RESEARCH ARTICLE

A searchable personal health records framework with fine-grained access control in cloud-fog computing

Jin Sun, Xiaojing Wang^{*}, Shangping Wang^{ID}, Lili Ren

School of Science, Xi'an University of Technology, Xi'an, Shaanxi, China

^{*} wxjdyb@163.com

Abstract

Fog computing can extend cloud computing to the edge of the network so as to reduce latency and network congestion. However, existing encryption schemes were rarely used in fog environment, resulting in high computational and storage overhead. Aiming at the demands of local information for terminal device and the shortcomings of cloud computing framework in supporting mobile applications, by taking the hospital scene as an example, a searchable personal health records framework with fine-grained access control in cloud-fog computing is proposed. The proposed framework combines the attribute-based encryption (ABE) technology and search encryption (SE) technology to implement keyword search function and fine-grained access control ability. When keyword index and trapdoor match are successful, the cloud server provider only returns relevant search results to the user, thus achieving a more accurate search. At the same time, the scheme is multi-authority, and the key leakage problem is solved by dividing the user secret key distribution task. Moreover, in the proposed scheme, we securely outsource part of the encryption and decryption operations to the fog node. It is effective both in local resources and in resource-constrained mobile devices. Based on the decisional q -parallel bilinear Diffie-Hellman exponent (q -DBDHE) assumption and decisional bilinear Diffie-Hellman (DBDH) assumption, our scheme is proven to be secure. Simulation experiments show that our scheme is efficient in the cloud-fog environment.

OPEN ACCESS

Citation: Sun J, Wang X, Wang S, Ren L (2018) A searchable personal health records framework with fine-grained access control in cloud-fog computing. PLoS ONE 13(11): e0207543. <https://doi.org/10.1371/journal.pone.0207543>

Editor: Rashid Mehmood, King Abdulaziz University, SAUDI ARABIA

Received: July 21, 2018

Accepted: November 1, 2018

Published: November 29, 2018

Copyright: © 2018 Sun et al. This is an open access article distributed under the terms of the [Creative Commons Attribution License](https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Data Availability Statement: All relevant data are within the paper and its Supporting Information files.

Funding: This research is supported by the National Natural Science Foundation of China (No. 61303223, <http://www.nsf.gov.cn>) to J.S, the National Natural Science Foundation of China (No.61572019, <http://www.nsf.gov.cn>) to S.W. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

1 Introduction

With the promotion of new medical reform policies and the rapid development of medical information, Electronic Medical Record (EMR) [1] has become an inevitable outcome of network information technology in the medical field. An EMR is an electronic patient record of a specific system that is created, stored and used electronically. The results of the patient's diagnosis and treatment results can be transmitted through the hospital's computer network or the health card (optical card and IC card). The sharing of information resources brings great convenience to medical care. Unlike EMR, Personal Health Records (PHR) [2] are health information created and managed by patients themselves through the Internet. With the development

Competing interests: The authors have declared that no competing interests exist.

of cloud computing, patients upload PHR files to cloud servers through mobile devices, which saves local storage space and expands information sharing.

A large amount of data is stored in the cloud, making traditional clouds difficult to meet the current needs. The massive increase in stored data would not only cause great pressure on the cloud, but also lead to network congestion and transmission delays. For example, large enterprises need to pay for expensive bandwidth if they completely relies on the cloud for complex data processing; some requirements require timely response, such as payment links, car avoidance technologies involved in autonomous driving, and even when providing emergency medical services, data delays or cloud network failures can have serious consequences that cannot be measured. In order to solve the above problems, Bonomi F et al. of the US company Cisco first proposed the concept of fog computing in 2012 [3]. Fog computing is an extension of cloud computing and is a service computing paradigm for paravirtualized frameworks [4,5]. The fog server is set between the cloud server and the IoT devices, so as to the storage and calculation of data are transferred as much as possible to the fog servers. So, the fog computing helps to reduce the workload of the cloud server and improve the efficiency of the entire system. The fog computing service framework is shown in Fig 1.

Fig 1. Fog computing service structure.

<https://doi.org/10.1371/journal.pone.0207543.g001>

When sensitive data are outsourced to fog nodes which are similar to cloud platform, the data security and privacy concerns still impede the adoption of fog computing as data owners lose the physical control over their data in fog nodes or cloud. Semi-trusted cloud servers may leak and tamper with PHR information or non-authorized users steal sensitive patient information for commercial benefit. If sensitive information is used by unauthorized users or third parties, the doctor may obtain a wrong medical record, resulting in misdiagnosis. How to solve the security and privacy issues in the PHR system has become one of the most important challenge.

Encryption technology is the most critical technology to ensure information security. Goyal et al. [6] formulated the ABE into two types: the ciphertext-policy attribute-based encryption (CP-ABE) and the key-policy attribute-based encryption (KP-ABE). CP-ABE is considered one of the most appropriate encryption methods to achieve fine-grained access control. This approach allows the data owner to perform access control by setting the access structure. Due to the one-to-many communication characteristics of the ABE system, flexible access control encryption schemes are being proposed. In 2010, Ibraimi et al. [7] applied ABE to PHR security management to achieve flexible access control. However, this scheme did not give concrete proof of security. In 2013, Li et al. [8] used attribute-based encryption technology to encrypt PHR files of patients, achieving scalability and fine-grained access control for PHR. Unlike other solutions, this scheme supports multiple data owner application scenarios. Compared to the traditional single-authority CP-ABE schemes, the attributes come from different attribute authorities in the multi-authority CP-ABE schemes. In addition, it does not cause single point of failure and key leakage, which makes the multi-authority CP-ABE schemes more practical in cloud-fog computing.

In addition to data security issues, supporting outsourced partial computing operations and efficient searching of encrypted data are also an important feature in practical applications.

1.1 Related work

In this section, we discuss the related work of this article.

1.1.1 Multi-authority ABE. A single authority attribute encryption scheme manages a large number of user attribute sets by only one attribute authority, which easily causes network congestion and reduces system efficiency. This does not meet the actual work needs. In order to improve the efficiency of the single attribute authority, multiple authority ABE schemes have been proposed. In 2007, Chase et al. [9] introduced the first multi-authority ABE scheme. Independent authority supervise attributes and distribute keys to improve the security of the key. However, the encryption algorithm of the scheme is not flexible enough. In 2013, Yang et al. [10] proposed a multi-authority cloud storage data access control (DAC-MACS) scheme. The partial decryption calculation is outsourced to the server by a token-based decryption method, and the scheme also supports instant attribute revocation. However, the global certification authority has a huge amount of bilinear calculations. Subsequently, some multi-authority large-universe ABE schemes [11,12] have been proposed. In 2018, Zhang et al. [13] proposed a multi-authority CP-ABE scheme with white box tracking. The access policy can be expressed as any monotonous access structure, and the ciphertext size grows linearly with the rows of the access matrix.

1.1.2 Keyword search over encrypted data. Searchable encryption (SE) can be classified into two types: symmetric encryption with keyword search (SEKS) and public key encryption with keyword search (PEKS). There are some attempts to combine data encryption and searchable encryption to ensure the security of uploading to the cloud. In 2004, Boneh et al. [14] proposed the first concept of PEKS. However, the server has a large computational

overhead during the matching process between the trapdoor and the index. Then, search schemes [15–19] with different characteristics have been proposed. In 2017, Cui et al. [20] proposed a keyword search encryption scheme that supports effective revocation in cloud computing. At the same time, it supports certifiable keyword search and effective user revocation to meet the application scenarios of multiple data owners and data users. To the best of our knowledge, there is no searchable encryption scheme designed for cloud-fog environments currently.

1.1.3 Outsourced ABE. In ABE, aim to reduce the local computation cost, outsourcing complex operations to a cloud server becomes an important and popular problem. In 2011, Green et al. [21] constructed the outsourced decryption ABE scheme in order to save the local computing time. The user did not require bilinear pairing operations during the decryption phase. But this scheme can not guarantee the correctness of the transformed key. In 2016, Wang et al. [22] proposed a verifiable outsourced attribute encryption scheme based on dual-system encryption technology and composite order bilinear group, it is less efficient. Recently, In 2018, Jiang et al. [23] proposed a revocable outsourcing attribute based encryption scheme. The storage service manager distributes the attribute key for the user through the binary status tree, thereby implementing user revocation and attribute revocation. There are several applications of outsourced ABE in [24–26].

1.2 Our contribution

In this article, we propose a searchable personal health records framework with fine-grained access control in cloud-fog computing. Roughly, the key points of our work are described below:

1. We designed a hybrid searchable encryption scheme based on cloud-fog computing. The fog node bridges between the intelligent terminal and the cloud, the data owner and data user can be directly connected to fog nodes, and each fog node is connected to the cloud, reducing unnecessary data transmission.
2. In order to meet the resource-constrained terminal equipment, the novel multi-authority CP-ABE was proposed to support both outsourced encryption and outsourced decryption scheme. Without divulging data privacy, most local calculations are outsourced to fog nodes, enabling data users to enjoy high-rate, low-latency, high-quality services.
3. This article proposes an attribute-based searchable encryption scheme, which realizes one-to-many communication. Data users can query relevant ciphertexts according to the keywords they specify, narrowing the scope of retrieval in massive document.
4. Formal security and performance analysis proves that our scheme is safe and feasible under cloud-fog computing. In addition, it achieve secure data sharing and effectively protect the confidentiality of data.

1.3 Organization

The remaining structure of this paper is organized as follows: In Section 2, we review the relevant background knowledge of this scheme. Section 3 presents system model and security model throughout the paper. In Section 4, we give a detailed description of the specific algorithm and the correctness analysis of the scheme. We analyzed the security and discuss the performance of our schemes with comparison to several related works in Section 5 and Section 6. Finally, we conclude this scheme in Section 7.

2 Preliminaries

This section mainly gives the basic concept of access structure; then introduces bilinear maps and uses it as the main mathematical tool to construct the encryption algorithm proposed in this paper; the definition of the linear secret sharing scheme is given; and finally some difficult problems are introduced to prove the security of this scheme.

2.1 Access structure

In order to achieve fine-grained access control in an ABE scheme, the following access control structure is defined.

Definition 1 (Access structure [21]). Let $P = \{P_1, P_2, \dots, P_n\}$ be a set of n participants. For $\forall B, C$, if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$, we call $\mathbb{A} \subseteq 2^{\{P_1, \dots, P_n\}}$ is monotonous. An access structure is a collection $\mathbb{A}$ of non-empty subsets of $P = \{P_1, P_2, \dots, P_n\}$, namely $\mathbb{A} \subseteq 2^{\{P_1, \dots, P_n\}} \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.

2.2 Bilinear maps

Definition 2 (Bilinear Maps [21]). Let $\mathbb{G}$ and $\mathbb{G}_T$ are two groups of prime order p . Let g be a generator of $\mathbb{G}$. The map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is called a bilinear pairing operation. The mapping e satisfies the following properties:

1. Bilinearity: For all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degenerate: $e(g, g) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_T}$ is the unit of $\mathbb{G}_T$.
3. Computability: For all $u, v \in \mathbb{G}$, there is a valid algorithm to calculate $e(u, v)$.

2.3 Linear secret sharing scheme

Definition 3 (Linear Secret Sharing scheme (LSSS) [6]). Let $P = \{P_1, P_2, \dots, P_n\}$ be a set of participants. (M, ρ) represents an access structure $\mathbb{A}$, where M is the shared generator matrix of $l \times n$ and ρ is a mapping. For all $i = 1, 2, \dots, l$, function ρ maps the i row of M to the corresponding attribute. A linear secret sharing scheme consists of the following two effective algorithms:

1. Secret sharing algorithm: To share a secret $s \in \mathbb{Z}_p$. The algorithm randomly chooses $v_2, \dots, v_n \in \mathbb{Z}_p$ and the column vector $\mathbf{v} = (s, v_2, \dots, v_n)$. Then calculate $\lambda_i = (M \cdot \mathbf{v})_i$, where λ_i belongs to the secret share value obtained by the entity $\rho(i)$.
2. Secret reconstruction algorithm: Let $S \in \mathbb{A}$ be any set of authorized users, we define $I \subset \{1, 2, \dots, l\}$ as $I = \{i: \rho(i) \in S\}$. There is a constant coefficient $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ that satisfies $\sum_{i \in I} \omega_i M_i = (1, 0, \dots, 0)$. The recovered secret will be $\sum_{i \in I} \omega_i \lambda_i = s$. The set of constants can be found in polynomial time.

2.4 Hardness assumptions

Decisional q -Parallel Bilinear Diffie-Hellman Exponent (q -parallel DBDHE) Assumption [27] A group $\mathbb{G}$ with prime order p is selected through security parameters, and g is a generator

of $\mathbb{G}$. Randomly choose $a, s, b_1, \dots, b_q \in \mathbb{Z}_p$, and given

$$\begin{aligned} \mathbf{y} = & (g, g^s, g^a, \dots, g^{a^q}, g^{a^{q+2}}, \dots, g^{a^{2q}} \\ & \forall_{1 \leq j \leq q} g^{s \cdot b_j}, g^{a/b_j}, \dots, g^{a^q/b_j}, g^{a^{q+2}/b_j}, \dots, g^{a^{2q}/b_j} \\ & \forall_{1 \leq j, k \leq q, k \neq j} g^{a \cdot s \cdot b_k/b_j}, \dots, g^{a^q \cdot s \cdot b_k/b_j}) \end{aligned}$$

It's hard to distinguish a valid tuple $e(g, g)^{a^{q+1} \cdot s} \in \mathbb{G}_T$ from a random element R in $\mathbb{G}_T$. An algorithm $\mathcal{B}$ outputs $v \in \{0, 1\}$ has advantage ϵ in solving q -parallel DBDHE in $\mathbb{G}$ if

$$|\Pr[\mathcal{B}(\mathbf{y}, T = e(g, g)^{a^{q+1} \cdot s}) = 0] - \Pr[\mathcal{B}(\mathbf{y}, T = R) = 0]| \geq \epsilon$$

Definition 4. we say that the q -parallel DBDHE assumption holds if no polynomial time algorithm to solve the q -parallel DBDHE problem with non-negligible advantage.

Decisional Bilinear Diffie-Hellman (DBDH) Assumption [28,29]. Let g be a generator of $\mathbb{G}$ and $\beta, \gamma, z \in \mathbb{Z}_p$ be selected at random. If the challenger gives adversary $(g, g^\beta, g^\gamma, g^z)$, it must be difficult for the adversary to distinguish a valid tuple $e(g, g)^{\beta\gamma z} \in \mathbb{G}_T$ from a random element $R \in \mathbb{G}_T$.

An algorithm $\mathcal{B}$ outputs $v \in \{0, 1\}$ has advantage ϵ in solving DBDH in $\mathbb{G}$ if

$$|\Pr[\mathcal{B}(g, g^\beta, g^\gamma, g^z, e(g, g)^{\beta\gamma z}) = 0] - \Pr[\mathcal{B}(g, g^\beta, g^\gamma, g^z, Z = R) = 0]| \geq \epsilon$$

Definition 5. we say that the DBDH assumption holds if all polynomial time algorithm have at most a negligible advantage in solving the DBDH problem.

3 System overview

3.1 Definition of system model

The system model of this system is shown in Fig 2. The labels (1)-(6) in the figure correspond to the 6 algorithms in our scheme, namely system establishment, key generation, file encryption, trapdoor generation, ciphertext retrieval and file decryption algorithms. It contains the following 6 entities:

Central Authorities (CA): We assume that there are K central authority CA, for example, the health department, education department, state government, etc. The CA generates global public parameters for the PHR system and distribute secret keys based on the user's globally unique identifier id . Each CA works independently and does not need to communicate with each other, and at least one of the CAs is honest and not curious. Note that CA does not participate in any attribute-related operations.

Attribute Authorities (AA): Here are D attribute authority AA. Each AA manages different attribute domains $S_d, \mathbb{S} = \cup_{d=1}^D S_d$ represents the set of all attributes in the entire system. For $\forall i \neq j \in \{1, 2, \dots, D\}$, where $S_i \cap S_j = \emptyset$. AA generates related secret keys based on the user's attributes. For example, a hospital can assign the Hospital A attribute to all employees. Health associations can assign different doctors or nurses attributes related to medical professional licenses, such as dermatologists and psychologists. Each AA is independent of each other, one attribute is managed by only one AA, but one AA can manage multiple attributes.

Cloud Server Provider (CSP): The Cloud Server Provider (CSP) is a semi-trusted entity that is mainly responsible for storing encrypted PHR files and keyword indexes, and providing access services for authorized users.

Fog Nodes (FN): A fog node is a trusted entity that is at the edge of the network and has the ability to computing, storage and network services. It is responsible for partial encryption and

Fig 2. System model.

<https://doi.org/10.1371/journal.pone.0207543.g002>

decryption operations. The fog node helps the DO to generate partial ciphertext and upload all ciphertext to the CSP. It can also decrypt some ciphertext downloaded from the CSP.

Data Owner (DO): The data owner specifies an access policy, encrypts PHR files and keyword sets, and uploads ciphertext and indexes to the CSP.

Data User (DU): The data user downloads the encrypted PHR file from the CSP. The DU can decrypt successfully only if the attributes of the DU satisfy the access policy. For example, doctors and nurses must visit the patient's PHR file in order to properly diagnose the condition and care.

For a more intuitive description of the scheme, we use personal health records to illustrate. Consider the following scenario: Alice who has a skin disease wants to find an expert to check through an online medical facility. Alice's needs is (hospital A $\wedge$ dermatologist). In order to protect the confidentiality of personal health records, Alice needs to encrypt medical health records under hospital A $\wedge$ dermatologist condition before uploading data to the CSP. The CSP then searches for a doctor who satisfies hospital A $\wedge$ dermatologist in its own database and sends Alice's personal health record to the qualified doctor Bob. Bob can decrypt the record for Alice to continue treatment.

3.2 Algorithm definition

This scheme consists of 6 algorithms: system setup, key generation, file encryption, trapdoor generation, search over ciphertext, and file decryption. Each algorithm is described as follows:

1. **System Setup:** The algorithm is executed by authority. It contains 3 sub-algorithms:

Global—Setup(1^λ): The global-setup algorithm is run by a trusted third party. It takes as no input other than the security parameters λ and outputs the global public parameter GPK.

CA—Setup(GPK, k): The CA-setup algorithm is run by each CA_k . It input the GPK and the tag k of the CA, and then output the public parameter (CPK $_k$, CAPK $_k$) and the master key CASK $_k$, and the CAPK $_k$ is used only by the CA_k .

AA—Setup(GPK, d , S_d): The AA-setup algorithm is performed by each AA_d . It takes as inputs the GPK, the tag d of the AA_d and attributes set S_d managed by the AA_d . It outputs the public parameter (APK $_d$, AAPK $_d$) and the master key AASK $_d$. AAPK $_d$ is used only by the AA_d .

2. **Key Generation:** This algorithm is performed by the CA_k and the AA_d . It contains 2 sub-algorithms:

CA—KeyGen(GPK, CMSK $_k$, AAPK $_d$, id): The CA-key generation algorithm is performed by the CA_k . It takes as inputs the global public parameter GPK, the master key CMSK $_k$ of the CA_k , part of the public parameters AAPK $_d$ of the AA_d , and the unique identifier id of the data user. It outputs the user-center-key (ucsk $_{id,k}$, ucpk $_{id,k}$), where ucpk $_{id,k}$ is called user-center-public-key.

AA — KeyGen(att , {ucpk $_{id,k}$ | $k \in \mathbb{K}$ }, {VerifyKey $_k$ | $k \in \mathbb{K}$ }, AMSK $_d$, GPK): The AA-key generation algorithm is executed by the AA_d . It intakes an attributes att , user-center-public-key ucpk $_{id,k}$, verification key VerifyKey $_k$, master key AMSK $_d$ and global public parameters GPK. It outputs the secret key SK $_{DU}$ of the DU and the public/secret key pair (pk_O , sk_O) of the DO.

3. **File Encryption:** This algorithm is executed by the FN and the DO. It contains 2 sub-algorithms:

Fog—Encrypt(GPK, {APK $_d$ }, (M , ρ)): The outsourced encryption algorithm is performed by the FN. The algorithm intakes the global public parameter GPK, the public parameter APK $_d$ of the AA_d , and an access structure (M , ρ). It outputs the intermediate ciphertext CT_{Fog} .

Do—Encrypt(m , GPK, CT_{Fog} , sk_O , pk_U): The local encryption algorithm is usually performed by the DO. It intakes the plaintext message m , the global public parameter GPK, DO's secret key sk_O , DU's public key pk_U and the ciphertext CT_{Fog} . It outputs all ciphertext CT_1 , CT_2 and delivers it to the FN. Finally, the FN uploads the ciphertext $CT = (CT_1, CT_2)$ to the CSP.

4. **TrapGen**($\tilde{w}$, SK $_{DU}$, pk_O): The trapdoor generation algorithm is performed by the DU. It intakes a desired keyword $\tilde{w}$, the secret key SK $_{DU}$ and the public key pk_O . Finally, it outputs the trapdoor $T_{\tilde{w}}$ and the pre-decryption key SK' $_{DU}$.

5. **Search over Ciphertext:** This algorithm is executed by the CSP to determine whether the keywords in the ciphertext match the keywords of the trapdoor.

Search(CT_2 , $T_{\tilde{w}}$, pk_O , pk_U): The search algorithm is performed by the CSP. It intakes the index CT_2 , trapdoor $T_{\tilde{w}}$ and the public key pk_O , pk_U . If the trapdoor and index match successfully, returns the search result to the FN.

6. **File Decryption:** This algorithm is executed by the FN and the DU. Including 2 sub-algorithms:

Fog – Decrypt(CT, SK'_{DU}): The outsourced decryption algorithm is performed by the FN. It intakes the ciphertext CT and the pre-decryption key SK'_{DU} . It outputs the partially decrypted ciphertext (C, Ω) to the DU.

Do—Decrypt(C, Ω, RK): The final decryption algorithm is performed by the DU. It intakes the partially decrypted ciphertext (C, Ω) and the retrieval key RK . It outputs the plaintext message m .

3.3 Definition of security model

In the fog-cloud storage system, the CSP is also curious about the contents of the encrypted data. We assume that the CSP will correctly perform the tasks assigned by the central authority and attribute authority. The AA and CA can be corrupted or attacked. To demonstrate the security of our scheme, we design two security games: indistinguishability against selective ciphertext-policy and chosen ciphertext attack (IND-sCP-CCA) game and trapdoor privacy game.

Game 1. Ciphertext indistinguishability. The security of this scheme is defined by the following game run between a challenger $\mathcal{B}$ and an adversary $\mathcal{A}$. $\mathcal{A}$ can corrupt CAs and AAs by specifying $\mathbb{K}'_c \subset \mathbb{K}$ and $\mathbb{D}'_c \subset \mathbb{D}$ after seeing the public parameters, where $\mathbb{K} \setminus \mathbb{K}'_c \neq \emptyset$ and $\mathbb{D} \setminus \mathbb{D}'_c \neq \emptyset$. The security game is defined as follows:

Init. $\mathcal{A}$ exposes a challenged access structure (M^*, ρ^*) , where M^* is an $l^* \times n^* \leq q$ matrix.

Setup. $\mathcal{B}$ runs algorithms **Global—Setup**, **CA—Setup** and **AA—Setup**. The public parameter GPK , $(CPK_k, CAPK_k)$ and $(APK_d, AAPK_d)$ are sent to $\mathcal{A}$. We allow $\mathcal{A}$ to corrupt authority $\mathbb{K}'_c \subset \mathbb{K}$ and $\mathbb{D}'_c \subset \mathbb{D}$ (where $\mathbb{K} \setminus \mathbb{K}'_c \neq \emptyset, \mathbb{D} \setminus \mathbb{D}'_c \neq \emptyset$). $\mathcal{A}$ submits an access policy (M^*, ρ^*) to $\mathcal{B}$, where M^* is a matrix of $l^* \times n^* \leq q$, ρ^* maps the rows of M^* to attributes. For uncorrupted authorities in $\mathbb{K} \setminus \mathbb{K}'_c$ and $\mathbb{D} \setminus \mathbb{D}'_c$, $\mathcal{B}$ sends only the public keys to $\mathcal{A}$. For corrupted authorities $\mathbb{K}'_c$ and $\mathbb{D}'_c$, $\mathcal{B}$ sends both the master key $\{CMSK_k | k \in \mathbb{K}'_c\}$ and $\{AMSK_d | d \in \mathbb{D}'_c\}$ to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ can performs the following secret key queries many times, in which limiting the secret key attribute set to be queried does not meet the access policy M^* to be challenged. In other words, $\mathcal{A}$ cannot ask for a key which can be decrypt in combination with any keys that can be obtained from corrupted CAs and AAs:

CKQ(id, k): For each unpurchased $CA_k (k \in \mathbb{K} \setminus \mathbb{K}'_c)$, $\mathcal{A}$ submits a tuple (id, k) to $\mathcal{B}$, where id is the user's global identifier and k is the tag of an uncorrupted authority CA. $\mathcal{B}$ runs the algorithm **CA—KeyGen** and returns the corresponding user-central-key $(ucsk_{id,k}, ucpk_{id,k})$ to $\mathcal{A}$.

AKQ($att, \{ucpk_{id,k} | k \in \mathbb{K}\}, d$): For each unpurchased $AA_d (d \in \mathbb{D} \setminus \mathbb{D}'_c)$, $\mathcal{A}$ submits a tuple $(att, \{ucpk_{id,k} | k \in \mathbb{K}\}, d)$ to $\mathcal{B}$, where att is an attribute of the attribute set S_d , $\{ucpk_{id,k} | k \in \mathbb{K}\}$ are the central-public-keys of the user id , and d is the tag of an uncorrupted authority AA. $\mathcal{B}$ runs algorithm **AA—KeyGen** and outputs $\perp$ if $ucpk_{id,k}$ are invalid. Otherwise, user-attribute-key $uask_{att,id}$ are returned to $\mathcal{A}$.

Challenge. $\mathcal{A}$ sends two equal length messages m_0 and m_1 . $\mathcal{B}$ selects a random bit $v \in \{0, 1\}$, and encrypts m_v under (M^*, ρ^*) . Then return the challenge ciphertext CT^*_1 to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ conducts more secret key queries similar to **Phase 1**.

Guess. $\mathcal{A}$ submits a guess $v' \in \{0, 1\}$. If $v' = v$, the $\mathcal{A}$ wins the safety game, otherwise $\mathcal{A}$ fails. The advantage of the $\mathcal{A}$ in breaking this game is $Adv^{\text{IND-sCP-CCA}}_{\mathcal{A}} = |\Pr[v = v'] - \frac{1}{2}|$.

Definition 6. The proposed scheme is IND-sCP-CCA secure if all polynomial time adversary $\mathcal{A}$ have at most a negligible advantage in the above security game.

Game 2. Trapdoor privacy. **Setup.** Given a security parameter λ , $\mathcal{B}$ generates the DU's public key pk_U and the DO's public key pk_O .

Phase 1. $\mathcal{A}$ adaptively issues polynomial many times following queries.

Trapdoor Query $\mathcal{O}_T$: $\mathcal{A}$ can ask any keyword's trapdoor.

Index Queries $\mathcal{O}_I$: $\mathcal{A}$ can ask any keyword's index.

Challenge. $\mathcal{A}$ sends two equal length keywords (w_0^*, w_1^*) , with the restriction that (w_0^*, w_1^*) have not been queried for trapdoors nor indexes. $\mathcal{B}$ selects a random bit $v \in \{0, 1\}$ and generates the trapdoor $T_{w_b^*}$ of keyword w_b^* and returns it to $\mathcal{A}$.

Phase 2. Same as **Phase 1**, with the restriction $w \neq w_0^*, w_1^*$.

Guess. $\mathcal{A}$ submits a guess $v' \in \{0, 1\}$. If $v' = v$, the $\mathcal{A}$ wins this game, otherwise $\mathcal{A}$ fails.

The advantage of the $\mathcal{A}$ in breaking this game is $\text{Adv}_{\mathcal{A}} = |\Pr[v = v'] - \frac{1}{2}|$.

Definition 7. The proposed scheme is trapdoor privacy secure if all polynomial time adversary $\mathcal{A}$ have at most a negligible advantage in the above security game.

4 Algorithm construction

In this part, the scheme address the hospital scene to construct a PHR sharing scheme based on cloud-fog computing. The patient encrypts personal medical data according to different access policies and stores it in the cloud. Doctors need to download the PHR file from the cloud if they want to view the case. As the distance of transmission of encrypted data from the cloud to the mobile device is going up, communication costs and delays are increasing. By deploying the fog server in the hospital, the total response time is reduced. In hospitals, it is difficult for a dermatologist to obtain dermatological data in a massive medical database. Using keyword-based search technology can not only access the data in the cloud, but also perform keyword search directly in the cloud. The doctor only downloads the files he needs, effectively reducing communication costs.

4.1 Detailed description of our scheme

A. System setup. During the system setup phase, global public parameters are generated. The CA_k and AA_d generate their own public key and secret key, respectively. The phase contains 3 sub-algorithms: **Global—Setup**, **CA—Setup** and **AA—Setup**.

Global—Setup(1^λ): Only trusted third parties run the algorithm. It takes as no input other than the security parameters λ . This algorithm chooses a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, where $\mathbb{G}$ and $\mathbb{G}_T$ are two multiplicative cyclic groups of prime order p ($p > 2^\lambda$). Let g be a generator of $\mathbb{G}$, and randomly pick $h \in \mathbb{G}$. The $\Sigma_{\text{sign}} = (\text{KeyGen}, \text{SignKey}, \text{VerifyKey})$ is a secure unforgeable signature scheme. It also chooses a hash function $H : \{0, 1\}^* \rightarrow \mathbb{G}$. Return the global public parameters $\text{GPK} = (\mathbb{G}, \mathbb{G}_T, p, g, h, \Sigma_{\text{sign}}, H_1)$.

CA—Setup(GPK, k): Each CA_k runs this algorithm. It takes as inputs the global public parameters GPK and the tag k of the CA . CA_k runs the key generation algorithm

KeyGen $\rightarrow (\text{SignKey}_k, \text{VerifyKey}_k)$ of the scheme Σ_{sign} and selects a random exponent $\alpha_k \in \mathbb{Z}_p$. Then Central authority publishes the public key $\text{CPK}_k = e(g, g)^{\alpha_k}$, $\text{CAPK}_k = \text{VerifyKey}_k$ and keeps the master key $\text{CMSK}_k = (\alpha_k, \text{SignKey}_k)$ secret.

AA—Setup(GPK, d, S_d): Each AA_d runs this algorithm. It takes as inputs the global public parameter GPK , the tag d of the AA and S_d is a set of attributes managed by AA_d . For each attribute $\text{att} \in S_d$, select $s_{\text{att}} \in \mathbb{Z}_p$ and calculates $T_{\text{att}} = g^{s_{\text{att}}}$. For each $k \in K$, AA_d randomly chooses $v_{d,k} \in \mathbb{Z}_p$ and computes $V_{d,k} = g^{v_{d,k}}$. Then it publishes $\text{APK}_d = \{T_{\text{att}} \mid \text{att} \in S_d\}$, $\text{AAPK}_d = \{V_{d,k} \mid k \in K\}$ as its public key and the master key is kept as $\text{AMSK}_d = (\{s_{\text{att}} \mid \text{att} \in S_d\}, \{V_{d,k} \mid k \in K\})$.

B. Key generation. We assume that there are two central authorities: the health department, the education department, and two attribute authorities: hospitals, health associations. Because the keys and attribute sets are related in the CP-ABE scheme, the attribute authority

Fig 3. CA_s and AA_s send key for user id .

<https://doi.org/10.1371/journal.pone.0207543.g003>

AA_d will generate a corresponding attribute key according to the user's attribute set. Consider the following scenario: Generate a key for a dermatologist working in Hospital A. The key generation phase contains 2 sub-algorithms: **CA—KeyGen** and **AA—KeyGen**. The key generation process is shown in Fig 3.

CA—KeyGen(GPK, CMSK_k, AAPK_d, id): A data user sends its globally unique identifier id to the CA_k to request the user-central-key. The CA_k randomly picks element $r_{id,k} \in \mathbb{Z}_p$ and sets the user-center-key: $ucsk_{id,k} = g^{z_k} h^{r_{id,k}}$, $\Gamma_{id,k} = g^{r_{id,k}}$. For $d \in [1, D]$, CA_k calculates $L_{id,k,d} = V_{d,k}^{r_{id,k}}$ and generates the signature $sign_{id,k} = \text{Sign}(\text{SignKey}_k, id || k || \Gamma_{id,k} || \{L_{id,k,d} | d \in [1, D]\})$. Simultaneously, the user-central-public-key $ucpk_{id,k} = (id, k, \Gamma_{id,k}, \{L_{id,k,d} | d \in \mathbb{D}\}, \delta_{id,k})$ is output.

AA—KeyGen(att, $\{ucpk_{id,k} | k \in \mathbb{K}\}$, $\{\text{VerifyKey}_k | k \in \mathbb{K}\}$, AMSK_d, GPK): For attribute $att \in S_d$. The data user id sends its user-center-public-key $\{ucpk_{id,k} | k \in \mathbb{K}\}$ to the AA_d to request the user-attribute-key.

① For any $k \in [1, K]$, the AA_d verifies the following equation:

$$\text{Verify}(\text{VerifyKey}_k, id || k || \Gamma_{id,k} || \{L_{id,k,d} | d \in [1, D]\} || sign_{id,k}) \stackrel{?}{\rightarrow} \text{True}$$

$$e(g, L_{id,k,d}) = e(V_{d,k}, \Gamma_{id,k})$$

If the above equation holds, then ② is performed. Otherwise, the AA_d outputs $\perp$. It indicates that the user-central-public-key $ucpk_{id,k}$ submitted by user is invalid.

② For any $k \in [1, K]$, the AA_d generates the user-attribute-key $uask_{att,id}$ for the user

$$uask_{att,id} = \prod_{k=1}^K (L_{id,k,d})^{s_{att}/v_{d,k}} = \prod_{k=1}^K (g^{v_{d,k} \cdot r_{id,k}})^{s_{att}/v_{d,k}} = T_{att} \sum_{k=1}^K r_{id,k}$$

After that, algorithm randomly picks $\beta, \gamma \in \mathbb{Z}_p^*$ and computes g^β, g^γ . It returns the public/secret key pair of DO's and DU's as $(pk_O, sk_O) = (g^\gamma, \gamma)$ and $(pk_U = g^\beta, sk_U = \beta)$, respectively. So DU's secret key $SK_{DU} = (\{ucsk_{id,k}, \Gamma_{id,k} \mid k \in K\}, \{uask_{att,id} \mid att \in S_d\}, \beta)$.

C. File encryption. Before uploading the PHR file to the CSP, the patient needs to encrypt the file based on the access policy (hospital A $\wedge$ dermatologist) and sends an access policy to FN. The encryption algorithm contains two sub-algorithms: **Fog—Encrypt** and **Do—Encrypt**.

Fog—Encrypt(GPK, $\{APK_d\}$, (M, ρ)): Here (M, ρ) is an LSSS access structure. Assuming that M is an $l \times n$ matrix. Function $\rho(\cdot)$, which is an injective function, maps each rows of M to different attributes. For $\forall i \in \{1, 2, \dots, l\}$, fog nodes are randomly selects $\lambda'_i, r_i \in \mathbb{Z}_p^*$ and sets the ciphertext CT_{Fog} as follows:

$$C_{i,1} = h^{\lambda'_i} \cdot T_{\rho(i)}^{-r_i}, C_{i,2} = g^{r_i}$$

Output part of the ciphertext $CT_{Fog} = ((M, \rho), \{C_{i,1}, C_{i,2}, \lambda'_i\}_{i \in [1,l]}).$

Do—Encrypt(m , GPK, CT_{Fog} , sk_O, pk_U): m is the PHR file to be shared by patient. The DO first creates vectors $\mathbf{V} = (s, v_2, \dots, v_n) \in \mathbb{Z}_p^n$, where s is the random secret to be shared. From $i = 1$ to l , it gets the sub-secret $\lambda_i = \mathbf{V} \cdot \mathbf{M}_i$ by computing, where $\mathbf{M}_i$ is the i -th row of matrix M . The intact data ciphertext CT_1 can be created by the following calculation:

$$C = m \cdot \prod_{k=1}^K e(g, g)^{\alpha_k \cdot s}, C' = g^s, \{C_{i,3} = \lambda_i - \lambda'_i \mid i \in [1, l]\}$$

All data ciphertext is published as $CT_1 = \{(M, \rho), C, C', \{C_{i,1}, C_{i,2}, C_{i,3}\}_{i \in [1,l]}\}.$

Then, the DO extracts keywords from the PHR file to form a set of keywords W_m . For each keyword $w_i \in W_m$, the algorithm randomly selects $t_i \in \mathbb{Z}_p^*$ to generate a keyword index $CT_2 = (\{I_i\}_{w_i \in W_m})$ and send it to the FN.

$$I_{1,i} = H(w_i)^{sk_O} \cdot g^{t_i}, I_{2,i} = pk_U^{t_i}$$

Subsequently, the FN uploads the ciphertext $CT = (CT_1, CT_2)$ to the CSP.

D. Trapdoor generation. After the doctor issues a search request to the CSP, it generates trapdoors to search for keywords. The trapdoor generation process as follows:

TrapGen($\tilde{w}, SK_{DU}, pk_O$): It takes search query for the keyword $\tilde{w}$ as inputs. It creates the trapdoor as $T_{\tilde{w}} = e(H(\tilde{w})^{sk_U}, pk_O)$. The DU delivers the trapdoor $T_{\tilde{w}}$ to the CSP. Then DU chooses a random number $\delta \in \mathbb{Z}_p^*$ to blind his secret key and computes $ucsk'_{id,k} = (g^{z_k} h^{r_{id,k}})^\delta$,

$\Gamma'_{id,k} = (g^{r_{id,k}})^\delta, uask'_{att,id} = (T_{att} \prod_{k=1}^K r_{id,k})^\delta$. DU holds the unique retrieval key $RK = \delta$. Finally, the DU outputs the pre-decryption key $SK'_{DU} = (\{ucsk'_{id,k}, \Gamma'_{id,k} \mid k \in K\}, \{uask'_{att,id} \mid att \in S_d\})$ and sends it to the FN.

E. Search over ciphertext. When the CSP receives the user's search request, the CSP performs the following algorithm to search the matched health records:

Search($CT, T_{\tilde{w}}, pk_O, pk_U$): When gained the DU's search query, the CSP first checks whether the DU's attribute set satisfies with access structure (M, ρ) . If it is true, the CSP uses the following equation to checks if the trapdoor $T_{\tilde{w}}$ and index CT_2 match.

$$e(I_{1,i}, pk_U) = T_{\tilde{w}} \cdot e(I_{2,i}, pk_O)$$

If the equation does not hold, $\perp$ is returned. Otherwise, it indicates that the matching is successful and then the CSP returns the corresponding search results CT to the FN.

F. File decryption. If the doctor's attribute satisfies the access policy, the doctor uses retrieval key to successfully decrypt the partially encrypted ciphertext and obtain the patient's PHR file. The decryption algorithm includes 2 aspects: **Fog—Decrypt** and **Do—Decrypt**.

Fog—Decrypt(CT, SK'_{DU}): The FN downloads the ciphertext CT from the CSP and the pre-decryption key SK'_{DU} from the DU. Once that attribute set of the DU does not satisfy the access policy, the FN outputs $\perp$. If not, there must be a constant set $\{c_i \in \mathbb{Z}_p\}_{i \in I}$ that satisfies $\sum_{i \in I} c_i M_i = (1, 0, \dots, 0)$. If $\{\lambda_i\}$ is a valid share of secret s , there is $\sum_{i \in I} c_i \lambda_i = s$. The FN uses the pre-decryption key SK'_{DU} to calculate the intermediate ciphertext Ω as follows:

$$\Omega = \frac{\prod_{k=1}^K e(C', \text{ucsk}'_{id,k})}{\prod_{i \in I} (e(C_{i,1} \cdot h^{C_{i,3}}, \prod_{k=1}^K \Gamma'_{id,k}) e(C_{i,2}, \text{uask}'_{att,id}))^{c_i}} = e(g, g)^{\sum_{k=1}^K \alpha_k} \text{ where } r_{id} = \sum_{k=1}^K r_{id,k}$$

Finally, the FN sends the partially decryption result (C, Ω) to the DU.

Du—Decrypt(C, Ω, RK): After receiving the partially decrypted ciphertext (C, Ω) by the DU, the DU decrypts it with retrieval key $RK = \delta$. The plaintext m can be recovered by the following equation.

$$m = \frac{C}{\Omega} = \frac{m \cdot \prod_{k=1}^K e(g, g)^{\alpha_k \cdot s}}{(e(g, g)^{\sum_{k=1}^K \alpha_k})^{\frac{1}{\delta}}} \text{ where } \alpha = \sum_{k=1}^K \alpha_k$$

4.2 Correctness analysis

In this part, we will prove the correctness of our scheme by the following equations:

- (1) In order to verify whether the user-central-public-key $\text{ucpk}_{id,k}$ is valid, the calculation process is as follows:

$$e(g, L_{id,k,d}) = e(g, V_{d,k}^{r_{id,k}}) = e(g, g)^{v_{d,k} r_{id,k}}$$

$$e(V_{d,k}, \Gamma_{id,k}) = e(g^{v_{d,k}}, g^{r_{id,k}}) = e(g, g)^{v_{d,k} r_{id,k}}$$

- (3) The index and trapdoor matching process is verified as follows:

$$\begin{aligned} e(I_{1,i}, pk_U) &= e(H(w_i)^{\gamma} \cdot g^{\beta}, g^{\beta}) \\ &= e(H(w_i)^{\gamma}, g^{\beta}) \cdot e(g^{\beta}, g^{\beta}) \end{aligned}$$

$$T_{\tilde{w}} \cdot e(I_{2,i}, g) = e(H(\tilde{w})^{\beta}, g^{\gamma}) \cdot e(g^{\beta}, g)$$

(4) The Fog-Decryption process calculates:

$$\begin{aligned}
 \Omega &= \frac{\prod_{k=1}^K e(C', \text{ucsk}'_{id,k})}{\prod_{i \in I} (e(C_{i,1} \cdot h^{C_{i,3}}, \prod_{k=1}^K \Gamma'_{id,k}) e(C_{i,2}, \text{uask}'_{att,id}))^{c_i}} \\
 &= \frac{\prod_{k=1}^K e(g^s, (g^{z_k} h^{r_{id,k}})^\delta)}{\prod_{i \in I} (e(h^{z_i} \cdot g^{-s_{att} r_i}, g^{\delta r_{id}}) e(g^{r_i}, (g^{\sum_{k=1}^K r_{id,k}})^\delta))^{c_i}} \\
 &= \frac{e(g, g)^{\sum_{k=1}^K \alpha_k} e(g^s, h^{\delta r_{id}})}{e(h, g^{\delta r_{id}})^{\sum_{i \in I} \lambda_i c_i}} \\
 &= e(g, g)^{\sum_{k=1}^K \alpha_k}
 \end{aligned}$$

5 Security analysis

Theorem 1. Our system is secure against IND-sCP-CCA based on the standard model if the decisional q —parallelBDHE assumption holds and the signature scheme Σ_{sign} is existent and unforgeable.

Proof. Assuming that exists a polynomial time adversary $\mathcal{A}$ who can break the IND-sCP-CCA security of our construction with the advantage of $\varepsilon = \text{Adv}_{\mathcal{A}}^{\text{IND-sCP-CCA}}$. In the following security game, given the decisional q —parallelBDHE problem instance $(p, \mathbb{G}, \mathbb{G}_T, \vec{y}, T)$, we can build a simulator $\mathcal{B}$ to decide whether $T = e(g, g)^{a^{q+1} \cdot s}$ or not. $\mathcal{A}$ can ask for the master key of any buying center authority $\mathbb{K}'_c$ and attribute authority $\mathbb{D}'_c$. The interaction between $\mathcal{B}$ and $\mathcal{A}$ is as follows:

Init. $\mathcal{B}$ is given a decisional q —parallelBDHE problem instance $(p, \mathbb{G}, \mathbb{G}_T, \vec{y}, T)$. $\mathcal{A}$ exposes the access structure (M^*, ρ^*) to be challenged, where M^* is a $l^* \times n^* \leq q$ matrix.

Setup. To expose public parameters, $\mathcal{B}$ performs the following operations:

1. $\mathcal{B}$ selects $g, h \in \mathbb{G}$, sets $h = g^a$. $\mathcal{B}$ sets global public parameter GPK and sends it to $\mathcal{A}$.
2. For each non-purchased $CA_k (k \in \mathbb{K} \setminus \mathbb{K}'_c)$, $\mathcal{B}$ randomly chooses $\hat{\alpha}_k \in \mathbb{Z}_p$ and implicitly sets $\alpha_k = \hat{\alpha}_k + a^{q+1}$ by letting $\text{CPK}_k = e(g^a, g^{a^q}) \cdot e(g, g)^{\hat{\alpha}_k}$. $\mathcal{B}$ selects the unforgeable signature algorithm Σ_{sign} and calls the algorithm **KeyGen** $\rightarrow$ (**SignKey** $_k$, **VerifyKey** $_k$) to generate a signature key pair, and sets $\text{CAPK}_k = \text{VerifyKey}_k$.
3. Symbol X denote a set which indices i satisfied $\rho^*(i) = x$. It means that all the row in the set X match the same attribute x . For any $x (1 \leq x \leq S)$, choose the random exponent

$\hat{u}_x, v_{d,k} \in \mathbb{Z}_p, \mathcal{B}$ calculates:

$$\text{APK}_d = T_x = g^{\hat{u}_x} \cdot \prod_{i \in X} g^{\frac{aM_{i,1}^*}{b_i} + \frac{a^2M_{i,2}^*}{b_i} + \dots + \frac{a^{n^*}M_{i,n^*}^*}{b_i}}$$

$$\text{AAPK}_d = V_{d,k} = g^{v_{d,k}}$$

Finally, $\mathcal{B}$ sends GPK, $(\{\text{CPK}_k, \text{CAPK}_k\} | k \in \mathbb{K})$ and $(\{\text{APK}_d, \text{AAPK}_d\} | d \in \mathbb{D})$ to $\mathcal{A}$. Note that the simulated public parameters have the same distribution as the actual parameters.

Phase1. At this stage, $\mathcal{B}$ accepts secret key queries from the $\mathcal{A}$. Limiting $\mathcal{B}$ receives a secret key query for a set S which does not satisfy M^* .

CKQ(id, k): For each non-purchased $CA_k (k \in \mathbb{K} \setminus \mathbb{K}'_c)$, $\mathcal{A}$ submits a tuple (id, k) to $\mathcal{B}$. According to the definition of LSSS, it is not hard to find that there exist a vector

$\mathbf{w} = (w_{1,k}, w_{2,k}, \dots, w_{n^*,k}) \in \mathbb{Z}_p^{n^*}$ such that $w_{1,k} = -1$. For any i where $\rho^*(i) \in S$, we have that $\sum_{j=1}^{n^*} w_{j,k} \cdot M_{i,j}^{n^*} = 0$. $\mathcal{B}$ chooses a random number $r'_{id,k} \in \mathbb{Z}_p$, then implicitly define $r_{id,k}$ as

$$r_{id,k} = r'_{id,k} + w_{1,k}a^q + w_{2,k}a^{q-1} + \dots + w_{n^*,k}a^{q-n^*+1}$$

It performs this by setting

$$\Gamma_{id,k} = g^{r'_{id,k} + w_{1,k}a^q + \dots + w_{n^*,k}a^{q-n^*+1}} = g^{r'_{id,k} + \sum_{i \in [n^*]} w_{i,k}a^{q-i+1}} = g^{r_{id,k}}$$

The parameter $g^{a^{q+1}}$ is included during calculating $\text{ucsk}_{id,k}$. By defining $r_{id,k}$, we find that $g^{a^{q+1}}$ contains a term $g^{-a^{q+1}}$. Therefore, the parameter $g^{a^{q+1}}$ included in $\text{ucsk}_{id,k}$ can be eliminated. $\mathcal{B}$ calculates $\text{ucsk}_{id,k}, L_{id,k,d}$ as follows:

$$\begin{aligned} \text{ucsk}_{id,k} &= g^{\hat{z}_k} g^{a^{q+1}} \cdot \prod_{i=2}^{n^*} g^{a^{q+2-i} \cdot w_{i,k}} \\ &= g^{\hat{z}_k} g^{a^{q+1}} g^{a^{q+1}} g^{-a^{q+1}} g^{a^q w_{2,k}} \dots g^{a^{q+2-n^*} \cdot w_{i,k}} \\ &= g^{\hat{z}_k} h^{r'_{id,k} + w_{1,k}a^q + \dots + w_{i,k}a^{q+1-n^*}} \end{aligned}$$

$$L_{id,k,d} = \Gamma_{id,k}^{v_{d,k}} = g^{r_{id,k} \cdot v_{d,k}}$$

$\mathcal{B}$ generates a signature $\text{sign}_{id,k}$ and returns $\text{ucsk}_{id,k}$
 $\text{ucpk}_{id,k} = (id, k, \Gamma_{id,k}, \{L_{id,k,d} | d \in \mathbb{D}\}, \text{sign}_{id,k})$ gives $\mathcal{A}$.

AKQ($att, \{\text{ucpk}_{id,k} | k \in \mathbb{K}\}, d$): For each non-purchased $AA_d (d \in \mathbb{D} \setminus \mathbb{D}'_c)$, $\mathcal{A}$ submits a tuple $(att, \{\text{ucpk}_{id,k} | k \in \mathbb{K}\}, d)$ to the $\mathcal{B}$. Then $\mathcal{B}$ calls algorithm **AA—KeyGen** to verify the validity of the signature on the $\text{ucpk}_{id,k}$. If the validation is successful, $\mathcal{B}$ calculates $\text{ucsk}_{att,id}$ as

follow, otherwise output $\perp$.

$$\begin{aligned} \text{uask}_{att,id} &= T_{att}^{\hat{u}_{att}} \cdot \prod_{k=1}^K \prod_{i \in X} \prod_{j=1}^{n^*} (g^{\frac{a^j}{b_i}} \cdot r'_{id,k}) \cdot \prod_{y=1, y \neq j}^{n^*} g^{(\frac{a^{q+1+j-y}}{b_i}) \cdot w_{y,k}} M_{ij}^* \\ &= T_{att}^{\hat{u}_{att}} \cdot \prod_{k=1}^K g^{(\sum_{i \in X} \sum_{j=1}^{n^*} (\frac{a^j}{b_i}) \cdot M_{ij}^*) r'_{id,k}} g^{(\sum_{i \in X} \sum_{j=1, y=1, y \neq j}^{n^*} (\frac{a^{q+1+j-y}}{b_i}) \cdot w_{y,k} \cdot M_{ij}^*)} \\ &= \prod_{k=1}^K g^{(\sum_{i \in X} \frac{a M_{i,1}^*}{b_i} + \frac{a^2 M_{i,2}^*}{b_i} + \dots + \frac{a^{n^*} M_{i,n^*}^*}{b_i} + \hat{u}_{att} (r'_{id,k} + \sum_{y=1}^{n^*} w_{y,k} \cdot a^{q+1-y}))} \\ &= g^{\sum_{k=1}^K r_{id,k} s_{att}} \end{aligned}$$

Finally, $\mathcal{B}$ returns $SK_{DU} = (\{\text{ucsk}_{id,k}, \Gamma_{id,k} \mid k \in K\}, \{\text{uask}_{att,id} \mid att \in S_d\})$ to the $\mathcal{A}$.

Challenge. After $\mathcal{A}$ finished the query **Phase1**, he sends two equal length messages m_0 and m_1 to $\mathcal{B}$. $\mathcal{B}$ then throws a random bit $v \in \{0, 1\}$ and encrypts m_v . It creates challenge ciphertext

$$C = m_v \cdot T \cdot \prod_{k=1}^K e(g^{\hat{z}_k}, g^s)$$

$$C' = g^s$$

The most difficult things for $\mathcal{B}$ is to simulate $C_{i,1}$ since it contains terms $g^{a^j s}$. However, $\mathcal{B}$ can do the secret splitting, so that these items can be cancel out. $\mathcal{B}$ chooses random and implicitly sets vector $\mathbf{V} = (s, sa + \chi_2, sa^2 + \chi_3, \dots, sa^{n^*-1} + \chi_{n^*}) \in \mathbb{Z}_p^{n^*}$ to share secret s . For vector $\mathbf{V}$, the sharing of the secret s can be constructed as

$$\begin{aligned} \lambda_i &= (\mathbf{v}, M_i^*) \\ &= s \cdot M_{i,1}^* + (sa + \chi_2) M_{i,2}^* + (sa^2 + \chi_3) M_{i,3}^* + \dots + (sa^{n^*-1} + \chi_{n^*}) M_{i,n^*}^* \\ &= s \cdot M_{i,1}^* + \sum_{j=2}^{n^*} (sa^{j-1} + \chi_j) \cdot M_{i,j}^* \end{aligned}$$

Let R_i be a collection of $i \neq y (i \in [1, l^*])$, such that $\rho^*(i) = \rho^*(y)$. Intuitively, $\mathcal{B}$ randomly chooses $\hat{\lambda}_i, \hat{r}_1, \dots, \hat{r}_{l^*} \in \mathbb{Z}_p^*$. For each row of matrix M^* , set $\rho^*(i) = x^*$. By implicitly setting $r_i = -\hat{r}_i - sb_i$. We can simulate the challenge ciphertext as follows:

$$C_{i,1} = T_{\rho^*(i)}^{\hat{r}_i} \cdot (\prod_{j=2}^{n^*} (g^a)^{M_{ij}^* \chi_j}) \cdot (g^{sb_i})^{-\hat{u}_{\rho^*(i)}} \cdot \prod_{y \in R_i} \prod_{j=1}^{n^*} (g^{\frac{a^j \cdot s \cdot b_i}{b_y}})^{M_{y,j}^*}$$

and

$$C_{i,2} = g^{-\hat{r}_i - sb_i}, C_{i,3} = \lambda_i''$$

$\mathcal{B}$ sends the challenge ciphertext $CT_1^* = \{C, C', \{C_{i,1}, C_{i,2}, C_{i,3}\}_{i \in [1, l^*]}\}$ to $\mathcal{A}$.

Phase2. $\mathcal{A}$ continues to perform a secret key query similar to **Phase1**.

Guess. $\mathcal{A}$ submits a guess $v' \in \{0, 1\}$. If $v' = v$, $\mathcal{B}$ outputs guess 0 which means

$T = e(g, g)^{a^{q+1} \cdot s}$. Otherwise $\mathcal{B}$ outputs guess 1 decides that T is a random element R in $\mathbb{G}_T$.

Therefore, the advantage of the $\mathcal{B}$ in solving the decisional q —parallelBDHE problem in a security game is

$$\begin{aligned} Adv_{\mathcal{B}} &= \frac{1}{2} (\Pr[\mathcal{B}(\vec{y}, T = e(g, g)^{a^{q+1} \cdot s}) = 0] + \Pr[\mathcal{B}(\vec{y}, T = R) = 0]) - \frac{1}{2} \\ &= \frac{1}{2} \left(\frac{1}{2} + \epsilon + \frac{1}{2} \right) - \frac{1}{2} \\ &= \frac{\epsilon}{2} \end{aligned}$$

Theorem 2. No polynomial time adversary $\mathcal{A}$ can win the trapdoor privacy game with a non-negligible advantage if DBDH assumption holds.

Proof. Suppose there is an adversary $\mathcal{A}$ which breaks the trapdoor privacy of our scheme with a non-negligible advantage ϵ_T , then we can construct an algorithm $\mathcal{B}$ to solve the DBDH problem. Let $\mathbb{G}$ be a group of prime order p with generator g and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear map. First, challenger $\mathcal{C}$ selects $\beta, \gamma, z \in \mathbb{Z}_p, v \in \{0, 1\}$ and an element $R \in \mathbb{G}_T$. We let $Z = e(g, g)^{\beta \gamma z}$ if $v = 0$. Otherwise, $Z = R$. Then $\mathcal{C}$ gives $(g, g^\beta, g^\gamma, g^z, Z)$ to $\mathcal{B}$. Now let $\mathcal{B}$ play the role of challenger in the following security games.

Setup. $\mathcal{B}$ announces the public key $pk_U = g^\beta, pk_O = g^\gamma$ with the implicit assumption that $sk_U = \beta, sk_O = \gamma$.

Phase1. $\mathcal{A}$ can polynomial query the following oracles:

Hash Oracle $\mathcal{O}_H$: $\mathcal{A}$ can ask the random oracle. $\mathcal{B}$ maintains a hash list

$\langle (pk'_{DO}, pk'_{DU}, w_i), h_i, e_i, c_i \rangle$ denoted as L_H . When

1. If the keyword w_i already appears on the L_H in a tuple $\langle (pk'_{DO}, pk'_{DU}, w_i), h_i, e_i, c_i \rangle$, then $\mathcal{B}$ responds with $H(w_i) = h_i \in \mathbb{G}$.
2. If the keyword w_i does not exist in the list L_H , $\mathcal{B}$ throw a random coin $c_i \in \{0, 1\}$ so that $\Pr[c_i = 0] = \sigma$ and σ will be determined later.

If $c_i = 0$, $\mathcal{B}$ computes $h_i = g^z \cdot g^{e_i} \in \mathbb{G}$;

If $c_i = 1$, $\mathcal{B}$ computes $h_i = g^{e_i} \in \mathbb{G}$, where $e_i \in \mathbb{Z}_p$ is randomly selected.

Then $\mathcal{B}$ adds the tuple $\langle (pk'_{DO}, pk'_{DU}, w_i), h_i, e_i, c_i \rangle$ to the list L_H , and returns $H(pk'_{DO}, pk'_{DU}, w_i) = h_i$ to $\mathcal{A}$.

Trapdoor Querie $\mathcal{O}_T$: $\mathcal{A}$ gives a tuple (pk'_{DO}, w_i) to ask about the trapdoor. $\mathcal{B}$ retrieves $\langle (pk'_{DO}, pk'_{DU}, w_i), h_i, e_i, c_i \rangle$ from list L_H .

If $c_i = 0$, $\mathcal{B}$ claims the failure and output $\perp$.

If $c_i = 1$, and hence $h_i = g^{e_i} \in \mathbb{G}$. $\mathcal{B}$ computes the trapdoor T_{w_i} as

$$\begin{aligned} T_{w_i} &= e(pk'_{DO}, pk_U)^{e_i} = e(H(pk'_{DO}, pk'_{DU}, w_i), pk_O)^{sk_U} \\ &= e(g^\gamma, g^\beta)^{e_i} = e(H(pk'_{DO}, pk'_{DU}, w_i), g^\gamma)^\beta \end{aligned}$$

Index Queries $\mathcal{O}_I$: $\mathcal{A}$ gives a tuple (pk'_{DU}, w_i) to ask about the index. $\mathcal{B}$ randomly chooses $t_i \in \mathbb{Z}_p$ and retrieves $\langle (pk'_{DO}, pk'_{DU}, w_i), h_i, e_i, c_i \rangle$ from list L_H .

If $c_i = 0$, $\mathcal{B}$ claims the failure and output $\perp$.

If $c_i = 1$, let $h_i = g^{e_i} \in \mathbb{G}$. $\mathcal{B}$ computes the index I_i as

$$I_i = (I_{1,i}, I_{2,i}) = ((pk_O)^{e_i} \cdot g^{t_i}, (pk_U)^{t_i}) = ((g^\gamma)^{e_i} \cdot g^{t_i}, (g^\beta)^{t_i})$$

Finally, $\mathcal{B}$ sends trapdoor T_{w_i} and index I_i to $\mathcal{A}$.

Challenge. $\mathcal{A}$ gives two equal length keywords (w_0^*, w_1^*) , with the restriction that (w_0^*, w_1^*) have not been queried to $\mathcal{O}_T$ nor $\mathcal{O}_I$. $\mathcal{B}$ selects a random bit $v \in \{0,1\}$ and generates the challenge trapdoor $T_{w_b^*}$ of keyword w_b^* and returns it to the $\mathcal{A}$.

If $c_0^* = 1$ and $c_1^* = 1$, $\mathcal{B}$ claims the failure and output $\perp$.

If $c_0^* = 0$ or $c_1^* = 0$, $\mathcal{B}$ selects a random bit $v \in \{0,1\}$ such that $c_v^* = 0$. $\mathcal{B}$ computes the trapdoor $T^* = Z \cdot e(g^\beta, g^\gamma)^{e_i^*}$. If $Z = e(g, g)^{\beta\gamma z}$, then $T^* = e(g, g)^{\beta\gamma(z+e_i^*)} = e(g^{\beta\gamma}, h_{v'})$. Otherwise, T^* is a random group element in $\mathbb{G}_T$.

Phase2. Same as **Phase1**, with the restriction $w \neq w_0^*, w_1^*$.

Guess. $\mathcal{A}$ will output a guess $v' \in \{0,1\}$. If $v' = v$, the $\mathcal{A}$ wins this game, otherwise $\mathcal{A}$ fails.

Probability Analysis. Now, we denote by *abort* the event that $\mathcal{B}$ aborts during the game. q_T and q_I express the query number of trapdoor oracle $\mathcal{O}_T$ and index oracle $\mathcal{O}_I$. There are two cases in which $\mathcal{B}$ aborts, as follows.

1. If $c_i = 0$ when simulating $\mathcal{O}_T$ and $\mathcal{O}_I$. Denote it by *abort*₁. The probability that *abort*₁ will not occur is $\Pr[\overline{\text{abort}_1}] = (1 - \sigma)^{q_T+q_I}$.
2. If $c_0^* = 1$ and $c_1^* = 1$ in the challenge phase. Denote it by *abort*₂. The probability that *abort*₂ will not occur is $\Pr[\overline{\text{abort}_2}] = 1 - (1 - \sigma)^2 = 2\sigma - \sigma^2$. Thence, the probability that $\mathcal{B}$ does not terminate the game is

$$\Pr[\overline{\text{abort}}] = \Pr[\overline{\text{abort}_1}] \cdot \Pr[\overline{\text{abort}_2}] = (1 - \sigma)^{q_T+q_I} \cdot (2\sigma - \sigma^2)$$

When $\sigma = 1 - \sqrt{q_T + q_I / (q_T + q_I + 2)}$, $\Pr[\overline{\text{abort}}]$ takes the maximum value

$$\Pr[\overline{\text{abort}}] = \left(\frac{q_T + q_I}{q_T + q_I + 2}\right)^{q_T+q_I/2} \cdot \frac{2}{q_T + q_I + 2}$$

which is approximately equal to $\frac{2}{(q_T+q_I)}$, and thus non-negligible. Conditioned on that $\mathcal{B}$ does not abort, if $\mathcal{A}$ succeeds in breaking the trapdoor privacy of our scheme, $\mathcal{B}$ also succeeds in telling $Z = e(g, g)^{\beta\gamma z}$ or a random element of $R \in \mathbb{G}_T$. Therefore, the probability that $\mathcal{B}$ succeeds in guessing the bit v (and thus solves the DBDH problem) is

$$\begin{aligned} \Pr[v' = v] &= \Pr[v' = v | \text{abort}] \cdot \Pr[\text{abort}] + \Pr[v' = v | \overline{\text{abort}}] \cdot \Pr[\overline{\text{abort}}] \\ &= \frac{1}{2}(1 - \Pr[\overline{\text{abort}}]) + \left(\frac{1}{2} + \epsilon_T\right) \cdot \Pr[\overline{\text{abort}}] \\ &= \frac{1}{2} + \epsilon_T \cdot \Pr[\overline{\text{abort}}] \end{aligned}$$

If ϵ_T is non-negligible, so is $|\Pr[v' = v] - \frac{1}{2}|$.

6 Performance analysis

In this part, we give theoretical and experimental analysis of the proposed scheme.

6.1 Theoretical analysis

(1) Capability. Here, we give the comparison between our scheme and several related works in terms of features (i.e. Keyword search, Fog computing, Multi-authority, etc.) in Table 1. Observe that, we can see that the schemes [10,22,29] do not have the function of keyword search. Only our scheme and scheme [29] are based on fog computing. With the exception of scheme [10] and our scheme, all users' attributes in the other schemes are not distributed by multiple authorities. That is to say, multiple authorities design improve the

Table 1. Features comparison with the main schemes.

Schemes	Keyword search	Fog computing	Multi-authority	Outsourcing encryption	Outsourcing decryption	Access control
[10]	×	×	✓	×	×	LSSS
[16]	✓	×	×	×	×	AND gate
[18]	✓	×	×	×	×	Access tree
[22]	×	×	×	✓	✓	LSSS
[24]	✓	×	×	×	✓	Access tree
[29]	×	✓	×	×	✓	AND gate
Ours	✓	✓	✓	✓	✓	LSSS

×: This scheme has this function.

✓: This scheme does not have this function.

<https://doi.org/10.1371/journal.pone.0207543.t001>

security of the key and reduce the computational pressure of a single authority. Moreover, our scheme and scheme [22] can provide outsourced encryption algorithm. But scheme [22] outsources their computational tasks to the corresponding service providers and does not address the latency response problem. The schemes [16,29] adopts AND-gate access policy and the schemes [18,24] uses a less computationally efficient tree access structure. Our scheme adopts efficient linear secret sharing (LSSS). Fortunately, only our scheme satisfies all properties which makes our scheme more suitable for cloud-fog computing system.

(2) Efficiency. In Table 2, we compare the computation cost of our scheme with the schemes [10,16,18,22,24,29] on the key generation, index encryption, trapdoor generation, search over ciphertext, DU-decryption. We mainly consider the time-consuming exponential operation e and the bilinear pairing operation p . In contrast, the time consumption of the remaining operations is negligible. It can be seen from Table 2 that the literature [10,22,29] does not support keyword search, so there is no computational cost in the index generation, trapdoor generation and search phases. Since our scheme is a multi-authority encryption scheme, the proposed scheme has lower efficiency in the key generation phase than other schemes, but our scheme protects the privacy of the user key and prevents the key from leaking. In the index generation phase, it is obvious that the schemes in [16] is less efficient than

Table 2. Comparisons of communication cost.

Schemes	KeyGen	Index	Trapdoor	Search	DU-Decrypt
[10]	$(2 S + 5)e + p$	×	×	×	e
[16]	$(2 S + 3)e$	$(\kappa + 2)e$	$(2 S + 1)e$	$e + (S + 1)p$	×
[18]	$(2 S + 4)e$	$(2\kappa + 1)e$	$(2 S + 4)e$	$ S e + (2 S + 1)p$	$e + p$
[22]	$(S + 8)e + 5p$	×	×	×	p
[24]	$(2 S + 4)e$	$3e + (\kappa + 1)p$	$(2 S + 1)e$	$2(S + 1)p$	$2p$
[29]	$3 S e + p$	×	×	×	$e + 3p$
Ours	$(4K + S_d)e$	$3\kappa e$	$3e + p$	$2p$	e

p : the pairing operation.

e : the group exponentiation operation in $\mathbb{G}$ or $\mathbb{G}_T$.

K : the total number of CAs in the system.

$|S_d|$: the number of attributes managed by the AA_d .

κ : the number of encrypted keywords in the index.

×: indicates that there is no corresponding algorithm in this scheme.

<https://doi.org/10.1371/journal.pone.0207543.t002>

our scheme. In addition, the computational complexity of the trapdoor generation and search algorithm are linear with the number of attributes. Our scheme is more efficient than other schemes. It only requires 3 exponential operations and 3 pairs of operations are independent of the number of attributes. In the decryption phase, the efficiency of scheme [10] and our scheme are much higher than that of scheme [18,22,24,29] and the decryption algorithm only needs 1 exponential operation. In general, our proposed scheme has higher search efficiency and lower cost of decryption.

6.2 Experimental analysis

In order to evaluate the practical performance of our scheme, our experiments use the Pairing-Based Cryptography (PBC) library [30]. The environment of the hardware runtime is Intel Core i5-3470 CPU @ 3.20GHz and RAM is 4.00GB. The software runtime environment is JDK 1.7.5, JPBC 2.0.0 and MyEclipse 10. In this section, we describe the efficiency comparison between our scheme and several related literatures [10,16,18,22,24,29]. For the sake of description, we assume the number of user attributes $|S| \in [10, 50]$ for the keygen, index, trapdoor and search algorithms, which the unified factor is described by the number of attributes. The time is given in milliseconds. From these sub-figures Fig 4(A)~4(D), we show that the number of attributes has an influence on the efficiency of the above four algorithms, respectively.

Fig 4. Performance comparison. (A) key generation time (B) index generation time (C) trapdoor generation time (D) search time.

<https://doi.org/10.1371/journal.pone.0207543.g004>

In Fig 4(A), we show the runtime of the key generation algorithm under different schemes. It can be seen that our solutions are better than all other solutions, and the running time of the schemes [16,18,24] is slightly higher. This is because the theoretical costs of KeyGen algorithm in aforementioned three schemes are $(2|S| + 3)e$, $(2|S| + 4)e$, $(2|S| + 4)e$, respectively. The scheme [29] requires the longest run time and the fastest growth rate. For example, when setting $|S| = n = 30$, the time required for these seven programs is 138.934ms, 118.566ms, 120.448ms, 151.836, 120.448ms, 185.444ms and 63.988ms, respectively. The growth of scheme [22] is relatively flat, and at $|S| = 10$, the time of key generation is the longest in all relevant literature.

Schemes in [10,22,29] do not have the function of keyword search, there are no index generation, trapdoor generation and ciphertext retrieval curve for the scheme of [10,22,29] in Fig 4(B)~4(D). In Fig 4(B) we show the time of index generation in the encryption algorithm. By changing the value of n from 10 to 50, we notice that the computational burden of our scheme is slightly higher than that of schemes [16] and [18]. However, since the encryption algorithm is a one-time cost, it does not affect the user's search experience. Therefore, its communication cost is acceptable in practical applications.

In Fig 4(C), we present the time cost of the trapdoor generation algorithm in all schemes. Schemes [10], [18] and [24] have only subtle differences in the trapdoor generation phase and the time spent is linearly increasing with the number of attributes. However, our scheme is a tiny constant that only needs $3e + p$ operations, regardless of the number of attributes.

Focusing on the search algorithm, we also tested the time spent in the ciphertext retrieval phase. In this experiment, the calculation cost of the scheme [18] and [24] were the highest, and the scheme [18] (or the scheme [24]) needs $|S|e + (2|S| + 1)p$ (or $(2|S| + 1)p$) operations. The computational cost of these three schemes [16,18,24] grows linearly with the number of attributes. While our scheme is optimal in all schemes. This is because the time of ciphertext search is independent of variable n , our scheme just needs $2p$ operations.

From Fig 4 we can see that the actual experimental simulation is completely consistent with the theoretical analysis. Therefore, our scheme is feasible and efficient in practical environment.

7 Conclusions

In this paper, we have presented a searchable encryption scheme based on cloud-fog computing, in which end users could greatly reduce the computational and storage burden by outsourcing part of the operation to the fog node. Specially, with the application scenarios named PHRs, the scheme enables patients to safely store PHRs shared with their doctor or family on a cloud server, while the patient's personal information remains confidential. Furthermore, our solution supports keyword search and fine-grained access control to further narrow down the search scope and avoid unauthorized user's access. Finally, our scheme is proven IND-sCP-CCA secure and trapdoor privacy secure. As part of our future work, we will continue to explore expressive search, fuzzy keyword search, multi-dimensional scope query or no central authority in the system, and so on. Meanwhile, we also need to further improve the efficiency of our system so that it can be applied to various programs.

Supporting information

S1 File. The runtime of cryptographic operations.
(DOC)

S2 File. The summary of the new notations.
(DOC)

Acknowledgments

This subject was funded by the National Natural Science Youth Fund No. 61303223, the National Natural Science Foundation of China No. 61572019. In addition, we would like to thank all anonymous experts for their valuable opinions and suggestions.

Author Contributions

Formal analysis: Jin Sun.

Methodology: Jin Sun, Shangping Wang.

Validation: Xiaojing Wang, Shangping Wang.

Writing – original draft: Xiaojing Wang.

Writing – review & editing: Xiaojing Wang, Lili Ren.

References

1. Dubarry E (2007) The Economic Interest of EMR (Electronic Medical Record). *American Chiropractor* 10: 70–78.
2. Endsley S, Kibbe DC, Linares A, Colorafi K (2006) An introduction to personal health records. *Fam Pract Manag* 13: 57–62.
3. Bonomi F, Milito R, Zhu J, Addepalli S. Fog computing and its role in the internet of things; 2012. pp. 13–16.
4. Stojmenovic I, Wen S. The Fog computing paradigm: Scenarios and security issues; 2014. pp. 1–8.
5. Khakimov A, Muthanna A, Muthanna MSA. Study of fog computing structure; 2018. pp. 51–54.
6. Goyal V, Pandey O, Sahai A, Waters B. Attribute-based encryption for fine-grained access control of encrypted data; 2006. pp. 89–98.
7. Ibraimi L, Asim M, Petkovic M. Secure management of personal health records by applying attribute-based encryption; 2009. pp. 71–74.
8. Li M, Yu S, Zheng Y, Ren K, Lou W (2013) Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption. *IEEE Transactions on Parallel & Distributed Systems* 24: 131–143.
9. Chase M. Multi-authority Attribute Based Encryption. *Theory of Cryptography*; 2007; Berlin, Heidelberg. Springer Berlin Heidelberg. pp. 515–534.
10. Yang K, Jia X, Ren K, Zhang B. DAC-MACS: Effective data access control for multi-authority cloud storage systems; 2013. pp. 2895–2903.
11. Rouselakis Y, Waters B. Efficient Statically-Secure Large-Universe Multi-Authority Attribute-Based Encryption. *Financial Cryptography and Data Security*; 2015; Berlin, Heidelberg. Springer Berlin Heidelberg. pp. 315–332.
12. Li D, Chen J, Liu J, Wu Q, Liu W. Efficient CCA2 Secure Revocable Multi-authority Large-Universe Attribute-Based Encryption. *Cyberspace Safety and Security*; 2017; Cham. Springer International Publishing. pp. 103–118.
13. Zhang K, Li H, Ma J, Liu X (2017) Efficient large-universe multi-authority ciphertext-policy attribute-based encryption with white-box traceability. *Science China Information Sciences* 61: 032102.
14. Boneh D, Di Crescenzo G, Ostrovsky R, Persiano G. Public Key Encryption with Keyword Search. *Advances in Cryptology—EUROCRYPT 2004*; 2004; Berlin, Heidelberg. Springer Berlin Heidelberg. pp. 506–522.
15. Zhao F, Nishide T, Sakurai K. Multi-User Keyword Search Scheme for Secure Data Sharing with Fine-Grained Access Control. *Information Security and Cryptology—ICISC 2011*; 2012; Berlin, Heidelberg. Springer Berlin Heidelberg. pp. 406–418.

16. Sun W, Yu S, Lou W, Hou T (2016) Protecting Your Right: Verifiable Attribute-based Keyword Search with Fine-grained Owner-enforced Search Authorization in the Cloud. *IEEE Transactions on Parallel & Distributed Systems* 27: 1187–1198.
17. Cui H, Wan Z, Deng R, Wang G, Li Y (2016) Efficient and Expressive Keyword Search Over Encrypted Data in the Cloud. *IEEE Transactions on Dependable & Secure Computing* PP: 1–1.
18. Miao Y, Ma J, Jiang Q, Li X, Sangaiah AK (2017) Verifiable keyword search over encrypted cloud data in smart city ☆. *Computers & Electrical Engineering* 65: 1–12.
19. Sun L, Su H, Zhu Z, Li Z. A Novel Attribute Based Keyword Search Scheme for Mobile Cloud Storage; 2017. pp. 229–234.
20. Cui J, Zhou H, Zhong H, Xu Y (2017) AKSER: Attribute-based Keyword Search with Efficient Revocation in Cloud Computing. *Information Sciences* 423.
21. Green M, Hohenberger S, Waters B (2011) Outsourcing the decryption of ABE ciphertexts. *Proceedings of the 20th USENIX conference on Security*. San Francisco, CA: USENIX Association. pp. 34–34.
22. Wang H, He D, Shen J, Zheng Z, Zhao C, Zhao M (2017) Verifiable outsourced ciphertext-policy attribute-based encryption in cloud computing. *Soft Computing* 21: 7325–7335.
23. Jiang ZL, Zhang R, Liu Z, Yiu SM, Hui LCK, Wang X, Fang J. A Revocable Outsourcing Attribute-Based Encryption Scheme. *Cloud Computing, Security, Privacy in New Computing Environments*; 2018; Cham. Springer International Publishing. pp. 145–161.
24. Li J, Lin X, Zhang Y, Han J (2017) KSF-OABE: Outsourced Attribute-Based Encryption with Keyword Search Function for Cloud Storage. *IEEE Transactions on Services Computing* 10: 715–725.
25. Sha FJ, Wei Y, Lin XN, Zhang QL, Wang HP (2017) Verifiable outsourced decryption of attribute-based encryption with constant ciphertext length. *Information Technology* 2017: 1–11.
26. Zhao Z, Wan J (2017) Verifiable Outsourced Ciphertext-Policy Attribute-Based Encryption for Mobile Cloud Computing. *Ksii Transactions on Internet & Information Systems* 11: 3254–3272.
27. Waters B (2011) Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. *Proceedings of the 14th international conference on Practice and theory in public key cryptography conference on Public key cryptography*. Taormina, Italy: Springer-Verlag. pp. 53–70.
28. Zhang P, Chen Z, Liang K, Wang S, Wang T (2016) A Cloud-Based Access Control Scheme with User Revocation and Attribute Update. Springer International Publishing. 525–540 p.
29. Zuo C, Shao J, Wei G, Xie M, Ji M (2016) CCA-secure ABE with outsourced decryption for fog computing ☆. *Future Generation Computer Systems* 78: 730–738.
30. Duquesne S, Lange T (2005) Pairing-based cryptography. *Mathiiscernetin* volume 22: 573–590.