Secure Cloudlet-Based eHealth Big Data System With Fine-Grained Access Control and Outsourcing Decryption from ABE

Kittur Philemon Kibiwott[1], Zhang Fengli[1], Omala A. Anyembe[2], Daniel Adu-Gyamfi[3]

(Corresponding author: Kittur Philemon Kibiwott)

School of Information and Software Engineering, University of Electronic Science and Technology of China[1]

No. 2006, Xiyuan Ave, West Hi-Tech Zone, Chengdu, Sichuan 611731, China

(Email: phkibiwott@gmail.com)

School of Computer Science and Engineering, University of Electronic Science and Technology of China[2]

University of Energy and Natural Resources, Ghana[3]

(Received June 19, 2017; revised and accepted Sept. 27, 2017)

Abstract

Integration of cloud computing and mobile computing with the proliferation of big data is making remarkable strides in the health care industry. Aside the benefits accrued from adopting this technologies, there are myriad of challenges to overcome such as confidentiality of data outsourced to the cloud, integrity of stored data, wide area network (WAN) latency delays, and the resource constraints of the mobile devices. In this paper we propose a Cloudlet-Based eHealth Big Data System with Outsourced Decryption (CBe-BDS-OD) to address the above challenges. To accommodate mobile devices with limited resources, the computation power is borrowed from the cloudlet server securely. Security analysis demonstrates that our scheme is secure. In addition, our performance approach through theory analysis and experimental simulation indicates a substantial improvement in computation efficiency by 99% and therefore the scheme can be deployed in resource-constrained mobile devices.

Keywords: Big Data; Cloud Computing; Cloudlet; eHealth; Outsourcing; Resource-constrained Device

1 Introduction

Big data describes a massive volume of both structured and unstructured data that are beyond the processing capability of traditional software and database techniques. Big data is grouped into four (4) major features commonly known as "4 Vs of Big data" thus Volume "scale of data", Velocity "analysis of streaming data", Variety "data in different forms" [10] and Veracity "uncertainty of data"[IBM]. The fifth other feature include Value "discoverable behavior of data" [ORACLE]. The high volume eHealth data is generated by different sources which include biometric devices, networked sensors, RFID, mobile devices (*i.e.* with Bluetooth and GPS etc) and others such as hospital systems, for instance, the adoption of Electronic Health Record (EHRs) and Electronic medical records (EMRs) [26, 32]. These are the major sources of eHealth big data. While big data has benefits attached to, it requires a large computation and storage [33] capacity, with which resource-constrained devices (*i.e.* with less storage, battery life, computation power) [16] like mobile smartphones do not have and hence leading to the leveraging of cloud to store such volume of data [23, 30].

Due to elastic scaling provided by the cloud [18], now big data can be stored [33] an edge which cloud has over traditional storage. This means the organization's data is outsourced to a third party on utility costing basis [22]. But cloud has notable limitations including requirement of fast, reliable internet connectivity and high latency [29]. For real time applications like in healthcare, the adoption of cloud computing achieves less because its located far away from the mobile users and incurs long WAN latency delays making inefficient [1, 23, 30] and therefore hurts the mobile 'thin clients'. The cloudlet concept was introduced in [29] to solve these issues. A cloudlet which is also referred to as mini cloud, can be considered as a "data center in a box" whose objective is to bring the cloud closer to the user [30] as shown in Figure 1. It is a resource-rich server with Internet access that is well connected to mobile devices via a high-speed local area network (LAN) [23]. Mobile devices can migrate expensive computations to the cloudlet stationed on discoverable, localized, stateless servers running on single or multiple virtual machines (VMs) [1]. As a result, it preserves mo-

bile device battery, provides a more powerful computation, enhances flexibility and mobility [16]. During task migration, mobile device does not need to communicate with the Cloud directly instead it communicates with the closest cloudlet [29].

Many works have been proposed to overcome the storage limitations of massive data, where data is now outsourced to the cloud. However, practical adoption of those techniques, poses security and privacy challenges [15, 22, 25]. To overcome these, the data needed to be shared are encrypted before they are uploaded to the cloud, and fine-grained access control should be enforced [31]. An Attribute Based Encryption technique proposed by Sahai and Waters [28] enables the data owners to encrypt their data such that only end users that satisfy given criteria can, perhaps, succeed in decrypting the data. In ABE scheme [28], private key can decrypt a given ciphertext only if the associated attributes and access policy tally. There are two flavours of ABE schemes: Key-Policy ABE (KP-ABE) [13] and Ciphertext-Policy ABE (CP-ABE) [2, 31]. In KP-ABE scheme, each ciphetext are labeled with sets of descriptive attributes and the access policy of this attributes are associated with end user's private key.

Decryption is realized only if the attributes on the ciphertext satisfy the access policy of the user's private key [13]. While in CP-ABE scheme, each ciphertext is associated with an access policy, and every end user's private key is associated with a set of descriptive attributes. To rightly recover the message, the attributes in the user's private key need to satisfy the access policy [2, 31]. ABE is one of the powerful cryptographic tool for realizing fine grained data access control in the cloud storage system [24, 28]. However, with the majority of ABE schemes, the major drawback is inefficiency since the size of ciphertext and decryption overhead (number of pairing operations) grow with the complexity of the access policy [17]. This becomes a bottleneck when ABE runs on the resource-constrained device's applications for example on smartphones applications [6, 21, 25].

To minimize the workload operations on the end user's side while executing decryption algorithm, Green *et al.* [11] proposed a scheme where expensive computation operations is offloaded to the third-party. In this scheme, a key blinding technique (*i.e.* transformation key) TK is sent to the third-party (proxy) for translation of any ciphertext CT satisfied by end user's attributes or access policy into a simple ciphertext CT'. The end user incurs minimal overhead to recover plaintext from transformed ciphertext CT' [14, 19] using the retrieving key RK. While carrying out this, the proxy cannot learn any information about the original plaintext. In [5], Chase proposed a multi-authority ABE scheme which is secure against selective ID security model. To avoid a single authority issuing keys which can lead to key compromise, decentralized key-policy attribute-based encryption with privacy preserving was proposed in [12] and decentralized CP-ABE with fully hidden access structure was proposed in [27]. The scheme proposed in [11] provides fine-grained access control solution to the resource-constrained devices such as mobile phones in the cloud. However, in our scheme we introduce a new architecture where cloudlet is stationed between mobile device user and cloud as shown in Figure 1. This means the security infrastructure has to be modified also. In our work each mobile user with attribute list $\boldsymbol{F}$ is associated with a private key/decrypt$_{out}$ key while the cloudlet is labeled with access policy $\mathbb{P}$ which describes the rightful users of the data.

The main contribution of this paper can be summarized as follows:

1) Our CBe-BDS-OD proposed scheme, introduces an efficient encryption for a cloudlet that achieves confidentiality, collusion resistance and integrity of data based on ciphertext policy-attribute based encryption.

2) To minimize the pairing load operations on end user's side with resource-constrained devices and thereby reducing expensive computation, we adopted server-aided transformation where the computational processing power is borrowed from the server by the mobile user. The mobile cloudlet server provided with blinding key, transforms the complex ciphertext into a simple one. In the process of performing this, the mobile cloudlet server cannot reveal anything about the underlying plaintext.

3) We have implemented our scheme to evaluate the performance. The results in our protocol with server-aided decryption demonstrate a substantial improvement in computation efficiency by 99% at the end user's side compared to its counterpart and therefore can be deployed in resource-constrained mobile devices.

The rest of this paper is organized as follows: In Section 2 we give the preliminaries related to our proposal. In Section 3 we provide our scheme proposal which is IND-AND-sAS-CCA2 secure for CP-ABE with outsourced decryption for eHealth big data in cloudlet computing, we also give the IND-AND-sAS-CCA2 security model. In Section 4 we give concrete construction for our scheme. Thereafter we give security proof for our proposal defined in our proposed IND-AND-sAS-CCA2 security model in Section 5 and in Section 6 we give experiment and analysis results. Lastly we give our conclusion.

2 Preliminaries

2.1 Bilinear Maps

Let $\mathbb{G}$ and $\mathbb{G}_\tau$ be two multiplicative cyclic groups of prime order p. Let g_1, g_2 be generators of $\mathbb{G}$ and e be a bilinear map; e: $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. Bilinear map e has the following properties:

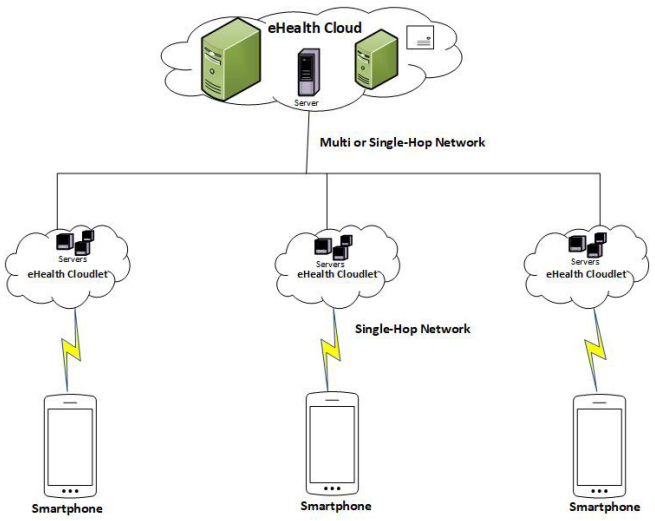

Figure 1: Three-tier architecture; Mobile device-eHealth; Cloudlet-eHealth cloud

1) *Bilinearity* : $\forall g_1, g_2 \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p^*$ we have $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab} = e(g_1^b, g_2^a)$.
2) $Non - degeneracy : e(g_1, g_2) \neq 1$.
3) *Computability*: There is an efficient algorithm to compute $e(g_1, g_2)$.

2.2 Definition of Access Structures

Our proposed Protocol is based on AND-gates multi-valued attributes [8].

Definition 1. *Access structure [8]:*
We let $\mathcal{U} = \{a_1, a_2, ...a_n\}$ *be a set of attributes. For* $a_i \in \mathcal{U}$ *the set of possible values is denoted as* $\mathbb{A}_i = \{q_{i,1}, q_{i,2}, ...q_{i,n_i}\}$, *where* n_i *is the size of possible values for* a_i. *We let* $F = F_1, F_2, .., F_n$, *where* $F_i \in \mathbb{A}_i$ *be an attribute list for a user, and* $\mathbb{P} = \mathbb{P}_1, \mathbb{P}_2, ..., \mathbb{P}_n$, $\mathbb{P}_i \in \mathbb{A}_i$ *be an access structure. From the notation* $F \vDash \mathbb{P}$, *an attribute list* F *is satisfying an access structure* $\mathbb{P}$ *where* $F_i = \mathbb{P}_i (i = 1, 2, .., n)$.

The size of the access structures corresponds to $\prod_{i=1}^{n} n_i$. *In every* a_i, *the hospital has to demonstrate a status* $q_{i,*}$ *from* $\mathbb{A}_i = \{q_{i,1}, q_{i,2}, ...q_{i,n_i}\}$.

2.3 Taget-collision Resistant Hashing [7]

A function H: X→ Y is a target-collision resistant(TCR) hash function if given a random preimage x∈ X it is hard to realize x′≠x with H(x′)=H(x).

2.4 The Decisional Bilinear Diffie-Hellman (DBDH) Assumption [28]

Let $x,y,z,c \in \mathbb{Z}_p^*$ be randomly chosen and g be a generator of $\mathbb{G}$. The decisional BDH assumption [28] is that no probabilistic polynomial-time $\mathbb{AD}$ can distinguish the tuple$(g, X = g^x, Y = g^y, Z = g^z, e(g,g)^{xyz})$ from the tuple $(g, X = g^x, Y = g^y, Z = g^z, e(g,g)^c)$ with more than a negligible advantage. An algorithm $\mathbb{AD}$ has advantage of ϵ in solving DBDH problem in $\mathbb{G}$ if $Adv_{DBDH}(\mathbb{AD}) := |Pr[\mathbb{AD}(g, g^x, g^y, g^z, e(g,g)^{xyz}) = 0] - Pr[\mathbb{AD}(g, g^x, g^y, g^z, e(g,g)^c) = 0]| > \epsilon$.

We say that the DBDH assumption holds in $\mathbb{G}$ and $\mathbb{G}_\tau$ if ϵ is negligible.

2.5 The Modified Decisional Diffie-Hellman (MDDH) Assumption [34]

Let $x, y, z \in \mathbb{Z}_p^*$ be randomly chosen and g be a generator of $\mathbb{G}$. The Modified DDH assumption [34] is that no probabilistic polynomial-time adversary $\mathbb{AD}$ can distinguish the tuple$(g, X = g^x, B = e(g,g)^y, C = e(g,g)^{xy})$ from the tuple $(g, A = g^x, B = e(g,g)^y, C = e(g,g)^z)$ with more than a negligible advantage. An algorithm $\mathbb{AD}$ has advantage ϵ in solving MDDH problem in $\mathbb{G}$ if $Adv_{MDDH}(\mathbb{AD}) := |Pr[\mathbb{AD}(g, g^x, e(g,g)^y, e(g,g)^{xy}) = 0] - Pr[\mathbb{AD}(g, g^x, e(g,g)^y, e(g,g)^z) = 0]| > \epsilon$.

We say that the MDDH assumption holds in $\mathbb{G}$ and $\mathbb{G}_\tau$ if ϵ is negligible.

2.6 Notations

For convenience and better understanding the notations used in our protocol and their description is as shown in Table 1.

Table 1: Notations

ACRONYM	DESCRIPTION
$\mathcal{U}$	Attribute universe
$\mathbb{P}$	Access structure
$\mathbb{A}$	Set of attributes
AA	Attribute Authority
ISP	Internet Service Provider
M	Message
CT	CipherText
MSK	Master Secret Key
PK	Public Key
SK	Private Key
TK	Transformation Key
RK	Retrieval key
k	length of a string
$\oplus$	An exclusive-OR(XOR)
CSP	Cloud Service Provider
H_1, H_2, H_3	Three hash functions
F	Attribute list

2.7 Generic ABE Definition [28]

There are two variants of ABE schemes; key-policy ABE (KP-ABE)[13] and ciphertext policy ABE (CP-ABE) [2, 31]. Each ciphertext is associated with a set of attributes and each secret key is associated with an access policy for the case of KP-ABE while in CP-ABE, the attribute

sets are associated with secret keys and access policies are associated with ciphertext.

Definition 2. *Attribute-Based Encryption (ABE) [28, 21]: ABE has an attribute universe defined by $\mathcal{U}$ with access structure $\mathbb{P}$ and is described using the following polynomial-time algorithms:*

Setup(1^{β},$\mathcal{U}$) $\rightarrow$ (PK, MSK): *This algorithm acquires as an input a security parameter β and an attribute universe $\mathcal{U}$ and yields a public key PK and a master secret key MSK.*

KeyGen(PK,MSK,$L_{key}$$\rightarrow$SK: *The algorithm acquires as input public key PK, the master secret key MSK and list of descriptive attributes $L_{key} \in \mathbb{P}$ for key generation in the case of KP-ABE and $L_{key} \subseteq \mathcal{U}$ for the case of CP-ABE. It yields the secret key SK.*

Encrypt(PK,m,L_{enc})$\rightarrow$$CT$: *This algorithm acquires as input a public key parameter PK, a message m, and an attribute list $L_{enc} \subseteq \mathcal{U}$ for KP-ABE or $L_{enc} \subseteq \mathbb{P}$ for CP-ABE. It yields ciphertext CT.*

Decrypt$(SK, CT) \rightarrow m$**:** *This algorithm acquires as input secret key SK and a ciphertext CT. It yields message m if the function G(L_{key},L_{enc})=1 holds, otherwise yields $\perp$.*

Correctness.
$\forall$(PK,MSK)$\leftarrow$Setup(1^{β},$\mathcal{U}$),$SK \leftarrow$KeyGen(PK, MSK, L_{key}), $\forall m$ in message space, if G(L_{key},L_{enc})=1 holds, m=Decrypt(SK,Encrypt(PK,m,L_{enc}).

2.8 Attribute-Based with Outsourcing Decryption [11]

Has the following polynomial-time algorithms:

Setup(1^{β},$\mathcal{U}$): Takes as an input a security parameter β and a universe $\mathcal{U}$. It yields public parameters PK and master key MSK.

Encrypt(PK,m,(A,ρ)): Takes as an input public parameter PK and a message m to be encrypted. It also takes as input LSSS access structure(A,ρ). It yields ciphertext CT.

KeyGen$_{out}$(MSK,S): Takes as input master key MSK and set S. It yields transformation key TK and a secret key SK.

Transform$_{out}$(TK,CT): Takes as input transformation key TK and ciphertext CT. It yields CT$'$.

Decrypt$_{out}$(SK,CT$'$): Takes as input a secret key SK and transformed ciphertext CT$'$. yields message m.

3 Our Secured CP-ABE with Outsourced Decryption in Cloudlet Computing

3.1 Algorithm Definition

Our new scheme consist of seven algorithms as in [14]:

Setup(1^{β},$\mathcal{U}$): This algorithm takes as input security parameter β and universe $\mathcal{U}$. It returns as an output public parameter PK and master secret key MSK.

KeyGen(PK,MSK,F): This algorithm takes as input public parameter PK, master key MSK and a set F. Returns as an output private key SK_F.

Encrypt(PK,m,$\mathbb{P}$): This algorithm takes as input public parameters PK, message m and access structure $\mathbb{P}$. Returns as an output the ciphertext CT.

Decrypt(PK,SK$_F$,CT): Takes public parameter PK, private key SK_F for the list F. Returns as an output message m if SK_F associated with F satisfies $\mathbb{P}$

TKGen$_{out}$(PK,SK$_F$): This algorithm takes as input public parameter PK and private key SK$_F$ associated with list F. It returns as an output transformation key TK_F associated with F and its corresponding retrieving key RK_F.

PartialDecrypt$_{out}$(PK,TK$_F$,CT): This algorithm takes as an input public parameter PK, transformation key TK_F and ciphertext CT. Returns as an output partially decrypted ciphertext CT'.

Decrypt$_{out}$(PK,RK$_F$,CT,CT$'$): This algorithm takes as an input public parameter PK,retrieving key RK_F, ciphertext CT and transformed ciphertext CT'. Returns as an output message m.

Correctness:

1) $Decrypt(PK, SK_F, Encrypt(PK, m, \mathbb{P})) = m$

2) $Decrypt_{out}(PK, RK_F$,PartialDecrypt(Encrypt(PK, m, $\mathbb{P}$),PK, TK_F))=m

3.2 Definition of Security Model

We define security notions in this section. We propose **selective AND access structure and chosen ciphertext security**(IND-AND-sAS-CCA2) game in our CBeBDS-OD scheme. We have two kinds of chosen ciphertext variants similar to [20, 34]:

1) The challenge ciphertext is original.

2) The challenge ciphertext is server-aided.

3.2.1 The challenge ciphertext is original

Init: The adversary $\mathbb{AD}$ sends challenge access structure $\mathbb{P}^*$ to the challenger $\mathbb{BD}$.

Setup: Challenger $\mathbb{BD}$ invokes setup algorithm to obtain public parameter (PK) and master secret key(MSK). $\mathbb{BD}$ then sends PK to the adversary $\mathbb{AD}$ and keeps MSK secret.

Phase 1: In the find stage $\mathbb{AD}$ has access to the following oracles queries:

Private Key query oracle $\mathcal{OR}_{SK_F}$, for the attribute list $F \not\models \mathbb{P}^*$: $\mathbb{BD}$ runs $SK_F \leftarrow$ Keygen(PK, MSK, F). The challenger then sends the private key SK_F to $\mathbb{AD}$.

Partial decryption key query oracle $\mathcal{OR}_{TK}$, for the attribute list F: On input F and access structure $\mathbb{P}^*$, challenger $\mathbb{BD}$ returns equivalent transformation key TK_F.

Retrieving key query oracle $\mathcal{OR}_{RK}$, for attribute list F: Challenger $\mathbb{BD}$ returns equivalent retrieving key RK_F on input of F.

Partial decryption query oracle $\mathcal{OR}_{PDec}$, for attribute list F and ciphertext $\boldsymbol{CT}$: On input of ($\boldsymbol{CT}$, F), $\mathbb{BD}$ outputs PartialDecrypt$_{out}$($\boldsymbol{PK}$,TK_F,$\boldsymbol{CT}$).

Decryption query oracle $\mathcal{OR}_{Dec}$, for attribute list F and ciphertext $\boldsymbol{CT}$. $\mathbb{BD}$ invokes $\boldsymbol{m}\leftarrow$Decrypt(PK, SK_F, CT), where $SK_F \leftarrow$Keygen(PK, MSK, F) and $F \models \mathbb{P}^*$ if the input ciphertext is original. Otherwise $\boldsymbol{m}\leftarrow$ Decrypt(PK, RK_F, CT) where $RK_F \leftarrow (SK_F, F)$ if ciphertext is server-aided. $\boldsymbol{m}$ is send to $\mathbb{AD}$.

Challenge: $\mathbb{AD}$ submits two plaintext messages m_0^* and m_1^* of equal length from the message space M and the access policy $\mathbb{P}^*$ on which it wishes to challenge with the constraint that F cannot satisfy $\mathbb{P}^*$. $\mathbb{BD}$ flips a random coin γ to choose $\gamma\in\{0,1\}$ and encrypts m_γ^* under the access structure $\mathbb{P}^*$ *i.e.* CT^*=*Encrypt(PK,*m_γ^**,*$\mathbb{P}^*$*)*. Then $\mathbb{BD}$ sends the challenge ciphertext CT^* to $\mathbb{AD}$.

Phase 2: $\mathbb{AD}$ maintains query as in *phase 1* adaptively for private key, transformation key, retrieving key, decryption and decryption$_{out}$ with the following constraints:

1) $\mathbb{AD}$ should not make private key query that results in attribute list F that will satisfy access policy $\mathbb{P}^*$.
2) $\mathbb{AD}$ should not trivially issue decryption queries.

Guess: The adversary $\mathbb{AD}$ gives $\gamma'\in\{0,1\}$ for γ and wins the game if $\gamma'=\gamma$. $|\Pr(\gamma'=\gamma)\text{-}\frac{1}{2}|$ is defined as the advantage for adversary $\mathbb{AD}$ in the game.

Definition 3. *Cloudlet-Based eHealth Big Data System with Outsourced Decryption (CBe-BDS-OD) is said to be IND-AND-sAS-CCA2-or secure*[1] *if for all polynomial adversaries the advantage* $\boldsymbol{Adv}_{CBe-BDS-OD}^{IND-AND-sAS-CCA-or}(\beta)$ *is negligible.*

The challenge ciphertext is server-aided:

Phase 1: Similar to the one of original challenge ciphertext.

Challenge: $\mathbb{AD}$ submits two plaintext messages m_0^* and m_1^* of equal length derived from the message space M and the access policy $\mathbb{P}^*$ which it wishes to challenge with the constraint that F has not been used to query $\mathcal{OR}_{RK}$. $\mathbb{BD}$ flips a random coin γ to choose $\gamma\in\{0,1\}$ and sets CT^*=(TK^*,Encrypt(PK, m_γ^*, F^*)). Then $\mathbb{BD}$ sends the challenge ciphertext CT^* to $\mathbb{AD}$.

Phase 2: Nearly similar to *phase 1* with the following restrictions:

$$\mathcal{OR}_{TK} : F = F^*$$
$$\mathcal{OR}_{Dec} : CT = CT^* \quad \text{also} \quad F = F^*.$$

Guess: Similar to that in original challenge ciphertext scenario. $|\Pr(\gamma'=\gamma)\text{-}\frac{1}{2}|$ is defined as the advantage for adversary $\mathbb{AD}$ in the game.

Definition 4. *Cloudlet-Based eHealth Big Data System with Outsourced Decryption (CBe-BDS-OD) is said to be IND-AND-sAS-CCA2-sa secure*[2] *if for all polynomial adversaries the advantage* $\boldsymbol{Adv}_{CBe-BDS-OD}^{IND-AND-sAS-CCA2-sa}(\beta)$ *is negligible.*

3.3 System Model

In this Section we give some intuition for the idea behind our scheme. Our scheme is partly based on scheme [8]. To realize outsourced decryption, our scheme partially follows [11, 34]. The hospital encrypts the data with public key PK_i. After which it outsources the encrypted data to the cloud for storage. Mobile device does not need to communicate with the Cloud directly instead it communicates with the closest cloudlet [29]. The mobile user offloads some intensive tasks to the cloudlet by letting it do partial decryption using token key (TK). Final decryption is carried out by the user using retrieving key RK. To achieve CCA2 security in our protocol we follow [4, 9] where decryptor is allowed to check whether the ciphertext is valid;

1) Before proxy transforms the original ciphertext by employing [4].
2) Before end user does final decryption utilizing [9].

[1] or stands for original
[2] sa stands for server-aided

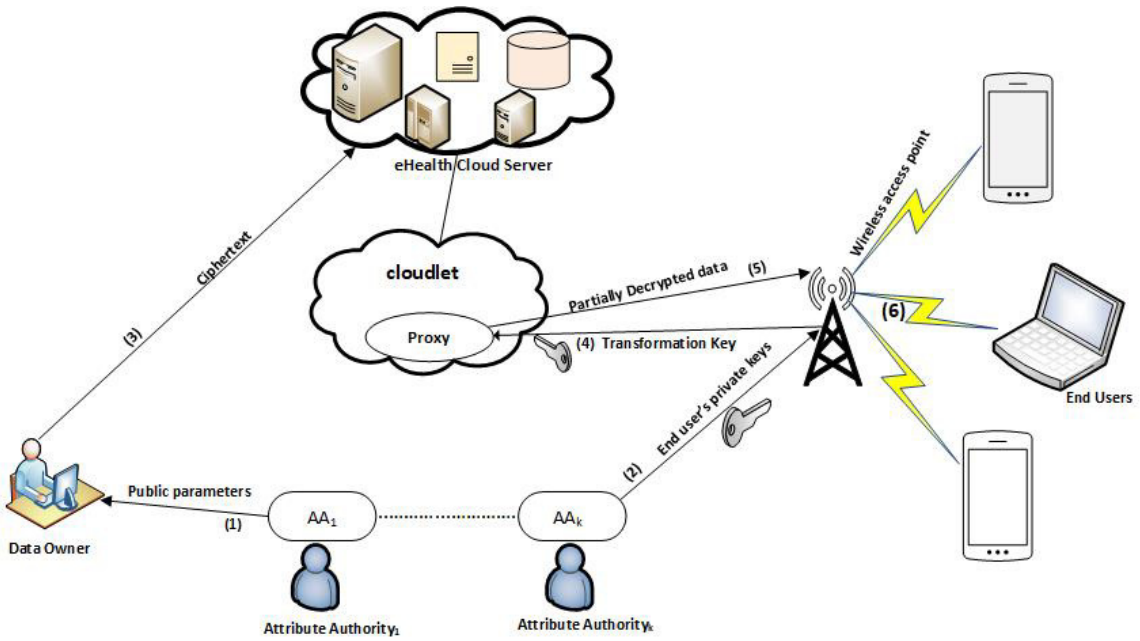

Figure 2: System model

We employ a cloudlet storage system that has multiple authorities as shown in Figure 2. The model has the following entities: cloud server, cloudlet, proxy, Trusted Authority/ authorities (AA), hospital, users.

1) *Trusted Authority (TA)*: Also referred to as an Attribute Authority (AA). It generates system public and secret keys. It is the only entity fully trusted by all other entities participating in the system. TA is used interchangeably with AA.

2) *Cloud server:* Cloud server stores eHealth big data for the hospital and also offers access service of data to the users.

3) *Cloudlet:* Cloudlet is located closer to mobile users.It hosts the offloaded tasks.

4) *Proxy:* Performs outsourced decryption for eHealth data user. It reduces the decryption load on users with resources-constrained devices.

5) *Hospital:* Describes the access policies and encrypts its data under those policies before sending them to the cloud.

6) *Users:* The user that uses resource-constrained device is assigned with the key and can access the ciphertext freely from the cloudlet.They can recover the plaintext only when its attributes satisfy the access policy stated in the ciphertext.

3.4 Threat Model

Users need solid assurance of the existence of adequate security and privacy aspects in cloud before trusting their data to it. In this subsection, we put into account the possible attackers and their corresponding attacks to our proposed protocol. Since cloudlet is closer to the user, fierce attacking can be realized than ever (active attacks). In addition, cloudlet proxy is assumed to be honest but curious (passive). It may deviate from the scheme specifications norms and may try to acquire as much private information as possible. To obtain the key to access data which individually could not access, authorized users can collude by combining their attributes (inner threats). Furthermore, the proxy can collude with unauthorized users to obtain some data. Integrity is another major issue as data may be mutated or get corrupted. The user must verify the integrity of outsourced data before decrypting.

3.5 Design Goals

Based on the possible attacks discussed in the preceding subsection our system achieves the following design security goals:

1) Confidentiality: The cloudlet service provider and malicious users cannot recover encrypted data without the owners consent. As shown by our system in Figure 2, only intended users are able to decrypt their messages. This ensures the privacy of the owner's data in storage, during partial decryption and final decryption.

2) Integrity: Our system verifies the correctness of original and transformed ciphertext.

3) Minimal overhead: Due to the resource-constraints in mobile devices, minimal overhead costs must be

provided during the time both of computation and communication.

4 Main Construction

As mentioned in Subsection 3.3 above, our CBe-BDS-OD is based on [8]. To realize outsourced decryption it follows partly [11, 34].

Our new scheme is composed of seven algorithms as in [14]:

Setup(1^β,$\mathcal{U}$): This algorithm takes as input security parameter β and universe $\mathcal{U}$. A trusted authority(TA) chooses a pairing group $\mathbb{BG}$=(p,e,$\mathbb{G}$,$\mathbb{G}_\tau$), two generators $g, h \in \mathbb{G}$, then selects two elements $b \in_R \mathbb{Z}_p$ and $\upsilon_{i,j} \in_R \mathbb{Z}_p (i \in [1,n], j \in [1,n_i])$. Trusted Authority furthermore computes $\mathbb{Y} = e(g,h)^b$ and $V_{i,j} = g^{\upsilon_{i,j}} (i \in [1,n], j \in [1,n_i])$ and selects the following TCR hash functions:

$$H_1 : \{0,1\}^* \to \mathbb{Z}_p^*,$$
$$H_2 : \mathbb{G}_\tau \to \{0,1\}^{2k},$$
$$H_3 : \{0,1\}^* \to \mathbb{G}.$$

It publishes public parameters $PK = (g,\ h,\ e,\ \mathbb{Y},\ V_{i,j},\ H_1,\ H_2,\ H_3)$ and master secret key $MSK = (b, \{\upsilon_{i,j}\}_{i\in[1,n],j\in[1,n_i]})$. Note that $\forall F, F' (F \neq F'), \sum_{q_{i,j}\in F} \upsilon_{i,j} \neq \sum_{q_{i,j}\in F'} \upsilon_{i,j}$ is assumed.

KeyGen(PK, MSK, F): This algorithm takes as input public parameter PK, master key MSK and a set F. TA then choses $\alpha \in \mathbb{Z}_p^*$. Next he sets

$$\begin{aligned} K_1 &= h^b (g^{\sum_{q_{i,j}\in F} \upsilon_{i,j}})^\alpha. \\ K_2 &= g^\alpha \\ SK_F &= (K_1, K_2). \end{aligned}$$

Encrypt($PK, m, \mathbb{P}$): This algorithm takes public parameters PK, message $m \in \{0,1\}^k$ and access structure $\mathbb{P}$ as input. The encryption algorithm works as follows.

1) Hospital chooses $\lambda \in \{0,1\}^k$ and sets $s = H_1(\mathbb{P}, m, \lambda)$, then it computes,

$$\begin{aligned} B_1 &= H_2(e(g,h)^{b.s}) \oplus (m \| \lambda) \\ B_2 &= g^s \\ B_3 &= (\prod_{q_{i,j}\in\mathbb{P}} V_{i,j})^s \\ B_4 &= h^s. \end{aligned}$$

Finally hospital computes $E = H_3(\mathbb{P} \| B_1 \| B_2 \| B_3 \| B_4)^s$.

2) The hospital then generates $\boldsymbol{CT}$=($\mathbb{P}$, B_1, B_2, B_3, B_4, E) then it sends to the cloud.

TKGen$_{out}$(PK, SK_F): This algorithm takes as input public parameter PK and private key SK_F associated with set F. Token key generation works as follows:
The user chooses $\sigma \in \mathbb{Z}_p^*$ then generates a partial decryption key pair as: $TK_F = (K'_{T1}, K'_{T2})$ where

$$\begin{aligned} K'_{T1} &= K_1^{1/\sigma} \\ K'_{T2} &= K_2^{1/\sigma}. \end{aligned}$$

Retrieving key is $RK_F = \sigma$.

It returns as an output transformation key TK_F associated with F and corresponding retrieving key RK_F.

PartialDecrypt$_{out}$(PK, TK_F, CT): This algorithm is executed by CSP. It takes as input public parameter PK, transformation key TK_F and ciphertext CT the proxy confirms first whether

$$\begin{aligned} e(B_2, h) &\stackrel{?}{=} e(g, B_4), \\ e(B_2, H_3(\mathbb{P}\|B_1\|B_2\|B_3\|B_4)) &\stackrel{?}{=} e(g, E), \\ e(\prod_{q_{i,j}\in\mathbb{P}} V_{i,j}, B_2) &\stackrel{?}{=} e(B_3, g), \\ F &\models \mathbb{P}. \end{aligned} \quad (1)$$

If does not hold it aborts with $\perp$, otherwise calculates the following:

$$\begin{aligned} T' &= \frac{e(B_2, K'_{T1})}{e(B_3, K'_{T2})} \\ &= \frac{e(g^s, h^{b/\sigma}(g^{\sum_{q_{i,j}\in F} \upsilon_{i,j}})^{\alpha/\sigma})}{e((\prod_{q_{i,j}\in\mathbb{P}} V_{i,j})^s, g^{\alpha/\sigma})} \\ &= \frac{e(g^s, h^{b/\sigma}(g^{\sum_{q_{i,j}\in F} \upsilon_{i,j}})^{\alpha/\sigma})}{e((\prod_{q_{i,j}\in\mathbb{P}} g^{\upsilon_{i,j}})^s, g^{\alpha/\sigma})} \\ &= \frac{e(g^s, h^{b/\sigma}(g^{\sum_{q_{i,j}\in F} \upsilon_{i,j}})^{\alpha/\sigma})}{e((g^{\sum_{q_{i,j}\in\mathbb{P}} \upsilon_{i,j}})^s, g^{\alpha/\sigma})} \\ &= \frac{e(g,h)^{s.b/\sigma}.e(g,g)^{\alpha.s/\sigma \sum_{q_{i,j}\in F} \upsilon_{i,j}}}{e(g,g)^{\alpha.s/\sigma \sum_{q_{i,j}\in\mathbb{P}} \upsilon_{i,j}}} \\ T' &= e(g,h)^{s.b/\sigma}. \end{aligned}$$

Returns as an output partially decrypted ciphertext $CT' = (B_1, T')$.

Decrypt(PK, SK_F, CT): Since there are two types of ciphertexts likewise two computations are to be carried out.

1) Parsing Original ciphertext as an input.
2) Parsing Server-aided ciphertext *i.e.* transformed data as an input.

Parsing Original ciphertext as an input: The decryption key employed is the private key $SK_F = (K_1, K_2)$ that corresponds to set F. The user confirms

Table 2: Performance evaluation benchmark

Operation	**Notation**	**Time computation in μs**	
		Desktop PC setting	**Smartphone setting**
Exponentiation in $\mathbb{G}_\tau$	T_e	0.067	42
Bilinear Pairing	T_p	16.06	63

the validity of the ciphertext as in PartialDecrypt$_{out}$ above. If it does not hold, he outputs $\perp$, otherwise he proceeds by taking public parameter PK, private key SK_F for the set F and original ciphertext CT then calculates the following.

$$\begin{aligned} T'' &= \frac{e(B_2, K_1)}{e(B_3, K_2)} \\ &= \frac{e(g^s, h^b(g^{\sum_{q_{i,j}\in F} v_{i,j}})^\alpha)}{e((\prod_{q_{i,j}\in \mathbb{P}} V_{i,j})^s, g^\alpha)} \\ &= \frac{e(g^s, h^b(g^{\sum_{q_{i,j}\in F} v_{i,j}})^\alpha)}{e((\prod_{q_{i,j}\in \mathbb{P}} g^{v_{i,j}})^s, g^\alpha)} \\ &= \frac{e(g^s, h^b(g^{\sum_{q_{i,j}\in F} v_{i,j}})^\alpha)}{e((g^{\sum_{q_{i,j}\in \mathbb{P}} v_{i,j}})^s, g^\alpha)} \\ &= \frac{e(g,h)^{sb}.e(g,g)^{\alpha.s\sum_{q_{i,j}\in F} v_{i,j}}}{e(g,g)^{\alpha.s\sum_{q_{i,j}\in \mathbb{P}} v_{i,j}}} \\ T'' &= e(g,h)^{sb}. \end{aligned}$$

Computes $m\|\lambda = H_2(T'') \oplus B_1$. Finally it outputs m if $B_1 = H_2(\mathbb{Y}^{H_1(\mathbb{P},m,\lambda)}) \oplus (m\|\lambda)$ holds, otherwise outputs $\perp$.

Parsing Server-aided ciphertext *i.e.* transformed data as an input: The input ciphertext is the partially decrypted one (CT') and decryption key employed here is the retrieving key $RK_F{=}\sigma$ that corresponds to attribute set F. If F satisfies $\mathbb{P}$ then the user computes:

$$m\|\lambda = H_2(T'^\sigma) \oplus B_1.$$

Finally it outputs m if $B_1 = H_2(\mathbb{Y}^{H_1(\mathbb{P},m,\lambda)}) \oplus (m\|\lambda)$ and $\mathbb{Y}^{H_1(\mathbb{P},m,\lambda)} = T'^\sigma$ holds, otherwise outputs $\perp$.

Correctness Analysis
Two faces:

1) Correctness for Original ciphertext.

2) Correctness for Server-aided ciphertext.

Correctness for Original ciphertext:

$$\begin{aligned} T'' &= \frac{e(B_2, K_1)}{e(B_3, K_2)} \\ &= \frac{e(g^s, h^b(g^{\sum_{q_{i,j}\in F} v_{i,j}})^\alpha)}{e((\prod_{q_{i,j}\in \mathbb{P}} V_{i,j})^s, g^\alpha)} \\ &= \frac{e(g^s, h^b(g^{\sum_{q_{i,j}\in F} v_{i,j}})^\alpha)}{e((\prod_{q_{i,j}\in \mathbb{P}} g^{v_{i,j}})^s, g^\alpha)} \\ &= \frac{e(g^s, h^b(g^{\sum_{q_{i,j}\in F} v_{i,j}})^\alpha)}{e((g^{\sum_{q_{i,j}\in \mathbb{P}} v_{i,j}})^s, g^\alpha)} \\ &= \frac{e(g,h)^{sb}.e(g,g)^{\alpha.s\sum_{q_{i,j}\in F} v_{i,j}}}{e(g,g)^{\alpha.s\sum_{q_{i,j}\in \mathbb{P}} v_{i,j}}} \\ T'' &= e(g,h)^{sb}. \end{aligned}$$

Therefore we have, $H_2(T'') \oplus B_1 = H_2(e(g,h)^{sb}) \oplus (m\|\lambda) \oplus H_2(e(g,h)^{sb}) = m\|\lambda$.

Correctness for Server-aided ciphertext:

$$\begin{aligned} T' &= \frac{e(B_2, K'_{T1})}{e(B_3, K'_{T2})} \\ &= \frac{e(g^s, h^{b/\sigma}(g^{\sum_{q_{i,j}\in F} v_{i,j}})^{\alpha/\sigma})}{e((\prod_{q_{i,j}\in \mathbb{P}} V_{i,j})^s, g^{\alpha/\sigma})} \\ &= \frac{e(g^s, h^{b/\sigma}(g^{\sum_{q_{i,j}\in F} v_{i,j}})^{\alpha/\sigma})}{e((\prod_{q_{i,j}\in \mathbb{P}} g^{v_{i,j}})^s, g^{\alpha/\sigma})} \\ &= \frac{e(g^s, h^{b/\sigma}(g^{\sum_{q_{i,j}\in F} v_{i,j}})^{\alpha/\sigma})}{e((g^{\sum_{q_{i,j}\in \mathbb{P}} v_{i,j}})^s, g^{\alpha/\sigma})} \\ &= \frac{e(g,h)^{s.b/\sigma}.e(g,g)^{\alpha.s/\sigma\sum_{q_{i,j}\in F} v_{i,j}}}{e(g,g)^{\alpha.s/\sigma\sum_{q_{i,j}\in \mathbb{P}} v_{i,j}}} \\ T' &= e(g,h)^{s.b/\sigma}. \end{aligned}$$

Therefore, $H_2(T'^\sigma) \oplus B_1 = H_2(e(g,h)^{s.b/\sigma)^\sigma} \oplus (m\|\lambda) \oplus$ and $H_2(e(g,h)^{s.b}) = m\|\lambda$.

5 Efficiency Evaluation

In this Section we present the performance evaluation for both original and server-aided proposed schemes in Personal Computer (PC) and Mobile setting environments. For better understanding we defined the following notations: T_e and T_p for pairing exponentiation and bilinear pairing respectively as shown in Table 2. We omitted

scalar multiplication intentionally as its running time is minimal. Our performance evaluation is based on experimental results we carried out in our PC settings with Core i-5 2.5GHz platform processor with memory 4 GB and the Windows XP operating system and for Mobile settings we adopted the experiment results due to [3], where the hardware platform is Samsung Galaxy S2 smartphone with a Dual-core Exynos 4210 1.2GHz processor ARM Cortex-A9 with the Android OS, V2.3(Gingerbread). In PC setting as indicated in Table 3 (PC setting), the total running time of original proposed scheme is 32.12 m/s while in mobile setting Table 4 it is 126 m/s. In Server-aided scheme(PC setting) the total running time is 16.127 m/s while for its counterpart in mobile setting it is 105 m/s. Figure 3 and Figure 4 shows the relative running times of the considered operations for both original and server-aided schemes in PC setting and mobile setting respectively. Based on the above results it is evident that the server-aided scheme is more practical to devices which are resource-constrained in nature.

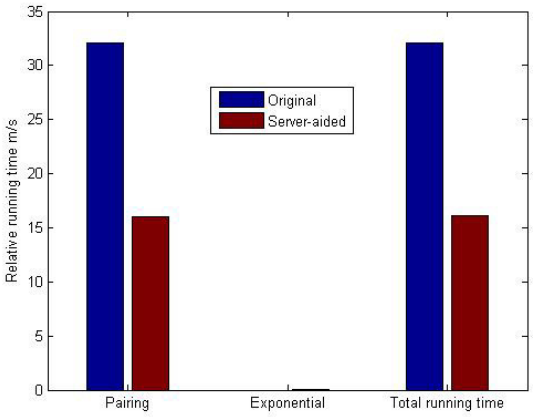

Figure 3: PC setting

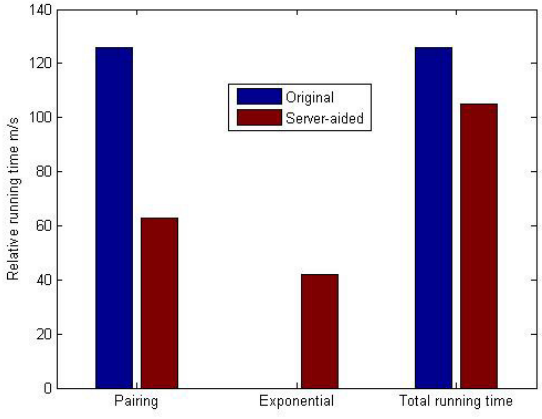

Figure 4: Mobile setting

6 Security Analysis

In our security analysis we show that our CBe-BDS-OD protocol is IND-AND-sAS-CCA2-secure in the definition of our proposed IND-AND-sAS-CCA2 security. The fundamental idea behind our security proof is similar to [34].

Theorem 1. *Suppose the DBDH assumption[28] holds in* $(\mathbb{G}, \mathbb{G}_\tau)$ *and* H_1, H_2, H_3 *are the TCR hash functions, then our CBe-BDS-OD is IND-AND-sAS-CCA2-or secure in the random oracle model.*

Proof. We show that if there exist an adversary $\mathbb{AD}$ that can break the IND-AND-sAS-CCA2-or security of the proposed CBe-BDS-OD scheme then we construct an algorithm $\mathbb{BD}$ using $\mathbb{AD}$ as a subroutine to solve DBDH problem. First DBDH challenger flips a fair coin δ and if δ=0, it sets $(g, X, Y, Z, C) := (g, g^x, g^y, g^z, e(g,g)^{xyz})$; otherwise it sets $(g, X, Y, Z, C) := (g, g^x, g^y, g^z, e(g,g)^c)$ where $x, y, z, c \in \mathbb{Z}_p$ are randomly chosen. The challenger then gives $\mathbb{BD}$ $(g, X, Y, Z, C) := (g, g^x, g^y, g^z, C)$. To be specific $\mathbb{BD}$ will act as a challenger with $\mathbb{AD}$ to play the following IND-AND-sAS-CCA2-or game.

Init: $\mathbb{AD}$ sends the challenge structure $\mathbb{P}^*$ to $\mathbb{BD}$. Let $\mathbb{P}^* = [\mathbb{P}_1^*, ..., \mathbb{P}_n^*]$.

Setup: $\mathbb{BD}$ sets $h = g^\omega$ and $\mathbb{Y} = e(g^x, (g^y)^\omega) = e(g,h)^{xy}$ where $\omega \in \mathbb{Z}_p^*$ is chosen randomly. Furthermore, $\mathbb{BD}$ chooses $\upsilon'_{i,j} \in \mathbb{Z}_p$(where i$\in$ [1,n], j$\in$ [1,n$_i$]). There are two cases: 1) where $q_{i,j} = \mathbb{P}_i^*$ he sets $\upsilon_{i,j} = \upsilon'_{i,j}$. 2) where $q_{i,j} \neq \mathbb{P}_i^*$ sets $\upsilon_{i,j} = y\upsilon'_{i,j}$. Then computes public keys $V_{i,j}$ (where $i \in n[1,n], j \in n_i[1,n_i]$) in the following manner:

$$V_{i,j} = g^{\upsilon_{i,j}} = \begin{cases} g^{\upsilon'_{i,j}}, & \text{where } q_{i,j} = \mathbb{P}_i^* \\ g^{\upsilon_{i,j}}, & \text{where } q_{i,j} \neq \mathbb{P}_i^* \end{cases} \quad (2)$$

$\mathbb{BD}$ then chooses the TCR hash functions as in the actual scheme and sends parameters=(p,$g, h, \mathbb{G}, \mathbb{G}_\tau, e, \mathbb{Y}, V_{i,j}$) to $\mathbb{AD}$. $\mathbb{AD}$ can adaptively query random oracles $H_i (i \in \{1, ., 3\})$ that are controlled by $\mathbb{BD}$. $\mathbb{BD}$ maintains the list $H_i^{List} (i \in \{1,.,3\})$ which originally were empty. If the query has been responded and recorded previously in the list, $\mathbb{BD}$ responds with the same result. $\mathbb{BD}$ response to random oracle queries is as follows:

1) H_1: Upon receiving H_1 query on ($\mathbb{P}^*$,m,λ), $\mathbb{BD}$ first confirms whether there's a tuple $(\mathbb{P}^*, m, \lambda, s)$ in the list H_1^{List}. If it exists it returns the predefined s to $\mathbb{AD}$, where $s \in \mathbb{Z}_p^*$. Otherwise, $\mathbb{BD}$ computes H_1($\mathbb{P}^*$,$m, \lambda) = s$, answers $\mathbb{AD}$ with s, then the tuple $(\mathbb{P}^*, m, \lambda, s)$ is added to the list H_1^{List}.
2) H_2: Upon receiving H_2 query on $RC \in \mathbb{G}_\tau$, $\mathbb{BD}$ first confirms whether there's a tuple(RC, κ_1) in the list H_2^{List}. If it exists it returns κ_1 to $\mathbb{AD}$. Otherwise $\mathbb{BD}$ computes $H_2(RC) = \kappa_1$, answers $\mathbb{AD}$ with κ_1 then the tuple(RC, κ_1) is added to the $H_2^{List}, \kappa_1 \in \{0,1\}^{2k}$.

Table 3: PC setting

	Number of operations and running time (m/s)				
	Exponential		Pairing		
Our scheme	**No.**	**Time(m/s)**	**No.**	**Time(m/s)**	**Total time (m/s)**
Original	0	0	2	32.12	32.12
Server-aided	1	0.067	1	16.06	16.127

Table 4: Mobile setting

	Number of operations and running time (m/s)				
	Exponential		Pairing		
Our scheme	**No.**	**Time(m/s)**	**No.**	**Time(m/s)**	**Total time (m/s)**
Original	0	0	2	126	126
Server-aided	1	42	1	63	105

3) H_3: Upon receiving H_3 query on tuple $(\mathbb{P}^*\|\mathrm{B}_1\|\mathrm{B}_2\|\mathrm{B}_3\|\mathrm{B}_4)$, $\mathbb{BD}$ first confirms whether there exists such tuple $(\mathbb{P}^*\|\mathrm{B}_1\|\mathrm{B}_2\|\mathrm{B}_3\|\mathrm{B}_4, \kappa_2, \eta)$ in the list H_3^{List}, $\mathbb{BD}$ returns the predefined value κ_2 to $\mathbb{AD}$ where $\kappa_2 \in \mathbb{G}, \eta \in \mathbb{Z}_p$. Otherwise $\mathbb{BD}$ computes $\kappa_2 = g^{\eta}$, returns κ_2 to $\mathbb{AD}$ and the tuple $(\mathbb{P}^*\|\mathrm{B}_1\|\mathrm{B}_2\|\mathrm{B}_3\|\mathrm{B}_4, \kappa_2, \eta)$ is added to the list H_3^{List}, $\eta \in \mathbb{Z}_p$.

Furthermore $\mathbb{BD}$ maintains the following lists which originally are empty:

- $SK^{List_{sk}}$ that stores (F, SK_F) tuple which are responds from $\mathcal{OR}_{SK_F}(F)$ queries.
- $TK^{List_{tk}}$ that stores (F, SK_F, TK_F, RK_F) tuple which are responds from $\mathcal{OR}_{TK_F}(F, \mathbb{P}^*)$ queries
- $RK^{List_{rk}}$ that stores (F, SK_F, TK_F, RK_F) tuple which are responds from $\mathcal{OR}_{RK_F}(F, \mathbb{P}^*)$ queries.

Phase 1: $\mathbb{AD}$ issues sequence of queries to which $\mathbb{BD}$ answers as follows:

Private key extraction oracle $\mathcal{OR}_{SK_F}$(F): $\mathbb{BD}$ computes the private key for the attribute list F in the following manner; If $F \vDash \mathbb{P}^*$ $\mathbb{BD}$ aborts with $\perp$, otherwise if $F \nvDash \mathbb{P}^*$ as in [8] there exist $q_{i,l}$ such that $q_{i,l} = F_i \wedge q_{i,l}$. Thus $\sum_{q_{i,j} \in F} \upsilon_{i,j} = V_1 + yV_2$ such that $V_1, V_2 \in \mathbb{Z}_p$. Also V_1, V_2 are represented by the sum of $\upsilon'^{i,j}$. Therefore $\mathbb{BD}$ can compute V_1 and V_2. $\mathbb{BD}$ selects $\phi \in \mathbb{Z}_p$, sets $\alpha = \frac{\phi - \omega x}{V_2}$ then computes the following:

$$\begin{aligned} SK_F &= (K_1, K_2) \\ K_1 &= (g^y)^{\phi} g^{\frac{V_1}{V_2}\phi} (g^x)^{-\frac{V_1 \omega}{V_2}} \\ K_2 &= g^{\frac{\phi}{V_2}} (g^x)^{-\frac{\omega}{V_2}} \end{aligned}$$

As in [8], we establish that SK_F is valid private key in the following manner:

$$\begin{aligned} K_1 &= (g^y)^{\phi} g^{\frac{V_1}{V_2}\phi} (g^x)^{-\frac{V_1 \omega}{V_2}} \\ &= g^{\omega xy}.g^{-\omega xy} (g^y)^{\phi} g^{\frac{V_1}{V_2}\phi} (g^x)^{-\frac{V_1 \omega}{V_2}} \\ &= g^{\omega xy}.g^{\frac{V_1}{V_2}(\phi - \omega x)} g^{y(\phi - \omega x)} \\ &= g^{\omega xy} (g^{V_1}.g^{yV_2})^{\frac{\phi - \omega x}{V_2}} \\ &= g^{\omega xy} (g^{V_1 + yV_2})^{\frac{\phi - \omega x}{V_2}} \\ &= h^b (g^{\sum_{q_{i,j} \in F} \upsilon_{i,j}})^{\alpha}. \end{aligned} \quad (3)$$

Note that in the above Equation 3, expression $g^{\omega xy}.g^{-\omega xy}$ is introduced as it is =1, since subtracting exponents $\omega xy - \omega xy$=0.

$K_2 = g^{\frac{\phi}{V_2}} (g^x)^{-\frac{\omega}{V_2}} = g^{\frac{\phi - \omega x}{V_2}} = g^{\alpha}$.
Therefore if V_2=0 mod p then $\mathbb{BD}$ outputs $\perp$. In the case V_2=0 mod p holds then it means there exists list F such that $\sum_{q_{i,j} \in F} \upsilon_{i,j} = \sum_{q_{i,j} \in \mathbb{P}^*} \upsilon_{i,j}$ holds. For more information we refer the reader to [8]. Finally $\mathbb{BD}$ stores the tuple(F, SK_F) to the $SK^{List_{sk}}$ and returns SK_F to $\mathbb{AD}$.

Transformation key extraction oracle $\mathcal{OR}_{TK_F}(F)$: Upon receiving attribute list F and access structure $\mathbb{P}^*$, $\mathbb{BD}$ confirms if there exists tuple (F, SK_F, TK_F, RK_F) in the list $TK^{List_{tk}}$. If it exists, $\mathbb{BD}$ returns the equivalent TK_F. Otherwise, if $F \nvDash \mathbb{P}^*$, $\mathbb{BD}$ responds with an equivalent private key SK_F by invoking $\mathcal{OR}_{SK_F}$ and its equivalent transformation key TK_F as in the actual scheme.

$$\begin{aligned} SK_F &= (TK_F, RK_F) \\ TK_F &= (K_1^{1/\sigma}, K_2^{1/\sigma}) \\ RK_F &= \sigma. \end{aligned}$$

If $F \vDash P^*$, then $\mathbb{BD}$ selects $\sigma \in \mathbb{Z}_p, (S_1, S_2) \in \mathbb{G}$ then computes $SK_F = (TK_F, RK_F) = ((S_1^{1/\sigma}, S_2^{1/\sigma}), \sigma)$. Finally $\mathbb{BD}$ adds the tuple(F, SK_F, TK_F, RK_F) to the list $TK^{List_{tk}}$ and responds $\mathbb{AD}$ with TK_F.

Retrieving Key extraction oracle $\mathcal{OR}_{RK_F}$: Almost similar to $\mathcal{OR}_{TK_F}$. It responds with retrieving key RK_F.

Partial decryption query oracle $\mathcal{OR}_{PDec}$: Upon receiving the tuple (CT, F), $\mathbb{BD}$ verifies whether $F \vDash \mathbb{P}$ embedded in CT. If it does not satisfies, it aborts with $\perp$. Otherwise it returns $PartialDecrypt_{out}(TK_F, CT)$. TK_F is the transformation key generated from $\mathcal{OR}_{TK_F}$.

Decryption query oracle $\mathcal{OR}_{Dec}$: Upon receiving the tuple (CT, F), $\mathbb{BD}$ verifies whether $F \vDash \mathbb{P}$ lodged in CT. If it does not satisfies, it aborts with $\perp$. Otherwise it carries out the following actions:

- If the ciphertext is original, first $\mathbb{BD}$ checks whether Equation 1 above is satisfied. If not $\mathbb{BD}$ outputs $\perp$. Else if $(F, SK_F) \in SK^{List_{sk}}$, $\mathbb{BD}$ recovers the message m using SK_F as in the actual scheme. Otherwise $\mathbb{BD}$ checks whether $(\mathbb{P}, m, \lambda, s) \in H_1^{List}$ and also $(RC, \kappa_1) \in H_2^{List}$ in such way that $\mathrm{B}_1{=}\kappa_1 \oplus (\mathrm{m}\|\lambda)$ while $RC = e(g,h)^{s.b}$. If such kind of tuple exists $\mathbb{BD}$ outputs m. Otherwise aborts with $\perp$.
- If the ciphertext is server-aided, first $\mathbb{BD}$ confirms if there exists tuple (F, SK_F, TK_F, RK_F) in the list $TK^{List_{tk}}$. If it does not exist, it aborts with $\perp$. Else $\mathbb{BD}$ checks whether $(\mathbb{P}, m, \lambda, s) \in H_1^{List}$ and also $(RC, \kappa_1) \in H_2^{List}$ in such way that $\mathrm{B}_1{=}\kappa_1 \oplus (\mathrm{m}\|\lambda)$ while $RC = e(g,h)^{sb}$. If such kind of tuple does not exists it outputs $\perp$. Otherwise $\mathbb{BD}$ confirms whether $T' = e(g,h)^{sb}/\varphi$ where $\varphi = H_1(\varepsilon)$ if its true it outputs m. Otherwise aborts with $\perp$.

Challenge: $\mathbb{AD}$ submits two plaintext messages m_0^* and m_1^* of equal length from the message space $\{0,1\}^k$. $\mathbb{BD}$ flips a random coin γ to choose $\gamma \in \{0,1\}$ and encrypts m_γ^* as follows:
$\mathbb{BD}$ selects $\lambda^* \in \{0,1\}^k, B_1^* \in \{0,1\}^{2k}$ then sets:

1) Compute $B_1^* = H_2(Z) \oplus (m_\gamma^* \| \lambda^*)$, $B_2^* = g^s$, $B_3^* = (\prod_{q_{i,j} \in \mathbb{P}^*} V_{i,j})^s$, and $B_4^* = (g^s)^\omega$.
2) Issue a H_3 query on $(\mathbb{P}^* \| B_1^* \| B_2^* \| B_3^* \| B_4^*)$ to obtain the tuple $(\mathbb{P}^* \| B_1^* \| B_2^* \| B_3^* \| B_4^*, \kappa_2, \eta)$. Finally computes $E^* = (g^s)^\eta$. Outputs the challenge ciphertext $CT^* = (\mathbb{P}^* \| B_1^* \| B_2^* \| B_3^* \| B_4^* \| E^*)$ to $\mathbb{AD}$. If $C = e(g,g)^{xyz}$, the challenge ciphertext is valid in relation to analysis in[8].

Phase 2: Nearly similar to phase 1 with stated constraints.

Guess: $\mathbb{AD}$ yields guess $\gamma' \in \{0,1\}$. If $\gamma' = \gamma$, $\mathbb{BD}$ outputs true (decides $C = e(g,g)^{xyz}$). Otherwise $\mathbb{BD}$ outputs false (decides $C \neq e(g,g)^{xyz}$).

From analysis in [9], the preceding simulation will terminate with negligible probability. Therefore we get the theorem. □

Theorem 2. *Suppose the MDDH assumption [34] holds in $(\mathbb{G}, \mathbb{G}_\tau)$ and H_1, H_2, H_3 are the TCR hash functions, then our CBe-BDS-OD is IND-AND-sAS-CCA2-sa secure in the random oracle model.*

Proof. We show that if there exist an adversary $\mathbb{AD}$ that can break the IND-AND-sAS-CCA2-sa security of the proposed CBe-BDS-OD scheme then we construct an algorithm $\mathbb{BD}$ using $\mathbb{AD}$ as a subroutine to solve MDDH problem. First MDDH challenger flips a fair coin δ and if δ=0, it sets $(g, X, Y, C) := (g, g^x, e(g,g)^y, e(g,g)^{xy})$; otherwise it sets $(g, X, Y, C) := (g, g^x, e(g,g)^y, (g,g)^c)$ where $x, y, c \in \mathbb{Z}_p$ are randomly chosen. The challenger then gives $\mathbb{BD}$ $(g, X, Y, C) := (g, g^x, e(g,g)^y, C)$. To be specific $\mathbb{BD}$ will act as a challenger with $\mathbb{AD}$ to play the following IND-AND-sAS-CCA2-sa game.

Init: Same as in Theorem 1.

Setup: $\mathbb{BD}$ selects two random elements $b \in_R \mathbb{Z}_p^*$ and $\upsilon_{i,j} \in_R \mathbb{Z}_p^*$ $(i \in [1,n], j \in [1, n_i])$. $\mathbb{BD}$ then computes $\mathbb{Y} = e(g,h)^b$ and $V_{i,j} = g^{\upsilon_{i,j}}$. It also maintains and responds to the TCR list $H_i^{List}(i \in \{1,.,3\})$ as in theorem 1. It chooses H_3 as the actual execution. It yields master secret key $\mathrm{MSK}{=}(b, \{\upsilon_{i,j}\}_{i \in [1,n], j \in [1,n_i]})$ which is known to $\mathbb{BD}$.

Phase 1: $\mathbb{AD}$ issues sequence of queries to which $\mathbb{BD}$ answers as follows:

Private key extraction oracle $\mathcal{OR}_{SK_F}$(F): $\mathbb{BD}$ receives attribute list F from $\mathbb{AD}$. $\mathbb{BD}$ then invokes $KeyGen(PK, MSK, F)$ to obtain SK_F the private key for the attribute list F.

Transformation key extraction oracle $\mathcal{OR}_{TK_F}(F)$: Upon receiving attribute list F and access structure $\mathbb{P}^*$, $\mathbb{BD}$ confirms if there exists tuple (F, SK_F, TK_F, RK_F) in the list $TK^{List_{tk}}$. If it exists, $\mathbb{BD}$ returns the equivalent TK_F. Otherwise, if $F \nvDash \mathbb{P}^*$, $\mathbb{BD}$ responds with a related private key SK_F by invoking $\mathcal{OR}_{SK_F}$ and the related server-aided key pair $(TK_F, RK_F) = ((K_1^{1/\sigma}, K_2^{1/\sigma}), \sigma)$ as in the actual scheme. if $F \vDash \mathbb{P}^*$, $\mathbb{BD}$ gets the related private key SK_F by querying $\mathcal{OR}_{SK_F}(F)$ with F^*, and the equivalent server-aided key pair $(TK_F, RK_F) = (((X^t)^b (X^{\sum_{q_{i,j} \in F} \upsilon_{i,j}})^\alpha, X^\alpha), \bullet)$ where t, α are randomly chosen elements to generate the private key SK_{F^*}, while

the • means the RK_F is undisclosed. In the case where $F \vDash P^*$, $\mathbb{BD}$ sets $\sigma = a$ that is undisclosed. Finally records (F, SK_F, TK_F, RK_F) in the list $TK^{List_{tk}}$.

Retrieving Key extraction oracle $\mathcal{OR}_{RK_F}$: Almost similar to $\mathcal{OR}_{TK_F}$. It responds with retrieving key RK_F.

Partial decryption query oracle $\mathcal{OR}_{PDec}$: Upon receiving the tuple (CT, F), $\mathbb{BD}$ verifies whether $F \vDash \mathbb{P}$ in CT. If it does not satisfies, it aborts with $\perp$. Otherwise it returns $PartialDecrypt_{out}(TK_F, CT)$. TK_F is the transformation key generated from $\mathcal{OR}_{TK_F}$.

Decryption query oracle $\mathcal{OR}_{Dec}$: Upon receiving the tuple (CT, F), $\mathbb{BD}$ verifies whether $F \vDash \mathbb{P}$ in CT. If it does not satisfies, it aborts with $\perp$. Otherwise it carries out the following actions:

- If the ciphertext is original, $\mathbb{BD}$ invokes $SK_F \leftarrow KeyGen(PK, MSK, F)$ and $m \leftarrow Decrypt(PK, SK_F, CT)$.
- If the ciphertext is server-aided, $\mathbb{BD}$ confirms if there exists tuple (F, SK_F, TK_F, RK_F) in the list $TK^{List_{tk}}$. If it does not exist, it aborts with $\perp$. Otherwise $\mathbb{BD}$ carries out the following actions:
  - if attribute list $F \nvDash P^*$ it employs the corresponding RK_F to retrieve the message m and sends it to $\mathbb{AD}$.
  - Else if attribute list $F \vDash P^*$, $\mathbb{BD}$ establishes whether $(\mathbb{P}, m, \lambda, s) \in H_1^{List}$ and also $(RC, \kappa_1) \in H_2^{List}$ in such way that $B_1 = \kappa_1 \oplus (m \| \lambda)$ where $RC = e(g,h)^{sb}$. If such kind of tuple does not exists it outputs $\perp$. Otherwise $\mathbb{BD}$ confirms whether $T'' = e(X,h)^{bH_1(\mathbb{P},m,\lambda)}$ if its true it outputs m. Otherwise aborts with $\perp$.

Challenge: $\mathbb{AD}$ submits two plaintext messages m_0^* and m_1^* of equal length from the message space $\{0,1\}^k$. $\mathbb{BD}$ flips a random coin γ to choose $\gamma \in \{0,1\}$ and encrypts m_γ^* as follows:
$\mathbb{BD}$ selects $\lambda^* \in \{0,1\}^k$, $B_1^* \in \{0,1\}^{2k}$ then sets

$$\begin{aligned} B_1^* &= H_2(Y^b) \oplus (m_\gamma^* \| \lambda^*) \\ T' &= C^{1/b}. \end{aligned}$$

If $C = e(g,g)^{xy}$, the challenge ciphertext is valid in relation to analysis in [8]

Phase 2: Nearly similar to phase 1 with the stated constraints.

Guess: $\mathbb{AD}$ outputs guess $\gamma' \in \{0,1\}$. If $\gamma' = \gamma$, $\mathbb{BD}$ outputs true (decides $C = e(g,g)^{xy}$). Otherwise $\mathbb{BD}$ outputs false(decides $C \neq e(g,g)^{xy}$).

□

From analysis in [9], the preceding simulation will terminate with negligible probability. Therefore we get the theorem.

7 Conclusion

In this paper, we proposed server-aided decryption in cloudlet for eHealth big data. We offloaded heavy computation to the server, in order to minimize pairing and reduce computation cost on the end user's (client) side. The validity of ciphertext is confirmed before partial decryption and final decryption is carried out by server and end user respectively. Further, the security analysis of the scheme has proven its authenticity in the random oracle model. We evaluated the performance of the scheme and the analysis of the output indicates that our proposal is IND-AND-sAS-CCA2 secure and practical and hence it's applicable to resource-constrained devices.

Acknowledgments

This work was supported by the Sichuan Province Science Technology Project [grant numbers 2014GZ0109 and 2015JY0178].

References

[1] A. Bahtovski and M. Gusev, "Cloudlet challenges," *Procedia Engineering*, vol. 69, pp. 704–711, 2014.

[2] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 321–334, 2007.

[3] S. Canard, N. Desmoulins, J. Devigne, and J. Traoré, "On the implementation of a pairing-based cryptographic protocol in a constrained device," in *Proceedings of the 5th International Conference on Pairing-Based Cryptography*, pp. 210–217, 2013.

[4] R. Canetti, S. Halevi, and J. Katz, "Chosen-ciphertext security from identity-based encryption," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 207–222, 2004.

[5] M. Chase, "Multi-authority attribute based encryption," in *Proceedings of the 4th Conference on Theory of Cryptography*, pp. 515–534, 2007.

[6] P. S. Chung, C. W. Liu, and M. S. Hwang, "A study of attribute-based proxy re-encryption scheme in cloud environments," *IInternational Journal Network Security*, vol. 16, no. 1, pp. 1–13, 2014.

[7] R. Cramer and V. Shoup, "Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack," *SIAM Journal on Computing*, vol. 33, no. 1, pp. 167–226, 2003.

[8] K. Emura, A. Miyaji, A. Nomura, K. Omote, and M. Soshi, "A ciphertext-policy attribute-based encryption scheme with constant ciphertext length," in *Proceedings of the 5th International Conference on Information Security Practice and Experience*, pp. 13–23, 2009.

[9] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," in *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pp. 537–554, 1999.

[10] Gartner, "Gartner's three-part definition of big data," 2013. `http://www.dataversity.net/gartners-three-part-definition-of-big-data/`.

[11] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of abe ciphertexts," in *Proceedings of the 20th USENIX Conference on Security*, pp. 34–34, 2011.

[12] J. Han, W. Susilo, Y. Mu, and J. Yan, "Privacy-preserving decentralized key-policy attribute-based encryption," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, pp. 2150–2162, Nov. 2012.

[13] P. Hu and H. Gao, "A key-policy attribute-based encryption scheme for general circuit from bilinear maps," *International Journal Network Security*, vol. 19, no. 5, pp. 704–710, 2017.

[14] J. Lai, R. H. Deng, C. Guan, and J. Weng, "Attribute-based encryption with verifiable outsourced decryption," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 8, pp. 1343–1354, 2013.

[15] C. C. Lee, P. S. Chung, and M. S. Hwang, "A survey on attribute-based encryption schemes of access control in cloud environments," *International Journal Network Security*, vol. 15, pp. 231–240, 2013.

[16] G. Lewis, S. Echeverria, S. Simanta, B. Bradshaw, and J. Root, "Tactical cloudlets: Moving cloud computing to the edge," in *IEEE Military Communications Conference*, pp. 1440–1446, Oct. 2014.

[17] J. Li, X. Huang, J. Li, X. Chen, and Y. Xiang, "Securely outsourcing attribute-based encryption with checkability," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 8, pp. 2201–2210, 2014.

[18] J. Li, W. Yao, Y. Zhang, H. Qian, and J. Han, "Flexible and fine-grained attribute-based data storage in cloud computing," *IEEE Transactions on Services Computing*, vol. PP, no. 99, 2016.

[19] K. Li and H. Ma, "Outsourcing decryption of multi-authority abe ciphertexts," *International Journal Network Security*, vol. 16, pp. 286–294, 2014.

[20] K. Liang, L. Fang, W. Susilo, and D. S. Wong, "A ciphertext-policy attribute-based proxy re-encryption with chosen-ciphertext security," in *Proceedings of the 5th International Conference on Intelligent Networking and Collaborative Systems*, pp. 552–559, 2013.

[21] S. Lin, R. Zhang, H. Ma, and M. Wang, "Revisiting attribute-based encryption with verifiable outsourced decryption," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 10, pp. 2119–2130, 2015.

[22] C. W. Liu, W. F. Hsien, C. C. Yang, and M. S. Hwang, "A survey of attribute-based access control with user revocation in cloud data storage," *International Journal Network Security*, vol. 18, no. 5, pp. 900–916, 2016.

[23] F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu, and B. Li, "Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications," *IEEE Wireless Communications*, vol. 20, pp. 14–22, June 2013.

[24] X. Liu, J. Ma, J. Xiong, and G. Liu, "Ciphertext-policy hierarchical attribute-based encryption for fine-grained access control of encryption data," *International Journal Network Security*, vol. 16, pp. 437–443, 2014.

[25] X. Mao, J. Lai, Q. Mei, K. Chen, and J. Weng, "Generic and efficient constructions of attribute-based encryption with verifiable outsourced decryption," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 5, pp. 533–546, 2016.

[26] I. Olaronke and O. Oluwaseun, "Big data in healthcare: Prospects, challenges and resolutions," in *Future Technologies Conference (FTC'16)*, pp. 1152–1157, Dec. 2016.

[27] H. Qian, J. Li, and Y. Zhang, "Privacy-preserving decentralized ciphertext-policy attribute-based encryption with fully hidden access structure," in *15th International Conference on Information and Communications Security*, pp. 363–372, 2013.

[28] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proceedings of the 24th Annual International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT'05)*, pp. 457–473, 2005.

[29] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, pp. 14–23, Oct. 2009.

[30] T. Verbelen, P. Simoens, F. D. Turck, and B. Dhoedt, "Cloudlets: Bringing the cloud to the mobile user," in *Proceedings of the Third ACM Workshop on Mobile Cloud Computing and Services*, pp. 29–36, 2012.

[31] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proceedings of the 14th International Conference on Practice and Theory in Public Key Cryptography Conference on Public Key Cryptography*, pp. 53–70, 2011.

[32] C. H. Wei, M. S. Hwang, A. Y. H. Chin, "A mutual authentication protocol for RFID", *IEEE IT Professional*, vol. 13, no. 2, pp. 20–24, Mar. 2011.

[33] M. Xiao, M. Wang, X. Liu, and J. Sun, "Efficient distributed access control for big data in clouds,"

in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS'15)*, pp. 202–207, Apr. 2015.

[34] C. Zuo, J. Shao, G. Wei, M. Xie, and M. Ji, "Cca-secure {ABE} with outsourced decryption for fog computing," *Future Generation Computer Systems*, vol. 78, pp. 730 – 738, 2018.

Biography

Kittur Philemon Kibiwott is currently pursuing Ph.D. degree at the University of Electronic Science and Technology of China (UESTC). He received Bsc degree in Computer Science from Periyar University (India) and Msc degree Computer Science from Bharathiar University (India). His research area of interest include cloud computing, cryptography and information security.

Zhang Fengli received her Ph.D. degree from the University of Electronic Science and Technology of China (UESTC) in 2007 and M.S. degree in 1986. She is currently a Professor at the University of Electronic Science and Technology of China (UESTC). She has published more than eighty papers in refereed international journals and conferences which more than 50 are indexed by SCI and EI. Her research area of interest include mobile data management and application, network security, database.

Omala A. Anyembe is currently a Ph.D candidate in the school of Computer Science and Engineering at the University of Electronic Science and Technology of China (UESTC). His research area of interest include cryptography, IoT and information security.

Daniel Adu-Gyamfi is currently a Ph.D candidate in the school of Information and Software Engineering at the University of Electronic Science and Technology of China (UESTC). His research area of interest include privacy preservation in cloud computing, Trajectory, cryptography and information security.