheme for personal health care data. For the sake of personal data security, the data owner can appoint the authorized users by himself/herself. In addition, according to the type of users, group management is carried out to efficiently and conveniently manage the data, so that data access can be carried out in a supervision mode, thus achieving a decentralized authority management mechanism. In addition, because of the need for the validity and fairness of data, any behavior that interferes with normal data

access will be completely and effectively identified to ensure the stability and normal operation of the system. Therefore, to achieve the above goals, we combine symmetric encryption and the secret sharing technique, and propose a verifiable and cheater-identifiable method on the Chaum-Pedersen protocol data sharing scheme based on threshold secret sharing. This scheme can have a fundamental impact to fulfill the above subject's medical data management in the personal health environment of Internet.

In summary, our scheme helps the patient achieve flexible and supervised control on his/her case file stored on the cloud server.

## 8 Conclusions and future work

In this paper, we have proposed a data sharing scheme over the cloud that is suitable for group co-supervision. In our scheme, the security and reliability for the data file can be adequately protected. Also, the proposed scheme achieves cheater identification without violating the honest person's rights. Compared with the existing cheating identification methods, our scheme can detect every dishonest user. The efficiency analysis indicated that the proposed scheme has a low computational cost and bandwidth usage. In future work, we will investigate how to execute efficient cloud searching, when a large number of files are stored on the cloud.

### Compliance with ethics guidelines

Xin WANG, Bo YANG, Zhe XIA, and Hong-xia HOU declare that they have no conflict of interest.

### References


Alhat RY, Kedari DB, Sangale BG, et al., 2014. Ensuring distributed accountability for data sharing in the cloud network. *Int J Eng Res Technol*, 3(2):494-501.

Bloom BH, 1970. Space/time trade-offs in hash coding with allowable errors. *Commun ACM*, 13(7):422-426. https://doi.org/10.1145/362686.362692

Chaum D, Pedersen TP, 1992. Wallet databases with observers. 12th Annual Int Cryptology Conf on Advances in Cryptology, p.89-105. https://doi.org/10.1007/3-540-48071-4_7

Dong X, Yu JD, Luo Y, et al., 2014. Achieving an effective, scalable and privacy-preserving data sharing service in cloud computing. *Comput Secur*, 42:151-164. https://doi.org/10.1016/j.cose.2013.12.002

Feng J, Yang LT, Dai GH, et al., 2018. A secure higher-order Lanczos-based orthogonal tensor SVD for big data reduction. *IEEE Trans Big Data*, in press. https://doi.org/10.1109/tbdata.2018.2803841

Fouque PA, Poupard G, Stern J, 2000. Sharing decryption in the context of voting or lotteries. Int Conf on Financial Cryptography, p.90-104. https://doi.org/10.1007/3-540-45472-1_7

Hoshino H, Obana S, 2016. Cheating detectable secret sharing scheme suitable for implementation. 4th Int Symp on Computing and Networking, p.623-628. https://10.1109/CANDAR.2016.0112

Kale P, Vaidya M, 2016. Key-aggregate cryptosystem for scalable data sharing in cloud storage. *Imper J Int Res*, 2(8):957-961.

Lai JZ, Deng RH, Li YJ, 2012. Expressive CP-ABE with partially hidden access structures. 7th ACM Symp on Information, Computer and Communications Security, p.18-19. https://doi.org/10.1145/2414456.2414465

Li HR, Xiong L, Zhang LF, et al., 2014. DPSynthesizer: differentially private data synthesizer for privacy preserving data sharing. *Proc VLDB Endowm*, 7(13):1677-1680. https://doi.org/10.14778/2733004.2733059

Liang KT, Au MH, Liu JK, et al., 2014. A DFA-based functional proxy re-encryption scheme for secure public cloud data sharing. *IEEE Trans Inform Forens Secur*, 9(10):1667-1680. https://doi.org/10.1109/tifs.2014.2346023

Liu Q, Wang GJ, Wu J, 2014. Time-based proxy re-encryption scheme for secure data sharing in a cloud environment. *Inform Sci*, 258:355-370. https://doi.org/10.1016/j.ins.2012.09.034

Liu XF, Zhang YQ, Wang BY, et al., 2013. Mona: secure multi-owner data sharing for dynamic groups in the cloud. *IEEE Trans Parall Distrib Syst*, 24(6):1182-1191. https://doi.org/10.1109/tpds.2012.331

Mohammed N, Alhadidi D, Fung BCM, et al., 2014. Secure two-party differentially private data release for vertically partitioned data. *IEEE Trans Depend Secur Comput*, 11(1):59-71. https://doi.org/10.1109/tdsc.2013.22

Obana S, Tsuchida K, 2014. Cheating detectable secret sharing schemes supporting an arbitrary finite field. In: Yoshida M, Mouri K (Eds.), Advances in Information and Computer Security. Springer, Cham, p.88-97. https://doi.org/10.1007/978-3-319-09843-2_7

Shamir A, 1979. How to share a secret. *Commun ACM*, 22(11):612-613. https://doi.org/10.1145/359168.359176

Shen J, Zhou TQ, He DB, et al., 2017. Block design-based key agreement for group data sharing in cloud computing. *IEEE Trans Depend Secur Comput*, in press. https://doi.org/10.1109/tdsc.2017.2725953

Xu SM, Yang GM, Mu Y, et al., 2018. Secure fine-grained access control and data sharing for dynamic groups in the cloud. *IEEE Trans Inform Forens Secur*, 3(8):2101-2113. https://doi.org/10.1109/tifs.2018.2810065

Yang JJ, Li JQ, Niu Y, 2015. A hybrid solution for privacy preserving medical data sharing in the cloud environment. *Fut Gener Comput Syst*, 43-44:74-86. https://doi.org/10.1016/j.future.2014.06.004

Yang LT, Huang GY, Feng J, et al., 2017. Parallel GNFS algorithm integrated with parallel block Wiedemann algorithm for RSA security in cloud computing. *Inform Sci*, 387:254-265. https://doi.org/10.1016/j.ins.2016.10.017

Yu SC, Wang C, Ren K, et al., 2010. Achieving secure, scalable, and fine-grained data access control in cloud computing. IEEE Int Conf on Computer Communications, p.1-9. https://doi.org/10.1109/INFCOM.2010.5462174