Authentication and Access Control in e-Health Systems in the Cloud

Nafiseh Kahani
School of Computing
Queen's University, Canada
kahani@cs.queensu.ca

Khalid Elgazzar
School of Computer Science
Carnegie Mellon University, USA
elgazzar@cs.cmu.edu

James R. Cordy
School of Computing
Queen's University, Canada
cordy@cs.queensu.ca

***Abstract*—The opportunity to access on-demand, unbounded computation and storage resources has increasingly motivated users to move their health records from local data centers to the cloud environment. This change can reduce the costs associated with the management of data sharing, communication overhead and improve Quality of Service (QoS). Processing, storing, hosting and archiving data related to e-Health systems without physical access and control can exacerbate authentication and access control issues in this new environment. Therefore, convincing users to move sensitive medical records to the cloud environment requires implementing secure and strong authentication and access control methods to protect the data. This paper proposes a new information access method that preserves both authentication and access control in cloud-based e-Health systems. Our method is based on a zero-knowledge protocol combined with two-stage keyed access control. In each access request, based on the maximum rights of user, the minimum access is extracted. To establish secure connections between different entities in the system, a two-step combination of public key encryption and DUKPT is used. We analyze our scheme with respect to data confidentiality and resistance to common attacks on the network. Experimental results show that the proposed method tolerates a high number of concurrent authentication requests with a reasonable response time.**

***Index Terms*—Cloud computing, e-Health systems, access control, authentication, secure communication channel**

I. Introduction

Cloud computing enables convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort and service provider interaction. The cloud promotes availability and features five essential characteristics including: on-demand self-service, ubiquitous network access, location-independent resource pooling, rapid elasticity, and measured service. It offers computing services though three delivery models including software as a service $(SaaS)$, platform as a service $(PaaS)$, and infrastructure as a service $(IaaS)$; and four deployment models including public, private, community, and hybrid clouds [1].

Electronic health (e-Health) services provide efficient exchange of the patient's data between different entities including nurses, doctors, lab technicians, receptionists, and insurance companies. In e-Health, the data owner represents a content provider who can store and publish health records in the cloud environment for sharing. The cloud computing paradigm provides great opportunities to support flexible and controlled information exchange. However, authentication and access control issues in the cloud pose serious challenges that hinder the wide adoption of cloud-based e-Health services. For example, data exchange between communicating entities over the cloud exacerbates the security issues. Current authentication and access control mechanisms are not well-suited to the cloud environment. A strong and safe e-Health security solution that complies with Health Insurance Portability and Accountability Act $(HIPAA)$ policies is essential to protect patients' data from unauthorized access in cloud environments.

Several methods have been proposed to address the problems related to cloud security and outsourced data (e.g., [2], [3], [5]). Existing cryptographic approaches are sufficient if data owners want to just store their sensitive data on the cloud. However, e-Health services require data exchange between patients and their health-care providers. To perform safe and secure data exchange, we must address two concerns: 1) How can we authenticate the various communicating entities?, and 2) How can we mange access by these communicating entities?

To address the first concern, we need a secure and accurate authentication protocol to authenticate authorized entities, while preserving the privacy of users. This authentication protocol must grant access to users according to policies set by data owners. Regarding the second concern, traditional access control mechanisms (e.g., [6]) cannot be applied in this environment since they assume servers in the same domain are fully trusted and can enforce access control policies. However, in cloud environments, servers are outside of the user's trusted domain, and hence sensitive data must be protected from unauthorized access. These security concerns become more challenging in real-time health-care services.

The proposed approach enables patients and data owners to place their data on the cloud and perform secure and safe data exchange with health-care providers. In this paper, we propose a new and secure scheme that supports simultaneously both secure authentication and scalable fine-grained data access control, in terms of which authorized user has the access right to which types of health records. Our authentication approach is based on a zero-knowledge protocol to verify and maintain the anonymity of the user's identity. The method also allocates flexible access rights to individual users based on their rights

and intention of using the data. Our method provides data privacy through encryption techniques while enabling search over encrypted data. To establish secure communication between interacting entities, our approach uses a combination of a system public key and a secret session key generated by a Derive Unique Key Per Transaction (DUKPT) scheme [7]. In this method, the session key changes in each session, which strengthens the security of the connection. Hence, if a derived key is compromised, future and past transmitted data remain protected since the next and prior keys cannot be determined [8].

The main contributions of this paper are as follows.

- A new secure authentication framework for cloud provisioned e-Health services based on a zero-knowledge protocol.
- A new fine-grained data access control method based on user rights and their intention of using the data.
- An analysis of the proposed framework to evaluate its security and flexibility in terms of allowing the data owner to delegate most computation tasks to the participating entities, while maintaining confidentiality.

The remainder of this paper is organized as follows. Section II examines the related work. Section III presents the proposed method, considering both its design and implementation. Section IV analyzes the method with respect to different aspects such as data confidentiality. Section V discusses the simulation results of the authentication aspect of the method. We conclude the paper in section VI.

II. Related Work

Several methods have been proposed to address the problems associated with authentication and access control [4] in the cloud environment. One of these models is attribute-based access control (ABAC) [19], [2]. In this model, a set of meaningful attributes in the context of interest are associated with each data file. To enforce access policies, a public key component for each attribute is defined. Data files are encrypted using public key components corresponding to their attributes. An access structure for users is defined based on the secret keys they have. During file access, a user can decrypt a ciphertext if the data file attributes satisfy her/his access structure. In this model, the data owner has to re-encrypt all the data files accessible to a revoked user. To solve this problem, they use attribute-based encryption (ABE) combined with proxy re-encryption [20], and lazy re-encryption to support the process of user revocation. In this work, although the data owner delegates most of the computational operations to cloud servers, a cloud server is not able to learn about outsourced data. Unlike our method, this approach does not consider the need for management of data access policies.

Li et al. [21] propose patient-centric and fine-grained data access control through a multiple-owner settings model. In this work, patients as owners of health-care data can generate their own decryption keys utilizing ABE and then distribute them to their authorized users. In this way, multiple owners can encrypt data based to their own ways using different sets of cryptographic keys. Patients can determine the access structure of users by encrypting each record of data according to a relevant set of attributes. This approach imposes huge computational overhead on patients in terms of key distribution and data, and user management.

Barua et al. [22] propose a patient-centric access control (ESPAC) scheme which uses ciphertext-policy attribute-based encryption. This method determines different access rights for users according to their roles, and then assigns different attribute sets to them. Barua et al. [23] also suggest hybrid security policy for wireless body area networks (WBANs) with Quality of Services (QoS) for secure e-Health care system. In this method, cryptographic approaches such as public key cryptography are used for session key management and private key cryptography is used for data encryption in WBANs environment.

Luna et al. [24] suggest a hierarchical identity-based encryption that classifies users into upper and lower levels. The user at the upper level can share the secure cloud storage services with all the users at the lower level. A sender can specify several users at the lower level as the recipients of a file by taking the number and public keys of the recipients as inputs of a hierarchical identity-based encryption algorithm, which enables only the user at the upper level, as well as intended recipients, to decrypt the file using their own private keys.

Wang et al. [25] suggest a fine-grained access control scheme works on the hierarchical identity-based encryption combining (HIBE) system and the ciphertext-policy attribute-based encryption (CP-ABE). This method also has intensive computational overhead.

Luna et al. [24] propose a mandatory access control model to protect patients' meta-data. The model uses a Message Authentication Code (MAC) to protect the meta-data, and cryptography with fragmentation to protect data. This model shows that fragmentation after encryption can improve overall security, since to acquire access, attackers need to compromise more data files.

Sanka et al. [26] propose a method which uses the concept of capability-based model to show access structure of each user. The data owner outsources the encrypted data files and capability list to cloud servers. To protect the outsourced data, a combined approach of access control and cryptography is applied. A Diffie-Hellman key exchange model is also used for users to access the outsourced data securely from cloud servers.

Gao et al. [27] propose novel data access control (NDAC) which provides an improvement to the method in [26] to resist against two major attacks namely the man-in-the-middle attack and the replay attack. Unlike our method, in the NDAC scheme whenever a user decides to access the data, the data owner needs to be online.

Danwei et al. [28] propose an access control model based on usage control-based access (UCON) and negotiation mechanisms. A Negotiation module enhances flexibility of an access control model. In the case of mismatching between an access request and the access rules, negotiation module needs to run

to give another chance to the user instead of refusing her/his request. UCON is just a conceptual model, and no concrete specification is given for it.

Zhu et al. [29] propose a temporal attribute-based access control which uses a proxy-based re-encryption mechanism. In this work, each outsourced file is associated with an access policy on a set of temporal attributes. The data owner uses a temporal access policy to encrypt the data before outsources it to cloud servers. During access request from a user, the cloud server checks whether corresponding temporal constraints are satisfied in temporal access policy according to the current time. After that, the cloud server applies a re-encryption method to convert the encrypted data into another ciphertext with the embedded current time. The cloud server sends this ciphertext to the user. Finally, the authorized user can use her/his private key to decrypt the received ciphertext.

Fan et al. [30] suggest the DACAR platform for the e-Health cloud. This method uses cryptographic approaches to verify identities of users, and role-based policies to manage the access control to resources. The DACAR platform uses a single point of contact (SPoC), a rule-based information sharing policy syntax and data buckets hosted by a cloud infrastructure, to allow the secure usage of sensitive health-care data. However, the overhead of patients' processes is high, and the patient needs to be involved in the process of protecting and disseminating her/his medical records.

Unlike our method, most of these works focus on just authentication or access control management. To support a secure cloud-based e-Health service, it is essential to consider both authentication and access control issues.

III. Overview of the Proposed Scheme

In our approach, users are health-care practitioners, and patients and medical centers represent data owners who store encrypted health-care records in the cloud, to perform most of the computation-intensive tasks including authentication and fine-grained data access control. The architecture minimizes the computational overhead on the data owner's side as well as the time that data owners are required to be available online.

To begin, data owners encrypt and store their data on the cloud server without disclosing the contents. They also upload encrypted searchable indexes to the cloud server to enable multiple keyword searching over the encrypted data. Decryption keys are disclosed only to authorized users during the access control process. Each user must present a certified and unexpired token to the cloud server in order to access to the data. Thus, the user needs to send a request to a server managing the authentication and access control tasks to receive an access token. After that, the server verifies the identity of the user and issues a token that includes the required information according to his/her access rights and interests. In the case of receiving an authorized and valid token from the user, the cloud server performs search over the encrypted data, and sends the result to the user through a secure communication channel. The user can have access to the requested data with the proper decryption keys he/she received according to his/her rights.

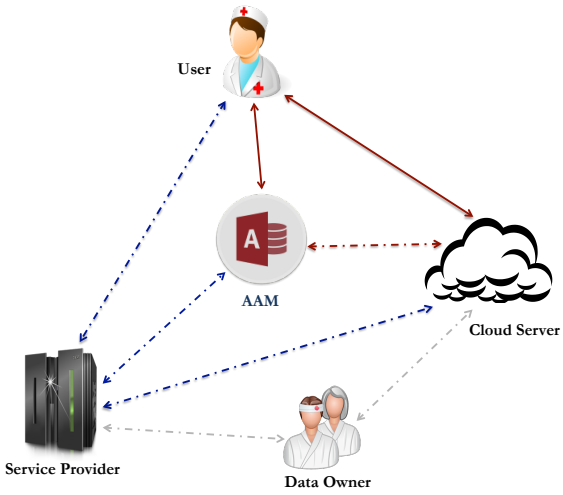

Fig. 1. Abstract view of the proposed scheme.

The proposed authentication model is based on a challenge-response protocol which keeps the sensitive user information confidential. Our philosophy of access control management is implemented in two phases. The first phase uses a static authorization method to determine the highest access rights of users. The second phase grants the user the minimum access permissions on the required data, according to the user's intention of access and the maximum rights determined by the first phase. We use the concept of intention to provide users with the minimum required access rights to limit threats stemming from unnecessary or over-provisioned rights that may cause unintended harm. Users can specify their intention through a received set of meaningful keywords related to their roles and job descriptions. These keywords provides varying levels of fine-grained access control to the required data. We use public keys and session keys using a DUKPT scheme to establish secure data exchange between various communicating entities. Figure 1 illustrates the architecture of the proposed approach. We discuss the different entities in our approach in the following subsections.

A. System Architecture and Parameters

The proposed system is composed of the following entities: data owners, service provider (SP), cloud server (CS), users (U), and an authentication and access control manager (AAM). The service provider sets and administrates access policies on behalf of data owners. The AAM is a trusted server which manages the process of authentication and access control. Data owners outsource their data files to an SP which determines and manages access control on their behalf. The CS provides the required resources to data owners to store and manage their data on the cloud. This server is typically managed by a cloud service provider (CSP) and can provide abundant storage capacity and computational power on demand.

TABLE I
SUMMARY OF NOTATIONS IN THE PROPOSED METHOD.

Notation	Description
Enc	Encryption function
Dec	Decryption function
PUBCS	Public key of CS
PUBAM	Public key of AAM
PRCS	Private key of CS
PRAM	Private key of AAM
SKF	Symmetric key for fragments
I	Indexes
T	Trapdoor
KSA	Secret session key during authentication process
KSD	Secret session key during data access process
p, q	Public prime numbers
F	Pseudo random function
r	Random commitment
si	Sequence index used for session key

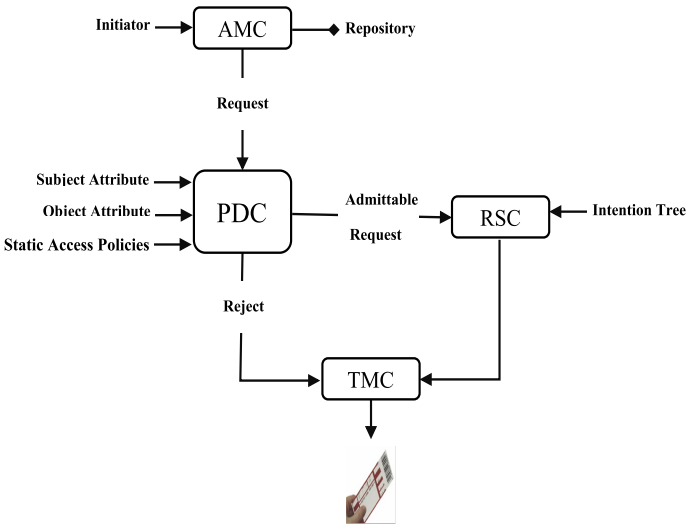

Fig. 2. Structure of authentication and access control manager (AAM) server.

To access health records shared by the SP, interested users submit a valid token with a search query request to the CS. The AAM authenticates the user's request and provides the proper decryption keys for valid requests. Users can then decrypt the data of interest with these decryption keys. In this model, neither data owners nor users are required to stay online, rather they connect when necessary to perform specific interactions. Although our model supports both reading and writing access rights, throughout this paper we assume that users have only read access rights for simplicity. We also assume that cloud servers are *honest* but *curious* [9]. This means that cloud servers honestly follow our proposed protocol, but they are also curious to analyze data and message flows received during the protocol to learn additional knowledge.

To perform the authentication step, we leverage a challenge-response protocol to prove the identity of the user. We use the Schnorr zero-knowledge identification protocol [10] as a challenge-response method. However, other zero-knowledge identification protocols are possible (e.g., [11]). We choose the Schnorr protocol since it satisfies three necessary properties: *completeness*, *soundness*, and *zero-knowledge* [12]. The completeness property means that if the statement is true, the honest verifier will be convinced of this fact by an honest prover. The soundness property shows that it is unlikely that a cheating prover would convince the honest verifier that a false statement is true. The zero-knowledge property ensures that no cheating verifier would gain any additional knowledge other than a statement is true [13].

Based on this protocol, the approach needs the following parameters. The SP selects a prime number p such that $p-1$ is divisible by another prime number q ($p = 2^{1024}$, $q \geq 2^{160}$), where p and q are public parameters. Another public value, β, is selected so that, $1 \leq \beta \leq p-1$, and $\beta = \alpha^{(p-1)/q} mod\ p$, where α is generator $mod\ p$. The SP also selects a number t, where $t \geq 40$, and $2^t < q$. The SP computes $v = \beta^{-ID} mod\ p$, where $0 \leq$ID$\leq (q-1)$ is the user's identity and v is known to both the AAM and the user. The user receives his/her ID during the registration with the system. Another system parameter is a pseudo random function $F : \{0,1\}^*$ used to generate session keys. F is shared between all entities of our model. Table I lists the attributes used in our method.

B. Data Structure

We suppose that data is represented in n fragments $R = \{R_1, ..., R_n\}$. These fragments do not overlap with each other and cover the whole data set. In addition, R is categorized into three levels of security as *secret*, *sensitive*, and *official* based on the administrative records, medical records and accounting records, respectively. Different encryption secret keys are used for each category. This classification and usage of different secret keys decreases the risk of data disclosure, especially during searching. Access levels are structured so that users with a lower access level cannot have access to data from higher levels without the explicit consent of the owner.

C. Authentication and Access Control Manager(AAM)

The authentication and access control manager (AAM) server is responsible for authentication and access control management. The AAM encompasses four components as follows. Figure 2 shows the structure of the AAM.

Authentication Management Component (AMC). The initiator is either a user or an organization that submits an access request to the AMC. An access request specifies the user's intention of access (i.e., why the user is interested to access a specific fragment of data). Authentication is performed based on the secret ID that users receive during their registration with the system. Users may enter private and sensitive personal information during registration that they do not want to reveal to the AAM. The challenge-response protocol can prove

the identify of the user anonymously without revealing any knowledge.

Policy Decision-maker Component (PDC). This component receives access requests from authenticated users. The PDC uses the access control policy defined by the SP to determine whether requests should be granted or denied. In section III-D5, we explain how these policies can be defined based on $Grant$ relationship. The PDC denies the request if it does not match the access rules. The PDC then communicates with the token manager component to issue a rejection message to the user. Otherwise, the PDC passes an admission request to the Rights Selection Component (RSC).

Rights Selection Component (RSC). This component receives access requests with admitted status from the PDC and generates the access rights that the user needs to perform the required actions, a trapdoor [9] to search over the encrypted documents in the cloud, and the required data fragments.

Token Management Component (TMC). This component generates valid tokens for authorized requests and sends rejection messages to users for denied access requests. Users who are granted access use these tokens to submit valid access requests to the CS.

D. Implementaion Details

In this section, we shed light on the different algorithms that we use for data storage, authentication, and data access control.

1) User Registration: During user registration, the SP generates a unique secret ID for each user, where $0 \leq ID \leq q-1$ is a long pseudo-random number generated by a universal one-way function. To leverage the zero-knowledge protocol, the SP generates a number $v = \beta^{-ID} mod\ p$ for each user to be used for authentication and token issuance.

2) Data Encryption: Based on the assumption that cloud servers are honest but curious, the SP (on behalf of data owners) encrypts the data before storing it in the cloud. The SP generates a symmetric data encryption key SKF for each data fragment category according to their level of sensitivity. The SP encrypts each fragment with its category-specific secret key and generates $C = \{C_1, ..., C_n\}$.

3) Data Search: Several methods have been proposed to securely search over encrypted data [9], [14], [15]. We use the method proposed by Chen et al. [9], which applies multiple keywords in the search request and returns data ordered by its relevancy to the keywords. This multiple keyword search produces accurate results. To enable searching over C, the SP builds encrypted searchable indexes I from F and sends both indexes and encrypted fragments to the CS. Fragments are also associated with a set of keywords. Users need to acquire a valid token to search the encrypted data with t given keywords. This access token includes a corresponding Trapdoor T [9] from AAM. This trapdoor is generated according to the keywords in the search request. These meaningful keywords are also shared with users to use to express their intentions.

4) User Authentication: Based on our proposed authentication method, the AMC component verifies the identity of the user without acquiring information about the user ID. The AMC caches all authenticated users and their corresponding information in an internal repository Rep. The AAM's public key and a session key are used to establish a secure connection between a user and the AAM. The user selects a Base Derivation Key $(BDKAM)$ to generate the session key and according to the following protocol to prove his/her identity to the AMC.

- The user chooses a random commitment r, $1 \leq r \leq q-1$ and $BDKAM$ as a secret key to generate the session key.
- The user calculates $x = \beta^r\ mod\ p$, and uses the public key of the AAM to encrypt x and $BDKAM$ and sends $Enc_{PUBAM}(x, v, BDKAM, si)$ to AAM, where si is a sequence index that is used to derive other session keys. It can be also used as a sequence number that is used to prevent attackers to run the replay attack.
- The AMC uses its private key to decrypt the received message $Dec_{PRAM}(Enc_{PUBAM}\ (x, v, BDKAM, si))$. The AMC uses v to find the information related to the user from Rep. In response, the AMC uses the function F and si to generate the session key $KSA_i = F(BDKAM, si + 1)$. This session key is used by the AMC to continue its communication with the user. The AMC selects a challenge random e, $1 \leq e \leq 2t < q$ along with a time-stamp $Time$ that helps to prevent forced delay of taking long time to respond. The AMC uses the KSA_i to encrypt the e and time-stamp and sends, $((Enc_{KSA_i}(e, Time)), si + 1)$ to the user.
- Based on the function F, $si + 1$ and BDK, the user generates KSA_i and decrypts the received message $Dec_{KSA_i}(Enc_{KSA_i}(e, Time))$. The user calculates $y = ID * e + r\ mod\ q$. The user also generates a new session key $KSA_{i+1} = F(KSA_i, si + 2)$ and sends $(Enc_{KSA_{i+1}}(y), si + 2)$ to AMC, during the time specified by $Time$, otherwise the session with AMC is rejected.
- The AMC uses $si + 2$ to calculate KSA_{i+1} to decrypt the received message from the user, and then computes $z = \beta^y * v^e\ mod\ p$. The AMC authenticates the user if $z = x$ and rejects otherwise. The output of this step is the request from a valid user that sends to the PDC of the AAM. The AMC also generates another session key $KSA_{i+2} = F(KSA_{i+1}, si + 3)$ that is used to encrypt the token issued by the TMC for the authorized user.

5) Access Control: Security policies specify and regulate the authorized actions that can be carried out in the system. In this section, we present our proposed method to regulate access control using security policies. First, we define the static access policy that determines the maximum rights of the users, and then we narrow down to the minimum rights that fit the user's intention to access the required data. One of the concerns in access control is that all users assigned to a role should not inherit the same permission. For example, a dentist and a neurologist with the same role as a doctor do not need to have access to the same health records. This model

can deal with this concern using a given permission granted to a specific user in a given role. In the following, we formalize the proposed access control method.

- ***Subjects and Roles***: In our model, a subject represents the user. A subject s_i is authorized in the system if $s_i \in S$, where $S = \{S_1, ..., S_n\}$ is a set of authorized subjects. $R = \{R_1, ..., R_m\}$ is the set of roles in an organization. Each subject s_i may have a set of roles $R_{s_i} = \{r_1, ..., r_j\}$, $1 \leq j \leq m$, where $R_{s_i} \in R$.
- ***Subject Attributes***: We use attributes to show the characteristics and capabilities of a subject. These attributes include role, job description, etc. Figure 2 shows how the PDC uses subject attributes in the decision-making process. Each subject is represented by a set of subject attributes $SAttr = \{SAtt_1, ..., SAtt_m\}$ where m is the total number of subject attributes.
- ***Object***: The entity objects are health records such as medical records, accounting records, and administrative records that are divided to n fragments. The object set is defined as $O = \{O_1, ..., O_n\}$. Based on the security level of objects, a user can have access to different objects, so the object set of each subject is defined as $S_i = \{O_1, ..., O_j\}$, $1 \leq i \leq n$ and $1 \leq j \leq n$.
- ***Object Attribute***: Attributes include important information about an object such as security level, type (e.g., medical records or accounting records), and so on. The object attribute set is defined as $OAttr = \{OAtt_1, ..., OAtt_m\}$, where m is the number of object attributes.
- ***Actions/Rights***: Rights are the set of actions that the subject has permission to perform on objects. The action set is defined as $a = \{a_1, ..., a_m\}$ and the right set as $\alpha = \{\alpha_1, ..., \alpha_n\}$ where m and n are the number of actions and defined rights in the system, respectively. Rights can include read and write. Actions are reading, writing, and consulting. Since in each organization, different actions need to different rights, so organizations define actions in terms of rights [17].

To define the static policies, we adopt the relationships *Employ* and *Define* from the method proposed by Kalam et al. [17] to define the *Grant* relationship.

- $Employ(org, S_i, R_j)$ means that the organization, org, employs subject S_i in role R_j, $1 \leq i \leq n$ and $1 \leq j \leq m$.
- $Define(org, a_i, \alpha_j)$ means that the organization, org, defines action a_i in terms of rights, $1 \leq i \leq n$ and $1 \leq j \leq m$, for instance the consulting action as a reading right.
- $Grant(S_i, org, R_j, \alpha_i, O_j)$ is based on the two aforementioned relationships *Employ* and *Define*. SP grants the right α_i on object O_j to the subject S_i with the role R_j, from the organization, org. This definition permits authorized users only to have access to specific health records.
- $Grant(S, \alpha_i, O_j, is\_condition)$. This relationship can be defined by the SP in the case that the owner wants to seek advice from a subject (e.g., a doctor, who is not employed by the organization). Moreover, based on our definition, a user who is granted access to an object with an *official* security level status cannot access an object with a *sensitive* security level status.

  There are emergency situations when a subject needs to access objects with higher security level. We use a condition term that defines the urgency situation or having consent from the data owner. Based on the consent and urgency situation, the $is\_condition(S_i, \alpha_i, O_j)$ function returns true.

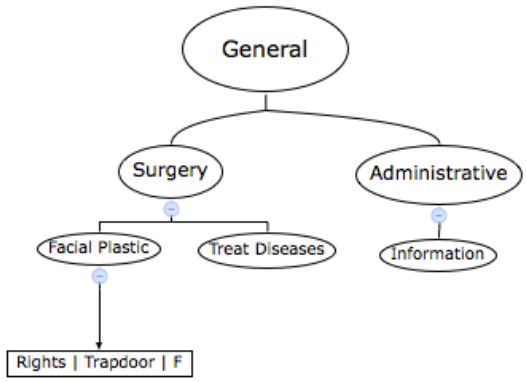

Fig. 3. Example intention tree.

Using the above relationships, access is granted or denied based on the output of the function *is_access* that we define as:

$$\forall S\ \forall O\ \forall \alpha\ is\_access(S, \alpha, O) =$$

$$\begin{cases} \text{True}, & \text{if } Grant(S, \alpha, O, is\_condition) \bigvee \\ & Grant(S, org, R, \alpha, O) \\ \text{False}, & \text{otherwise} \end{cases} \quad (1)$$

We use these relationships to determine the maximum rights of each subject. When the system receives an authenticated access request, it extracts the least rights that satisfy the request. To achieve this goal, we use the concept *"Intention"* to narrow down the user rights according to the intention per request. This requires the user to express her/his reasons for accessing the data. We use the hierarchical relationships [18] to represent all possible intentions. We organize intentions $I = \{I_1, ..., I_n\}$, in a tree structure, named Intention Tree (IT). Different paths in the tree represent various intentions from specialized to generalized ones. Figure 3 demonstrates an example of IT in e-Health systems. We build a separate IT for each data fragment. Each leaf node is linked to a record that determines the required rights for an intention, trapdoor, and the data fragment(s) of interest to this intention. The intention is expressed in a set of keywords $W = \{W_1, ..., W_n\}$, where n is the number of keywords that characterize the intention. We calculate the trapdoor from these keywords based the method proposed by Chen et al. [9]. Users use the meaningful keywords shared by the SP to express their intention. For

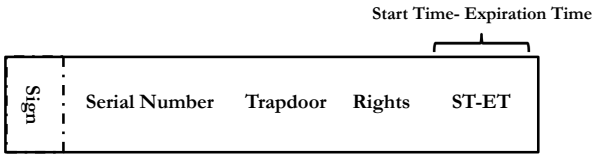

Fig. 4. Token architecture.

example, a health-care accountant can use the keywords such as financial records or company payroll to express his/her intention. The PDC uses static policies, object attributes, and subject attributes to check whether the user intention complies with the user's role(s), job descriptions and so on. If match is true, the PDC passes the authenticated access request, including the intention to the RSC. To determine the least access rights, RSC based on the user's information and intention processes the IT and generates a token.

Figure 4 shows the structure of tokens. A token includes the following:

- Serial number: To monitor the actions of users and for non-reputation processes, user access is logged. To save and search users' log easily, a serial number is assigned for each user. The AAM and CS use the serial number to manage the user's log.
- Token rights: The rights of the user holding the token.
- Trapdoor: The trapdoor created from the search keywords.
- Start time: The time at which the token becomes valid.
- Expiration time: The time at which the token becomes invalid.
- Signature: A digital signature to ensure the integrity of the token contents. The TMC uses the private key $PRAM$ to sign the token. The signature is also used to verify the start time and expiration time to ensure the token validity period. The TMC uses the session key KSA_{i+2} to encrypt both the token contents and the user decryption keys and sends $((Enc_{KSA_{i+2}}(token, decryptionkeys(SFK))), si + 3)$ to the user. The user uses the decryption key to decrypt the health records.

6) Data Access Control: The user can send the received token as the search right through a secure channel to the cloud provider hosting the data. To establish a secure communication channel between the user and CS, the user selects a secret key $KSCS$ as a session key. The user then sends $Enc_{PUBCS}(token, KSCS, N)$, where N is a sequence number to prevent replay attack. The CS decrypts the message with its private key and verifies the token by applying the system public key on the token's signature. The CS also verifies that the token is timely valid. The CS then searches the encrypted data with the trapdoor included in the token. The CS uses the session key received from the user to encrypt the search result and sends $Enc_{KSCS}(Data, N + 1)$ to the user. The users decrypts the message with the session key $Dec_{KSCS}(Enc_{KSCS}(Data, N+1))$. The user checks $N+1$ and uses the decryption keys received from AAM to decrypt the data.

Figure 5 summarizes the data access control process in our method.

E. Algorithm

Each user who wants to access health records shared by the SP sends its registration request to the SP. The SP generates a unique secret ID for the user. The user then submits an access request to the AAM to receive a valid token for accessing the health records. The AMC component uses a zero-knowledge identification protocol to authenticate the user's request. Based on the user's identity, the PDC component uses the access control policy defined by the SP to determine the minimum rights of the user. The PDC then communicates with the RSC to issue a valid token along with the proper decryption keys for the authenticated user. Algorithm 1 shows the process of generating a valid token for the user. In the next step, the user sends the received token to the CS. The CS verifies that the token is timely and valid and then searches the encrypted data with the trapdoor included in the token and sends the results to the user. The user can then decrypt the received health records using the decryption keys from the previous step.

Algorithm 1: Algorithm for generating a valid token.

Input : *user request*

$rights = ''''$
if *Authentication(user request)==TRUE* **then**
  $rights = MaximumAccess(user)$
  $access = TRUE$
  if *access==TRUE* **then**
    $token = MinimumAccess(rights, intention)$
  end
  else
    $error = IssueError(rejectedmessage)$
  end
end
else
  $access = FALSE$
  $error = IssueError(unauthorizeduser)$
end

Output: *token*, *error*

IV. Features and Security Risk Analysis

Data security is one of the biggest concerns in cloud computing. In this section, we analyze the proposed method from different security perspectives including data confidentiality and resistance to various security attacks.

A. Data Confidentiality

Our scheme uses a standard symmetric-key algorithm (e.g., AES or DES) to store data in an encrypted form in the cloud. Therefore, our scheme intrinsically inherits its security

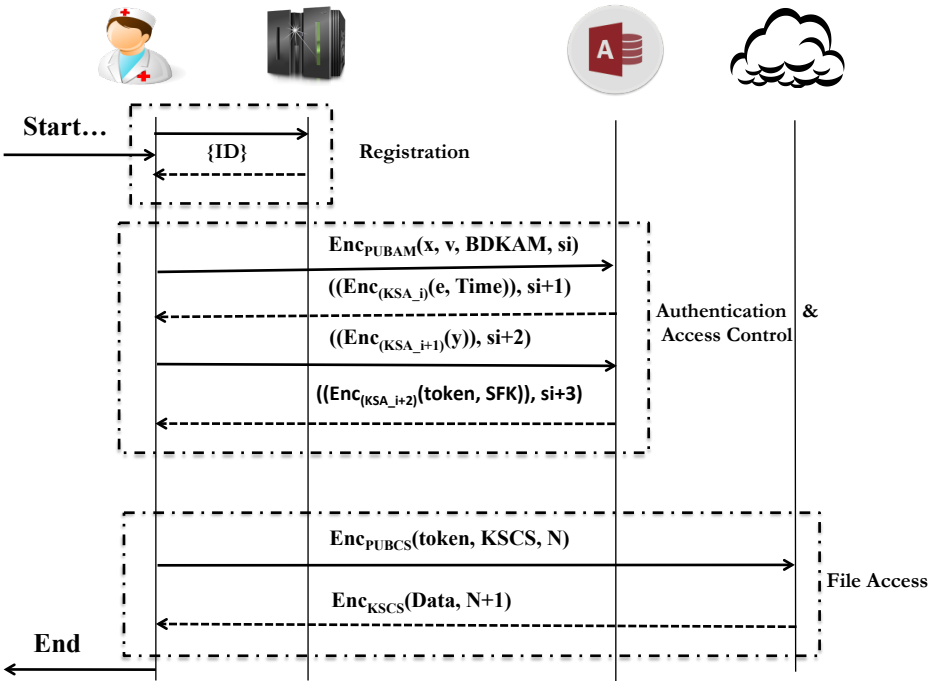

Fig. 5. Data control access procedure.

strength from the security of the symmetric-key algorithm. The decryption keys are only shared with the authorized users, so CS cannot acquire any information about the outsourced data or keys. Furthermore, the communication between entities are carried out in an encrypted form, so adversaries cannot have access to the transmitted messages. For instance, to exchange the decryption keys between AAM and users, a public key and a session key are applied to protect the disclosure of transmitted messages. For data access, a second encryption is applied to increase the security of the outsourced data. In addition, our proposed scheme does not disclose extra information to CS. Thus, the CS is unable to derive data or user access right information, since everything is encrypted, from data to trapdoors. The classification of fragments to various security levels adds another layer of fortification to secure the data.

B. Fine-grained Access Control

In the proposed scheme, we apply a two-step access control. The first step limits the rights of users based on their roles. The second step then limits the user rights based on the intention of his/her task. For example, an accountant user wanting to prepare a financial statement for a specific month does not need to have access to all of the financial/accounting records of the patient. These two steps result in making access decisions based on the minimum required rights of each user. Moreover, changing the trapdoor results in different file access rights. Therefore, the data owner is able to define and enforce a flexible access policy for each user.

C. Non-repudiation

The proposed method logs all user actions to support non-repudiation.

D. Authentication and User Privacy

A unique ID is assigned to the user upon registration. The communication between the user and the AAM is authenticated based on this ID with the challenge-response protocol. The security of authentication protocol relies on the complexity of discrete logarithm problems, which means that the attacker needs to solve the discrete logarithm problem, which is believed to be hard. Hence, the identity of users is preserved as the zero-knowledge protocol does not reveal the user ID, as shown in Figure 5.

E. Withstanding Security Attacks

Our approach is well-designed to withstand the various security attacks. In the following, we elaborate on how our scheme avoids and resists common network security attacks.

- *Replay Attack:* All communications between parties in our model are encrypted. Therefore, attackers cannot easily intercept any message and replay it. Since our authentication and authorization method also uses a sequence index si, our method is highly immune to replay attacks. The time-stamp of tokens and timeout period ensures the validity and freshness of the requested message. In addition, the sequence number N prevents a replay attack during data access, so attackers cannot replay outdated data to the user.

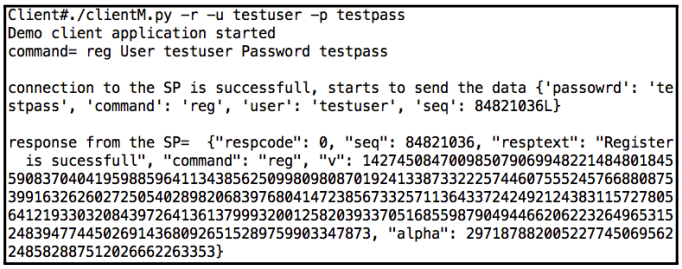

Fig. 6. User registration.

Fig. 7. User authentication.

- *Man-in-the-Middle Attack:* In this attack, the attacker between two communicating parties intercepts transferring messages and alters or injects new contents. In our scheme, attackers cannot intercept any message due to encryption. The combination of public key encryption and session key prevents the man-in-the-middle attack. Only the AAM can decrypt the user's access request using its own private key and only the target user can decrypt the response coming from the AAM using the session key. Hence, the communication between users and both the AAM and the CS is safe.
- *Brute-force Attack:* The session key between the user and the AAM is changed in each session by employing the concept of DUKPT. Therefore, the brute-force attack is almost impossible.
- *Denial-of-Service Attack:* During the authentication process, the AAM determines the value of $Time$ which means that the user has to respond during the specific time frame which effectively prevents a forced delay in response.

V. Performance Evaluation

We built a prototype to test and evaluate the proposed method. The prototype focuses on the authentication and token management process. The prototype is developed in Python, and uses the JSON format to exchange messages between various entities in our system. Our prototype mainly includes four entities: users U, a cloud server CS, a service provider SP, and an authentication and access control manager AAM. We used three Amazon EC2 instances of the type *t2.medium* for the CS, SP, and AAM. User requests were initiated from local machines in our lab. As shown in Figures 6 and 7, the authentication process is done in the following steps:

- The user runs the client application and sends a request to the SP to receive the authentication protocol parameters. The SP verifies the user information and then sends the requested information (e.g., v). The user name is the ID which the user receives during the registration. The client application saves the received protocol parameters on its system (Figure 6). The information related to the user ID and password are in clear format for debugging and demonstration purposes.
- During the authentication process, the user sends the authentication request to the AAM. The AAM sends the challenge to the user. The user uses the information which received in the previous step to calculate the challenge and respond to it. The AAM verifies the response and sends a signed token/ticket to the user (Figure 7). Any authentication request with invalid data is denied.

In the above steps, the user does not need to send his/her credentials to the AAM server. In addition, all exchange data is encrypted based on the public key and DUKPT scheme.

We conducted experiments to evaluate the throughput of our system under varying load testing. We submit concurrent requests and measure the average response time (i.e., the time to complete those requests). A bash script generates concurrent requests from different users to the AAM and we instrumented our prototype to report the response time. Figure 8 shows the average response time under varying numbers of concurrent requests. Experimental results show that our system (with the current setup) can accommodate up to 100 concurrent authentication requests with a reasonable response time of $2.16s$.

In Figure 8, the challenge request time shows the time that the server takes between receiving the request and generating the challenge request based on user information. Challenge validation shows the time that the SP needs to verify the received challenge request from the user. The session time comprises the entire user-perceived time from sending a request until receiving a response. This time includes network time, challenge request, challenge response and challenge validation time.

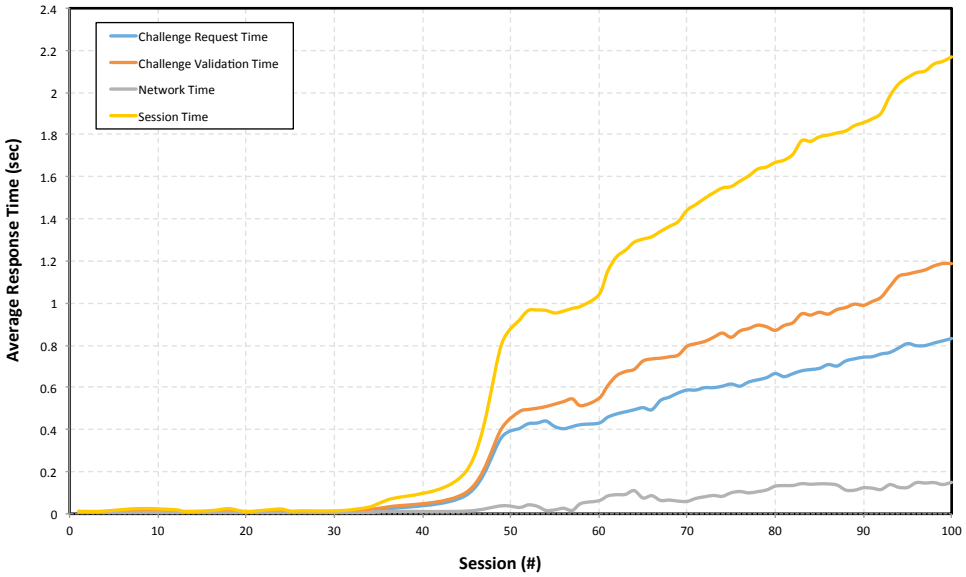

Fig. 8. Average response time of concurrent requests.

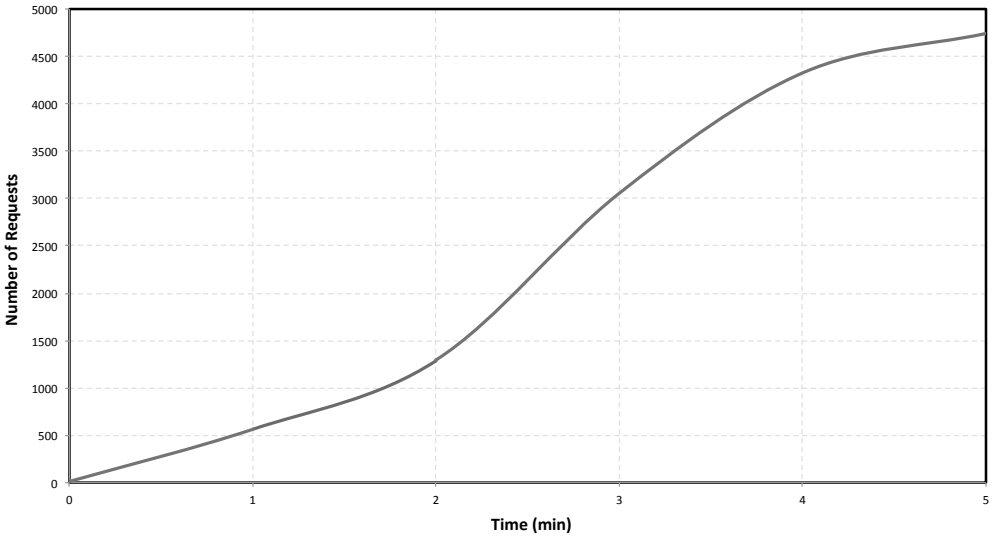

Fig. 9. System availability.

We also measure the breakdown of the response time to show the contribution of each process to the total execution time. Figure 10 shows the three major components of the total response time. We observed that the *validation operation* (i.e., the process of verifying the user response) represents the highest percentage of the overall execution time and the *challenge request* (i.e., the step of sending challenge for the user) is the second highest contributor. Network latency represents a very small component of the request response time.

Regarding system availability and AAM performance in terms of system throughput and varying workload, Figure 9 shows the increasing number of requests in minutes. It shows that the AAM is always responsive and can manage $13,984$ transactions over 5 minutes. At minute 5, the AAM can respond to $4,742$ requests simultaneously. We can push the knee of the curve forward by choosing a higher VM type or adding more instances of the same type to increase the system capacity.

VI. Conclusion

In this paper, we propose a new method that simultaneously achieves authentication and fine-grained access control. In our model, the data owner has a set of health records to publish on cloud servers for sharing. To keep user's data confidential against malicious users and semi-trusted cloud servers, encrypted data are stored in the cloud. To decrease heavy computational and communication overhead on data owners, most of the process of authentication and access control is

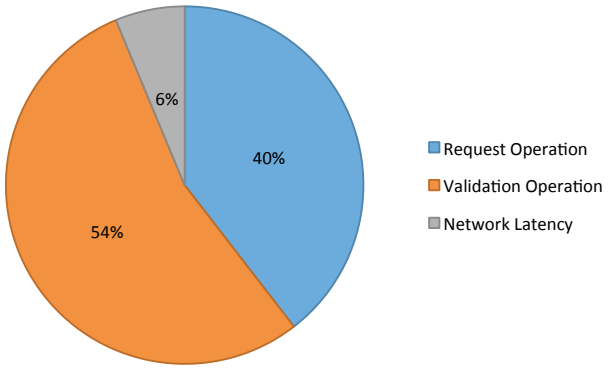

Fig. 10. Percentage of response time based on the session time.

delegated to an AAM, which is responsible for authentication, key distribution and access control management. The authentication process, based on a zero-knowledge method, determines authorized users. The access control method determines user privileges to access data. Our method guarantees data confidentiality since the cloud servers are unable to access the plaintext of any data file. Our authentication, user privacy, and fine-grained access control show high resilience to withstand common network attacks.

REFERENCES

[1] National Institute of Standards and Technology, Computer Security Resource Centre. Available: http://csrc.nist.gov

[2] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," *in IEEE Proceedings, INFOCOM*, pp. 1-9, March 2010.

[3] L. Yibin, W. Dai, Z. Ming, and M. Qiu, "Privacy protection for preventing data over-collection in smart city," *IEEE Transactions on Computers*, vol. PP, pp. 1, 2015.

[4] A. Majumder, S. Namasudra, and S. Nath, "Taxonomy and classification of access control models for cloud environments," *in Continued Rise of the Cloud*, pp. 23-53, 2014.

[5] N. Kahani, and H. R. Shahriari, "An approach for providing privacy and access control in outsourced databases," *14th International CSI Computer Conference*, 2009.

[6] D. Ferraiolo, and D. Kuhn, "Role-based access controls," *in Proceedings of the 15th National Computer Security Conference*, pp. 554-563, 1992.

[7] American National Standards Institute. ANSI X9.24-1:2009 Retail Financial Services Symmetric Key Management Part 1: Using Symmetric Techniques, 2009.

[8] E. Brier, and T. Peyrin, "A forward-secure symmetric-key derivation protocol," *in Advances in Cryptology-ASIACRYPT*, pp. 250-267, 2010.

[9] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Transaction on Parallel and Distributed Systems*, vol 25, pp. 222-233, 2014.

[10] C. P. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, vol 4, pp. 161-174, 1991.

[11] U. Feige, A. Fiat, and A. Shamir, "Zero-knowledge proofs of identity," *Journal of Cryptology*, vol. 1, pp. 77-94, 1988.

[12] U. Maurer, "Unifying zero-knowledge proofs of knowledge," *in Progress Cryptology AFRICACRYPT*, pp. 272-286, 2009.

[13] J. T. McCall, "Zero Knowledge compilers," *in UMM CSci Senior Seminar Conference*, pp. 19, December 2013.

[14] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," *in IEEE Proceedings, INFOCOM*, 2014.

[15] W. Sun, W. Lou, Y. T. Hou, and H. Li, "Privacy-preserving keyword search over encrypted data in cloud computing," *in Secure Cloud Computing*, pp. 189-212, 2014.

[16] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," *in Proceedings of ACM Conference on Computer and Communications Security*, pp. 965-976, 2012.

[17] A. A. Kalam, R. E. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, and G. Trouessin, "Organization based access control," *in IEEE 4th International Workshop*, pp. 120-131, June 2003.

[18] J. W. Byun, E. Bertino, and N. Li, "Purpose based access control of complex data for privacy protection," *in Proceedings of the 10th ACM Symposium on Access Control Models and Technologies*, pp. 102-110, June 2005.

[19] B. Cha, J. Seo, and J. Kim, "Design of attribute based access control in cloud computing," *in Proceedings of International Conference on IT Convergence and Security*, pp. 41-50, 2011.

[20] R. Canetti, and S. Hohenberger, "Chosen-ciphertext secure proxy re-encryption," *in CCS, ACM*, pp. 185-194, 2007.

[21] M. Li, S. Yu, K. Ren, W. Lou, "Securing personal health records in cloud computing: Patient-centric and fine-grained data access control in multi-owner settings," *in Proceedings of the 6th International ICST Conference on Security and Privacy in Communication Networks*, pp. 89-106, September 2010.

[22] M. Barua, X. Liang, R. Lu, and X. Shen, X., "ESPAC: Enabling security and patient-centric access control for eHealth in cloud computing," *International Journal of Security and Networks*, vol. 6, pp. 67-76, 2011.

[23] M. Barua, M. S. Alam, X. Liang, X. and Shen, "Secure and quality of service assurance scheduling scheme for wban with application to ehealth," *in IEEE Conference on Wireless Communications and Networking, QuintanaRoo*, pp. 1-5, 2011.

[24] J. Luna, M. Dikaiakos, M. Marazakis, and T. Kyprianou, "Data-centric privacy protocol for intensive care grids," *in IEEE Transaction on Information Technology in Biomedicine*, vol. 14, pp.1327-1337, 2010.

[25] G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," *in Proceedings of the 17th ACM Conference on Computer and Communications Security*, pp.735-737, 2010.

[26] S. Sanka, C. Hota, and M. Rajarajan, "Secure data access in cloud computing," *in IEEE 4th International Conference on Internet Multimedia Services Architecture and Application (IMSAA)*, pp. 1-6, 2010.

[27] X. Gao, Z. Jiang, R. Jiang R, "A novel data access scheme in cloud computing," *in Proceedings of the 2nd International Conference on Computer and Information Applications*, pp. 124-127, December 2012.

[28] C. Danwei, H. Xiuli, and R. Xunyi, "Access control of cloud service based on ucon," *in Cloud Computing*, pp. 559-564, 2009.

[29] Y. Zhu, H. Hu, G. J. Ahn, D. Huang, and S. Wang, "Towards temporal access control in cloud computing," *in IEEE Proceedings, INFOCOM*, pp. 2576-2580, March 2012.

[30] L. Fan, W. Buchanan, C. Thummler, O. Lo, A. Khedim, O. Uthmani, A. Lawson, and D. Bell, "DACAR Platform for eHealth services cloud," *in Proceedings of the 4th International Conference on Cloud Computing*, pp. 219-226, July 2011.