Securing Organization's Data: A Role-Based Authorized Keyword Search Scheme with Efficient Decryption

Nazatul Haque Sultan, Télécom SudParis, Institut Polytechnique de Paris
Maryline Laurent, Télécom SudParis, Institut Polytechnique de Paris
Vijay Varadharajan, The University of Newcastle

For better data availability and accessibility while ensuring data secrecy, organizations often tend to outsource their encrypted data to the cloud storage servers, thus bringing the challenge of keyword search over encrypted data. In this paper, we propose a novel authorized keyword search scheme using Role-Based Encryption (RBE) technique in a cloud environment. The contributions of this paper are multi-fold. First, it presents a keyword search scheme which enables only the authorized users, having proper assigned roles, to delegate keyword-based data search capabilities over encrypted data to the cloud providers without disclosing any sensitive information. Second, it supports a multi-organization cloud environment, where the users can be associated with more than one organization. Third, the proposed scheme provides efficient efficient decryption, conjunctive keyword search and revocation mechanisms. Fourth, the proposed scheme outsources expensive cryptographic operations in decryption to the cloud in a secure manner. Fifth, we have provided a formal security analysis to prove that the proposed scheme is semantically secure against Chosen Plaintext and Chosen Keyword Attacks. Finally, our performance analysis shows that the proposed scheme is suitable for practical applications.

CCS Concepts: •**Security and privacy** → **Privacy-preserving protocols; Management and querying of encrypted data;**

Additional Key Words and Phrases: Role-based encryption, role-based access control, searchable encryption, keyword search, outsourced decryption, provable security, cloud data privacy

1. INTRODUCTION

With the ever-increasing amount of digital information, individuals and organizations are now storing/outsourcing their data in the cloud to make use of features such as better accessibility, high availability, reduction of maintenance and initial investment costs [Ferrer et al. 2019]. However, with sensitive data stored in the cloud (e.g. see McAfee report [McAfee 2018]) and legal concerns (such as compliance to the European General Data Protection Regulation - GDPR[1]), security and privacy have become major issues in cloud data storage[2]. To preserve privacy and confidentiality of outsourced data in the cloud, a preferred technique that is often used is *encryption-before-outsourcing*. The encryption-before-outsourcing technique enables the data owners (i.e.

[1] https://ec.europa.eu/commission/priorities/justice-and-fundamental-rights/data-protection/

[2] In this paper, cloud represents the public cloud that provides storage facilities to the general public (i.e., individuals and organizations). In general, the public cloud is maintained by a third-party entity referred to as *Cloud Service Provider* [Mell and Grance].

This paper is supported in part by European Union's Horizon 2020 research and innovation programme under the grant agreement No 830892, project SPARTA.
Author's addresses: N. H. Sultan and Maryline Laurent, RST Department, Télécom SudParis, Institut Polytechnique de Paris, France; V. Varadharajan, Faculty of Engineering and Built Environment, Global Innovation Chair Professor at The University of Newcastle, Callaghan, Australia.

DOI: http://dx.doi.org/10.1145/0000000.0000000

entities owning the data) to outsource their sensitive data in the cloud in an encrypted form. As such, no entity including the cloud service provider can access the sensitive plaintext data without having access to proper decryption key. This, however, restricts data retrieval/search over encrypted data [Bösch et al. 2014]. A trivial solution is to download the whole encrypted database, and then perform the search operation locally after decryption. It is clear that this is not practical. An alternative approach is to allow the service provider to decrypt all the encrypted data so that it can perform search operation over the plaintext data. However, this violates data privacy.

Searchable Encryption (SE) has gained a considerable amount of interest from the research community to address the issue of searching over encrypted data [Han et al. 2016]. In SE, users delegate data search capabilities for some keywords over the encrypted data to a service provider without disclosing any useful information about the searched keywords and the actual content of the encrypted data. This process is also referred to as *keyword search*. Typically, in keyword search, data owners outsource their data in an encrypted form along with an encrypted index of keywords. Whenever a user wants to access data, the user sends the desired keywords in the form of trapdoors to the service provider. In return, the service provider uses the trapdoors to perform search over the encrypted indexes and sends the associated encrypted data, if there is a match between the keywords associated with the trapdoor and encrypted indexes.

Many works have been done in the area of keyword search, achieving search authorization in a coarse-grained way. That is, the users can search all the keywords using their secret keys [Hu et al. 2017]. However, this kind of authorization may disclose sensitive information. For example, Organization A outsources its data files to the cloud so that its employees can easily access them. Assume Organization A is a participant in a consortium with another Organization B and other organizations. Suppose, some files are associated with the keywords "Organization B" and "Project X" which are only allowed to be accessed by the Managers in the Organization A. In this case, if an adversary can search for the keywords "Organization B" and "Project X" and gets all the encrypted files associated with these two keywords. This will eventually reveal, without knowing the actual content, that Organization A and Organization B are collaborating on Project X, which may not be desirable.

To address this problem, several authorized keyword search schemes have been proposed for *multi-user settings* using different cryptographic techniques, e.g. Pairing-Based Encryption [Bao et al. 2008], Predicate Encryption [Li et al. 2011] and Attribute-Based Encryption [Sun et al. 2016], [Hu et al. 2017], where multiple users are able to perform keyword search operations based on some access policies. However, none of these techniques efficiently support hierarchies in an organization, where higher level authorities can inherit access rights of their subordinates. As such, all these schemes [Bao et al. 2008], [Li et al. 2011], [Sun et al. 2016], [Hu et al. 2017] are not able to reflect efficiently organization's policies and structures[3] [Marn Prez et al. 2017].

Role-Based Encryption (RBE) [Zhou et al. 2011], [Zhou et al. 2013], [Zhu et al. 2013] is an emerging cryptographic technique, which combines both properties of the traditional Role-Based Access Control (RBAC) [Sandhu et al. 1996] and cryptographic encryption methods, to achieve data access control over encrypted data. In RBE, the data owner encrypts data using a RBAC access policy defined over some roles[4], and any user having proper roles can derive the secret keys for decryption. Unlike the traditional RBAC method, RBE enables the data owners to define and enforce RBAC

[3] In an organization, typically employees are organized in a hierarchical way based on their responsibilities and qualification [Zhou et al. 2013].

[4] In an organization, roles are typically created based on job functions.

access policies on the encrypted data itself. This, in turn, reduces the dependency of the data owners on untrusted service provider for defining and enforcing access policies while sharing data with other authorized users. Moreover, similar to the RBAC, in RBE, roles can inherit access permissions from other roles [Zhou et al. 2013]. Hence, the roles can be organized in a hierarchical structure. This is one of the main advantages of RBE over other encryption mechanisms such as Attribute-Based Encryption [Bethencourt et al. 2007], [Goyal et al. 2006], as it can reflect closely a real-world organisation's policies and structure. The inheritance property of the RBE makes it more suitable for large scale organizations such as enterprises with a complex hierarchical structures [Zhou et al. 2013]. Therefore, RBE is a more suitable cryptographic technique for designing a keyword search mechanism compared with other cryptographic techniques such as the ABE.

RBE has been used to provide data access control in cloud environments over encrypted data [Zhou et al. 2013], [Zhu et al. 2013], [Marn Prez et al. 2017]. However, they mainly focus on a single organization cloud environment scenario, where users can have roles only in a single organization and hence can access data associated with only that organization. In many practical scenarios in a cloud environment, a data owner may want to share his/her data with users in several organizations having different roles. For example, a user may work as a researcher and doctor in a clinical research laboratory and hospital respectively. As such, the same user will hold roles in the clinical research laboratory and the hospital. The data owner can specify a RBAC access policy in such a way that only the users having the access privileges for the roles "Researcher" and "Doctor" can gain access to the actual content corresponding to the encrypted data.

This paper further investigates the aforementioned research gaps and proposes a novel keyword search scheme using the RBE technique where organizations outsource their data to a public cloud. The proposed scheme supports a multi-organization environment, where users can possess roles from more than one organization. It also enables the data owners to define and enforce RBAC access policies on encrypted data, thereby allowing any a user having authorized roles to perform a keyword search along with the ability to decrypt. The salient features of the proposed scheme are as follows:

(1) An authorized keyword search mechanism is proposed using RBE technique so that only the users possessing authorized roles can delegate keyword search capabilities over encrypted data to the public cloud.
(2) The proposed scheme supports multi-organization cloud environment, where a user can be associated with more than one organization, having one or more roles in different organizations.
(3) Conjunctive keyword search[5] functionality is supported without any significant overhead in the system.
(4) A user revocation mechanism has been introduced to revoke unintended users.
(5) An outsourced decryption mechanism is combined with the proposed scheme enabling the users to delegate most of the computationally expensive cryptographic operations to the public cloud, thereby reducing the overhead on the user-side.
(6) A formal security analysis of the proposed scheme has been given demonstrating that the scheme is secure against the Chosen Plaintext Attacks and the Chosen Keyword Attacks.

[5] In conjunctive keyword search, a user can search for multiple keywords in a single request [Ferrer et al. 2019].

(7) A performance analysis of the proposed scheme has been provided which shows that the proposed scheme is sufficiently efficient to be used in practical applications.

The organization of this paper is as follows: Section 2 presents a brief overview of some existing works related to the proposed scheme. Section 3 outlines the problem statement, where the system model, threat model, design and security goals, frameworks and security model of the proposed scheme are presented. Section 4 gives a brief overview of the role hierarchy, bilinear pairing properties, a group key distribution technique, and some mathematical assumptions, which will be used throughout this paper. Section 5 details the proposed scheme including an overview followed by its main construction. Section 6 presents a detailed security and performance analyses of the proposed scheme, and finally section 7 concludes this paper.

2. RELATED WORKS

This section presents a brief overview of some notable works in the keyword search area, including some cryptographic RBAC based data access control schemes.

2.1. Keyword Search over Encrypted Data

Data search over encrypted data has been extensively studied since the past decade. Song *et al.* presented the first practical symmetric key cryptography based searchable encryption scheme that can search full text over encrypted data [D. X. Song et al. 2000]. Leter, several searchable encryption schemes have been proposed, for various functionalities and security requirements, based on either symmetric key cryptography (SKC) [Curtmola et al. 2006], [Kamara et al. 2012], [Li et al. 2019], [Hoang et al. 2019], [Liu et al. 2018] or public-key cryptography (PKC) [Boneh et al. 2004], [Boneh and Waters 2007], [Sun et al. 2016], [Hu et al. 2017], [Miao et al. 2017], [Chaudhari and Das 2019].

In [Curtmola et al. 2006], Curtmola *et al.* proposed a SKC based keyword search scheme for *multi-user* settings[6], which can perform single keyword search. In [Kamara et al. 2012], Kamara *et al.* proposed a dynamic version of the scheme [Curtmola et al. 2006] that can add and delete files at any time efficiently. However, the scheme [Kamara et al. 2012] leaks significant information while performing update operation [Hoang et al. 2019]. In [Li et al. 2019], Li *et al.* proposed a SKC based forward search privacy scheme, which prevents any leakage of information about the past queries. Later on, in [Liu et al. 2018], Liu *et al.* proposed a keyword search scheme which enables the users to verify the search results against the dishonest servers. Although the SKC based keyword search schemes provides better efficiency in terms computation cost, PKC based keyword search schemes provide more flexible and expressive search queries [Sun et al. 2016].

Recently, many PKC based authorized keyword search schemes have been proposed based on Attribute-Based Encryption (ABE) [Sun et al. 2016], [Hu et al. 2017], [Miao et al. 2017], [Chaudhari and Das 2019], where any user having a qualified set of attributes that satisfy an access policy can perform search operation using some keywords. That is, these schemes provide authorized keyword search, which allows only intended users to do the search in multi-user settings. In [Sun et al. 2016], Sun *et al.* proposed a keyword search scheme using ABE technique. The scheme provides both single and conjunctive keyword search without introducing any additional overhead in the system. In [Hu et al. 2017], Hu *et al.* proposed another ABE based keyword

[6] Multi-user settings enable the data owners to authorize any number of users to perform keyword search operations.

search scheme for dynamic policy update, where the data owners can securely update the access policies using proxy re-encryption and secret sharing techniques. In [Miao et al. 2017], Miao *et al.* proposed an ABE based keyword search scheme for hierarchical data, which also supports conjunctive keyword search. Recently, in [Chaudhari and Das 2019], Chaudhari *et al.* proposed an authorized keyword search scheme using ABE, which hides the access policy from all the intended entities including the public cloud. However, all the aforementioned schemes do not support role hierarchy property and inheritance property.

2.2. Cryptographic RBAC based Data Access Control

A cryptographic RBAC based data access control mechanism integrates the traditional RBAC model with cryptographic encryption method to enforce RBAC access policy on encrypted data. It enables the data to be encrypted using RBAC access policy defined over some role(s). Any user, possessing the required role(s) satisfying the associated RBAC access policy is allowed to decrypt the data. Some notable works in this area are [Akl and Taylor 1983], [Lin and Hsu 2011], [Tang et al. 2016], [Chen and Tzeng 2017], [Pareek and Purushothama 2018], [Zhou et al. 2013], [Zhu et al. 2013], [Marn Prez et al. 2017], where [Akl and Taylor 1983], [Lin and Hsu 2011], [Tang et al. 2016], [Chen and Tzeng 2017], [Pareek and Purushothama 2018] are based on Hierarchical Key Assignment (HKA) method and [Zhou et al. 2013], [Zhu et al. 2013], [Marn Prez et al. 2017] are based on RBE method.

Access control using HKA method has been studied in the early 1980s. In [Akl and Taylor 1983], Akl *et al.* presented the first cryptographic hierarchical access control technique to solve the hierarchical multi-level security problem, where authorized users are allowed to possess different access privileges. The users are grouped into disjoint sets (or classes) and form a hierarchical structure of classes. Each class is assigned with a unique encryption key and a public parameter in such a way that a higher-level class can derive encryption keys of any lower-level classes using its own encryption key and some public parameters. Later on, several other hierarchical access control schemes have been proposed using different techniques, e.g. [Lin and Hsu 2011], [Tang et al. 2016], [Chen and Tzeng 2017], [Pareek and Purushothama 2018]. However, the main drawback of the HKA schemes is the high complexity in setting up the encryption keys for a large set of users [Zhou et al. 2013]. Also, the user revocation is a challenging task, as all the encryption keys that are known to the revoked users, and their related public parameters need to update per user revocation which may incur a high overhead on the system.

In [Zhou et al. 2013], Zhou *et al.* proposed the first RBE scheme for data sharing in an untrusted hybrid cloud environment. In [Zhou et al. 2013], the ciphertexts and secret keys of the users are constant in size. This scheme also offers user revocation capability. In [Zhu et al. 2013], Zhu *et al.* proposed another RBE scheme. In this scheme, the ciphertext size linearly increases with the number of roles. Recently, in [Marn Prez et al. 2017], Perez *et al.* proposed a data-centric RBAC based data access control mechanism for cloud storage systems using the concept of proxy re-encryption and identity-based encryption techniques. To share data with the authorized users, the data owner generates proxy re-encryption keys based on some RBAC access policies and keeps the re-encryption keys along with the ciphertexts in the cloud storage servers. When an authorized user accesses the ciphertext, the service provider re-encrypts the ciphertext using the proxy re-encryption keys based on a RBAC access policy. However, none of [Zhou et al. 2013], [Zhu et al. 2013], [Marn Prez et al. 2017] support multi-organization cloud storage systems, where a user can possess roles from more than one organization. Moreover, [Zhou et al. 2013], [Zhu et al. 2013], [Marn Prez et al. 2017] do not address keyword search functionality.

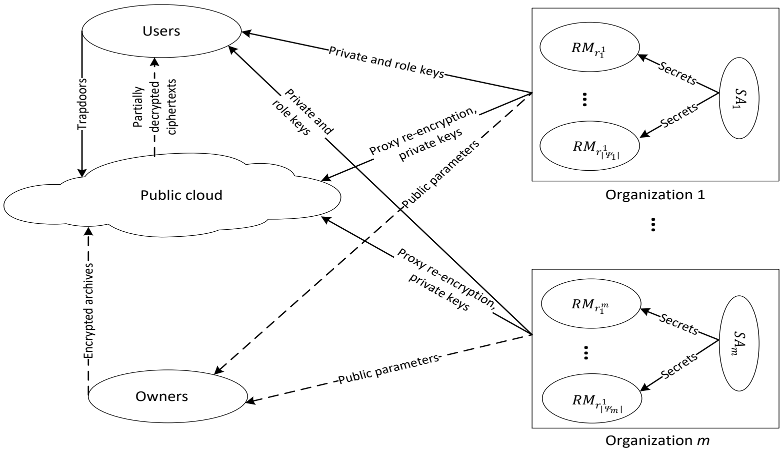


Fig. 1: Proposed System Model

3. PROBLEM STATEMENT

This section presents the *System Model*, *Threat Model*, *Design and Security Goals*, *Framework*, and *Security Models* of the proposed scheme.

3.1. System Model

Figure 1 shows the proposed system model, where the doted and dark lines represent public channel and secure-channel such as SSL (Secure Sockets Layer) respectively. It comprises five entities, namely, *System Authorities*, *Role-Managers*, *Data Owners*, *Users*, and *Public Cloud* having the following responsibilities:

- *System Authority (SA)*: Each organization has one SA, which maintains the role hierarchy of that organization. It generates system public parameters and master secrets for the organization. SA also maintains all the role-managers that are associated with the organization, and it issues secret keys for each role-manager. In addition, SA issues private and public keys for all the registered users. Further, it issues private, public and proxy re-encryption keys to the public cloud. Moreover, SA is responsible for revoking users from the system when needed.
- *Role-Manager (RM)*: It is an entity of an organization which manages the role(s). Note that, the roles are assigned by the SA. In addition, it also issues and manages role-keys for the users.
- *Data Owners (owners)*: It is an entity who owns the data and wants to outsource his/her data to the public cloud. An owner first encrypts data using a RBAC access policy before outsourcing to the public cloud. The owner first encrypts a plaintext data using a random secret key by following any secure symmetric key encryption algorithm, e.g., *Advanced Encryption Standard* (AES). Afterward, the owner chooses a set of keywords associated with the plaintext data and encrypts those keywords along with the random key using the chosen RBAC access policy. The owner then combines all the ciphertexts into one archive and outsources it to the cloud storage servers.
- *Users*: It is an entity who wants to access the outsourced data. Each user must register with SA(s) to receive private and public keys associated with the organization(s) from which he/she wants to access data. Also, a user receives a unique role-key for each role he/she possesses from the respective role-manager. When a user wants to

access data, the user computes a trapdoor using his/her private keys, role-keys, the desired keyword(s) and sends it to the public cloud.

- *Public Cloud*: It is a third-party entity which manages the cloud storage servers. The main responsibility of the public cloud is to store owners' encrypted data. Moreover, it is also responsible for performing keyword search operation over the encrypted data. It is assumed that the public cloud correctly performs search operations using the received trapdoors if and only if the requested user has proper roles. It is also assumed that it partially decrypts all the ciphertexts that have a matching keyword(s) with the trapdoors.

3.2. Threat Model

Public cloud is considered as an honest-but-curious entity. That is, public cloud honestly performs all the assigned tasks, but it may try to gain additional privacy information from the data available to it. The users may be malicious, and they may try to collude among themselves to gain access to the data beyond their access privileges. The users, having insufficient access rights, may also try to collude with the public cloud for gaining access to the data beyond their access rights. It is assumed that all the SAs and RMs are fully trusted entities. The threat model is supplemented by a Security Model in Section 3.5.

3.3. Design and Security Goals

The proposed scheme aims to achieve the following functionality and security goals.

Functionality Goals: The proposed scheme should provide the following functionalities.

(1) *Authorized Keyword Search*: Only the users, having proper roles according to the defined RBAC access policy, are authorized to perform keyword search operations over the encrypted data. That is, any unintended users should not get access to the encrypted (outsourced) data.
(2) *Role-Based Data Sharing*: Only the users, possessing the proper roles according to the defined RBAC access policy, can have access to the plaintext data through the decryption operation.
(3) *Role Management by Multiple organizations*: The roles assigned to users can be managed by more than one organization and can be simultaneously used for data sharing and keyword search operations.
(4) *Conjunctive Keyword Search*: Users can search for multiple keywords using a single search request.
(5) *Outsourced Decryption*: Users can delegate most of the computationally expensive operation to the public cloud without disclosing any sensitive information.
(6) *Prior Authentication*: The public cloud can authenticate a user before performing the costly keyword search and outsourced decryption operations for the user.
(7) *Revocation*: Revocation is supported in two following ways:
    - *Complete user revocation*: SA can prevent unintended users from accessing its data.
    - *Role-level user revocation*: SA can revoke one or more roles of a user. The idea is that the revoked user can no longer use the revoked roles for accessing data, while the same user should be able to access data using his/her non-revoked roles if they are qualified enough according to the RBAC access policy.

Security Goals: The proposed scheme should fulfil the following security requirements:

Table I: NOTATIONS

Notation	Description
q	a large prime number
$\mathbb{G}_1, \mathbb{G}_T$	two cyclic multiplicative groups of order q
$H_1(.), H_2(.)$	hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ and $H_2 : \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$
Φ	set of system authorities in the system
m	total number of system authorities in the system
Ψ_k	set of roles associated with a role hierarchy of the k^{th} system authority
Γ	set of all roles associated with a ciphertext
Γ_Φ	system authorities associated with a ciphertext
$\mathbb{S}_{\mathtt{ID_u}}$	set of roles associated with the user $\mathtt{ID_u}$
$r_{k,i}$	i^{th} role managed by k^{th} system authority
$\mathbb{R}_{k,i}$	the set of ancestor roles of $r_{k,i}$
$\mathtt{ID_u}$	unique identity of the u^{th} user
$\mathtt{ID_c}$	unique identity of the public cloud
$\mathtt{RM}_{\mathtt{r}_\mathtt{i}^\mathtt{k}}$	role-manager which manages role r_i^k
ts	current timestamp

(1) *Data Confidentiality*: Any entity, including the public cloud should not be able to access the plaintext data unless they have proper roles satisfying the defined RBAC access policy. This security notion can be captured by *Semantic Security*. This security notion is also referred to as *Indistinguishability against Chosen Plaintext Attack (IND-CPA)*.

(2) *Keyword Secrecy*: Using unqualified search requests or trapdoors, any entity including the public cloud should not be able to learn any useful information about the plaintext keywords associated with the encrypted data. Similarly, any outsider (neither the requesting user nor the public cloud) should be able to learn any useful information about the keywords from the trapdoors. These two security notions can be captured by *Keyword Semantic Security*. This security notion is also referred to as *Indistinguishability against Chosen Keyword Attack (IND-CKA)*.

(3) *Forward and Backward Secrecy*: Forward secrecy represents that any new user having qualified roles should be able to decrypt the ciphertexts which are encrypted before he/she joined the system. Backward secrecy represents that a revoked user should not be able to decrypt the ciphertexts which are published after his/her revocation using the revoked roles.

(4) *Resistance against Replay Attacks*: If one or more valid trapdoor is exposed to an adversary, the adversary should not be able to launch replay attacks. Many recent keyword search schemes, e.g., [Sun et al. 2016], [Hu et al. 2017] are susceptible to replay attacks if the trapdoors are exposed, as the adversary can re-use the exposed trapdoors using a fresh random number each time she/he wants to perform a keyword search.

3.4. Framework

Broadly the proposed scheme is divided into nine main phases, namely, *System Setup, Management of Roles, Public Cloud Key Generation, New User Enrolment, Role Assignment, Data Encryption, Trapdoor Generation, Data Search*, and *Decryption*. SAs initiate the *System Setup* phase to generate mutually agreed public parameters and master secret through the SYSTEMSETUP algorithm. SA performs the *Manage of Role* phase to initialize its role hierarchy and generates role related parameters (both public and secret parameters). It also generates proxy re-encryption keys for the public cloud. It consists of the MANAGEROLE algorithm. SA generates private and public keys for the public cloud in the *Public Cloud Key Generation* phase using the PUBCLOUDKEYGEN algorithm. In the *New User Enrolment* phase, SA mainly issues private and public keys for each registered users through the USERPRIVKEYGEN algorithm. Role-managers perform *Role Assignment* phase, where they assign roles in the form of role-keys to the users based on their responsibilities and profile in the organization. It consists of the USERROLEKEYGEN algorithm. In the *Data Encryption* phase, the owner encrypts data and associated keywords using a RBAC access policy. It consists of the ENC algorithm. To perform keyword search as well as outsourced decryption, the users generate trapdoors in the *Trapdoor Generation* phase using the TRAPGEN algorithm. The public cloud performs the *Data Search* phase, which consists of AUTHENTICATION, KEYSEARCH, and PARTIALDEC algorithms. In the AUTHENTICATION, the public cloud authenticates the requesting user and checks freshness of the keyword search request (i.e., trapdoor) to prevent any replay attacks. In the KEYSEARCH, the public cloud performs keyword search operation on the encrypted data using the received trapdoor. In the PARTIALDEC, the public cloud performs outsourced decryption operations. In this algorithm, the public cloud partially decrypts the ciphertexts which are returned by the KEYSEARCH algorithm. Finally, the user performs *Decryption* phase to decrypt all the partially decrypted ciphertexts received from the public cloud. This phase comprises DEC algorithm. A brief overview of the different algorithms of these phases are explained next. The notations used in this paper are shown in Table I.

— SYSTEMSETUP $\big((\mathtt{PP}, \{\mathtt{MS}_\mathtt{k}\}_{\forall \mathtt{k} \in \Phi}) \leftarrow 1^\Lambda\big)$: It takes a security parameter Λ as input. It outputs public parameter $\mathtt{PP}$ and master secret $\mathtt{MS}_\mathtt{k}$ for each SA in the system.

— MANAGEROLE $\Bigg(\Bigg(\mathtt{RP}_\mathtt{k}, \{\mathbb{PK}_{r_i^k}\}_{\forall r_i^k \in \Psi_k}, \{\mathtt{RS}_{\mathtt{r}_\mathtt{i}^\mathtt{k}}\}_{\forall r_i^k \in \Psi_k}, \left\{\left\{\mathtt{PKey}_{\mathtt{r}_\mathtt{i}^\mathtt{k}}^{\mathtt{r}_\mathtt{w}^\mathtt{k}}\right\}_{\forall r_w^k \in \mathbb{R}_{r_i^k} \setminus \{r_i^k\}}\right\}_{\forall r_i^k \in \Psi_k} \Bigg) \leftarrow \Big(\mathcal{H}, \mathtt{PP}\Big)\Bigg)$: It takes a role hierarchy $\mathcal{H}$ and public parameter $\mathtt{PP}$ as input. It outputs role secret parameter $\mathtt{RP}_\mathtt{k}$, and for each role $r_i^k \in \Psi_k$, it outputs the role public key $\mathbb{PK}_{r_i^k}$, role secret $\mathtt{RS}_{\mathtt{r}_\mathtt{i}^\mathtt{k}}$ and proxy re-encryption keys $\mathtt{PKey}_{\mathtt{r}_\mathtt{i}^\mathtt{k}}^{\mathtt{r}_\mathtt{w}^\mathtt{k}}$.

— PUBCLOUDKEYGEN $\big((\mathtt{Priv}_\mathtt{c}^\mathtt{k}, \mathtt{Pub}_\mathtt{c}^\mathtt{1k}, \mathtt{Pub}_\mathtt{c}^\mathtt{2k}) \leftarrow (\mathtt{PP}, \mathtt{MS}_\mathtt{k}, \mathtt{ID}_\mathtt{c})\big)$: It takes public parameter $\mathtt{PP}$, master secret $\mathtt{MS}_\mathtt{k}$ and identity $\mathtt{ID}_\mathtt{c}$ of the public cloud as input. It outputs a private key $\mathtt{Priv}_\mathtt{c}^\mathtt{k}$ and two public keys $(\mathtt{Pub}_\mathtt{c}^\mathtt{1k}, \mathtt{Pub}_\mathtt{c}^\mathtt{2k})$ for the public cloud.

— USERPRIVKEYGEN $\big(\big(\mathtt{SK}_{\mathtt{ID}_\mathtt{u}}^\mathtt{k}, \mathtt{Pub}_{\mathtt{ID}_\mathtt{u}}^\mathtt{k}, \mathtt{US}_{\mathtt{ID}_\mathtt{u}}\big) \leftarrow (\mathtt{MS}_\mathtt{k}, \mathtt{PP}, \mathtt{ID}_\mathtt{u})\big)$: It takes master secret $\mathtt{MS}_\mathtt{k}$, public parameter $\mathtt{PP}$, and unique identity of a user $\mathtt{ID}_\mathtt{u}$ as input. It outputs a secret key $\mathtt{SK}_{\mathtt{ID}_\mathtt{u}}^\mathtt{k}$, a public key $\mathtt{Pub}_{\mathtt{ID}_\mathtt{u}}^\mathtt{k}$ and a user secret $\mathtt{US}_{\mathtt{ID}_\mathtt{u}}$ for the user $\mathtt{ID}_\mathtt{u}$.

— USERROLEKEYGEN $((\mathtt{RK}_{\mathtt{r}_\mathtt{x}^\mathtt{k}}^{1,\mathtt{u}}, \mathtt{RK}_{\mathtt{r}_\mathtt{x}^\mathtt{k}}^{2,\mathtt{u}}) \leftarrow (\mathtt{PP}, \mathtt{US}_{\mathtt{ID}_\mathtt{u}}, \mathtt{RS}_{\mathtt{r}_\mathtt{x}^\mathtt{k}}, t_{r_x^k}))$: It takes public parameter $\mathtt{PP}$, user secret $\mathtt{US}_{\mathtt{ID}_\mathtt{u}}$, and role secret $\mathtt{RS}_{\mathtt{r}_\mathtt{x}^\mathtt{k}}$, role related secret $t_{r_x^k} \in \mathbb{Z}_q^*$ of r_x^k as input. It outputs two role-keys $(\mathtt{RK}_{\mathtt{r}_\mathtt{x}^\mathtt{k}}^{1,\mathtt{u}}, \mathtt{RK}_{\mathtt{r}_\mathtt{x}^\mathtt{k}}^{2,\mathtt{u}})$ associated with the role r_x^k for the user $\mathtt{ID}_\mathtt{u}$.

— ENC $\Big(\mathbb{CT} \leftarrow (\mathtt{PP}, \mathtt{Pub}_\mathtt{c}^\mathtt{1k}, \mathtt{Pub}_\mathtt{c}^\mathtt{2k}, \mathtt{M}, \mathbb{W}, \Gamma, \Gamma_\Phi)\Big)$: It takes public parameter $\mathtt{PP}$, both the public keys $(\mathtt{Pub}_\mathtt{c}^\mathtt{1k}, \mathtt{Pub}_\mathtt{c}^\mathtt{2k})$ of the public cloud, actual plaintext message $\mathtt{M}$, keyword set $\mathbb{W}$ (associated with the actual plaintext message $\mathtt{M}$), a RBAC access policy Γ, and a set Γ_Φ of SAs which are associated with Γ as input. It outputs a ciphertext $\mathbb{CT}$.

— TRAPGEN $((\mathtt{Trap}, v) \leftarrow (\{\mathtt{RK}_{\mathtt{r}_\mathtt{x}^\mathtt{k}}^{1,\mathtt{u}}, \mathtt{RK}_{\mathtt{r}_\mathtt{x}^\mathtt{k}}^{2,\mathtt{u}}\}_{\forall r_x^k \in \mathbb{S}_{\mathtt{ID}_\mathtt{u}}}, \mathtt{SK}_{\mathtt{ID}_\mathtt{u}}^\mathtt{k}, \mathbb{S}_{\mathtt{ID}_\mathtt{u}}, w))$: It takes both the role-keys $(\mathtt{RK}_{\mathtt{r}_\mathtt{x}^\mathtt{k}}^{1,\mathtt{u}}, \mathtt{RK}_{\mathtt{r}_\mathtt{x}^\mathtt{k}}^{2,\mathtt{u}})$, secret key $\mathtt{SK}_{\mathtt{ID}_\mathtt{u}}^\mathtt{k}$, user role set $\mathbb{S}_{\mathtt{ID}_\mathtt{u}}$ of a user $\mathtt{ID}_\mathtt{u}$, and keyword w as input. It outputs a trapdoor $\mathtt{Trap}$ and a random number $v \in \mathbb{Z}_q^*$.

- AUTHENTICATION $\Big((V_3^1/\perp) \leftarrow \Big(\{\texttt{Priv}_\texttt{c}^\texttt{k}\}_{\forall \texttt{k}\in\Gamma_\Phi}, \texttt{Trap}, \texttt{Pub}^\texttt{k}_{\texttt{ID}_\texttt{u}}, \texttt{ID}_\texttt{u}, ts'\Big)\Big)$: It takes private keys of the public cloud $\texttt{Priv}_\texttt{c}^\texttt{k}$ issues by all the system authorities in the set Γ_Φ, trapdoor $\texttt{Trap}$, public key $\texttt{Pub}^\texttt{k}_{\texttt{ID}_\texttt{u}}$ of a user $\texttt{ID}_\texttt{u}$, identity $\texttt{ID}_\texttt{u}$ of the user, and current timestamp ts' as input. If the user $\texttt{ID}_\texttt{u}$ is legitimate and the trapdoor was not previously issued, it outputs V_3^1 for a successful authentication. Otherwise, it outputs $\perp$ which represents either an unsuccessful authentication or an invalid trapdoor.
- KEYSEARCH $((\mathbb{CT}/\perp) \leftarrow (\mathbb{CT}, \texttt{Trap}, V_1^3))$: It takes trapdoor $\texttt{Trap}$, V_1^3 and a ciphertext $\mathbb{CT}$ as input. It outputs the ciphertext $\mathbb{CT}$ if and only if for all $r_i^k \in \Gamma$ there is $r_x^k \in \mathbb{S}_{\texttt{ID}_\texttt{u}}$ such that $r_x^k \in \mathbb{R}_{r_i^k}$ and the keyword w associated with the trapdoor has a match with a keyword associated with the ciphertext $\mathbb{CT}$. Otherwise, it outputs $\perp$, which represents an unsuccessful search operation.
- PARTIALDEC $\big(\mathbb{CT}' \leftarrow (\mathbb{CT}, \texttt{Trap}, \{\texttt{Priv}_\texttt{c}^\texttt{k}\}_{\forall k\in\Gamma_\Phi}, \mathbb{S}_{\texttt{ID}_\texttt{u}})\big)$: It takes the ciphertext $\mathbb{CT}$, trapdoor $\texttt{Trap}$, private keys $\texttt{Priv}_\texttt{c}^\texttt{k}$ of the public cloud associated with the system authorities in Γ_Φ, and user role set $\mathbb{S}_{\texttt{ID}_\texttt{u}}$ as input. It outputs a partially decrypted ciphertext $\mathbb{CT}'$.
- FULLDEC$(\texttt{M} \leftarrow (\mathbb{CT}', \texttt{Priv}_{\texttt{ID}_\texttt{u}}, v))$: It takes the partially decrypted ciphertext $\mathbb{CT}'$, user private key $\texttt{Priv}_{\texttt{ID}_\texttt{u}}$, and v as input and outputs the actual plaintext message $\texttt{M}$.

3.5. Security Model

The two games, namely, *Semantic Security against Chosen Plaintext Attack* (IND-CPA) and *Semantic Security against Chosen Keyword Attack* (IND-CKA) are used to define the security model of the proposed scheme. These two games are defined next.

3.5.1. Semantic Security against Chosen Plaintext Attack. The semantic security of the proposed scheme defined on *Chosen Plaintext Attack* (CPA) security under *Selective-ID Model*[7]. The CPA security can be illustrated using the following security game IND-CPA between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}_1$.

INIT Adversary $\mathcal{A}_1$ sends a challenged role set Γ^*, a keyword w and two identities $\texttt{ID}_\texttt{u}^*, \texttt{ID}_\texttt{c}^*$ to the challenger $\mathcal{C}$.

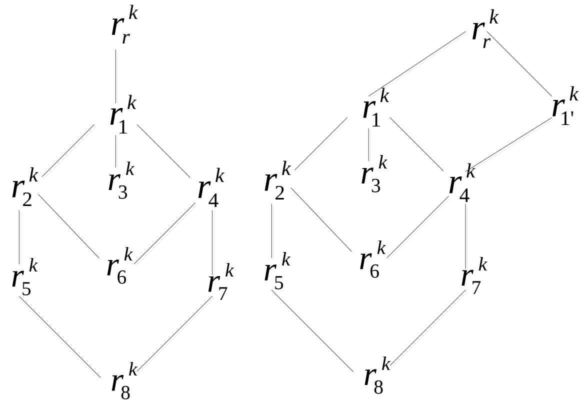
SETUP Challenger runs the SYSTEMSETUP algorithm to generate public parameters and master secrets. Challenger $\mathcal{C}$ generates role public keys, role secrets and proxy re-encryption keys using the MANAGEROLE algorithm. It also generates public and private keys using the PUBCLOUDKEYGEN and USERPRIVKEYGEN algorithms. Challenger $\mathcal{C}$ sends the public parameter, role public keys, proxy re-encryption keys, public and private keys to the adversary $\mathcal{A}_1$. It keeps the master secret and role secrets in a secure place.

PHASE 1 Adversary $\mathcal{A}_1$ submits a role set $\mathbb{S}^*$ to the challenger $\mathcal{C}$ for role-keys so that there exits at least one role $r_x^k \in \mathbb{S}^*$ such that $r_x^k \notin \mathbb{R}_{r_i^k}$, where $r_i^k \in \Gamma^*$. Challenger $\mathcal{C}$ runs the USERROLEKEYGEN algorithm to generate role-keys for the adversary $\mathcal{A}_1$. Adversary $\mathcal{A}_1$ can send queries for the role-keys to the challenger $\mathcal{C}$ by polynomially many times.

CHALLENGE When adversary $\mathcal{A}_1$ decides that PHASE 1 is over, it submits two equal length messages $\texttt{K}_0$ and $\texttt{K}_1$, which were not challenged before, to the challenger $\mathcal{C}$. Challenger $\mathcal{C}$ flips a random binary coin ω and encrypts message $\texttt{K}_\omega$ using the ENC algorithm for the challenged role set Γ^*. Challenger $\mathcal{C}$ sends the encrypted message of $\texttt{K}_\omega$ to adversary $\mathcal{A}_1$.

PHASE 2 Same as **PHASE 1**.

[7] In the Selective-ID security model, the adversary must submit a set of challenged roles before starting the security game. This is essential in our security proof to set up the role public key (please refer Section 6.1 for more details).



(a) Role Hierarchy 1

(b) Role Hierarchy 2

Fig. 2: Sample Role Hierarchy (RH)

Guess Adversary $\mathcal{A}_1$ outputs a guess ω' of ω. The advantage of winning this game for adversary $\mathcal{A}_1$ is $Adv_{\mathcal{A}_1}^{IND-CPA} = \left|Pr[\omega' = \omega] - \frac{1}{2}\right|$.

Definition 3.1. The proposed scheme is secure against chosen plaintext attack if $Adv_{\mathcal{A}_1}^{IND-CPA}$ is negligible for any polynomial time adversary $\mathcal{A}_1$.

3.5.2. Semantic Security against Chosen Keyword Attack. The semantic security of the proposed keyword search scheme defined on Chosen Keyword Attack (CKA) security under the same *Selective ID Model* as described in Section 3.5.1. The CKA security can be demonstrated using the following security game IND-CKA between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}_2$.

Init Adversary $\mathcal{A}_2$ sends a set of challenged roles Γ^* and two identities $\mathtt{ID}_\mathtt{u}^*, \mathtt{ID}_\mathtt{c}^*$ to the challenger $\mathcal{C}$.

Setup Challenger runs the SystemSetup algorithm to generate public parameters and master secrets. Challenger $\mathcal{C}$ generates role public keys, role secrets and proxy re-encryption keys using the ManageRole algorithm. It also generates public and private keys using PubCloudKeyGen algorithm and a public key using UserPrivKeyGen algorithm. Challenger $\mathcal{C}$ sends the public parameter, role public keys, proxy re-encryption keys, public and private keys to the adversary $\mathcal{A}_2$. It keeps the master secret and role secrets in a secure place.

Phase 1 Adversary $\mathcal{A}_2$ submits a set of roles $\mathbb{S}^*$ and a keyword w to the challenger $\mathcal{C}$ so that there exits at least one role $r_x^k \in \mathbb{S}^*$ such that $r_x^k \notin \mathbb{R}_{r_i^k}$, where $r_i^k \in \Gamma^*$. Challenger initiates the TrapGen algorithm to generate a trapdoor for the adversary $\mathcal{A}_2$. Finally, challenger $\mathcal{C}$ sends the generated trapdoor to the adversary $\mathcal{A}_2$. Afterwards, adversary $\mathcal{A}_2$ can send queries for the trapdoor to the challenger $\mathcal{C}$ by polynomially many times.

Challenge When adversary $\mathcal{A}_2$ decides that Phase 1 is completed, it submits two equal length keywords w_0 and w_1, which were not challenged before, to the challenger $\mathcal{C}$. Challenger $\mathcal{C}$ flips a binary coin ω and encrypts keyword w_ω using the Enc algorithm for the challenged role set Γ^*. Challenger $\mathcal{C}$ sends the encrypted ciphertext of w_ω to the adversary $\mathcal{A}_2$.

Phase 2 Same as **Phase 1**.

GUESS Adversary $\mathcal{A}_2$ outputs a guess ω' of ω. The advantage of winning this game for adversary $\mathcal{A}_2$ is $Adv_{\mathcal{A}_2}^{IND-CKA} = \left|Pr[\omega' = \omega] - \frac{1}{2}\right|$.

Definition 3.2. The proposed scheme is secure against the chosen keyword attack if $Adv_{\mathcal{A}_2}^{IND-CKA}$ is negligible for any polynomial time adversary $\mathcal{A}_2$.

4. PRELIMINARIES

This section presents an overview of a role hierarchy and bilinear pairing. It also presents an overview of a group key distribution mechanism and a mathematical assumption which is used in this paper.

4.1. Role Hierarchy notations

In the proposed scheme, roles are organized in a hierarchy where ancestor roles can inherit access privileges of its descendant roles. Figure 2 shows two sample role hierarchies, namely Role Hierarchy 1 (Figure 2a) and Role Hierarchy 2 (Figure 2b). We consider Role Hierarchy 1 (Figure 2a) as an example to define the following notations of a role hierarchy.

- r_r^k: root role of a role hierarchy. We assume that in any role hierarchy there can be only one root role.
- Ψ_k: set of all roles in the role hierarchy. For example, $\Psi_k = \{r_r^k, r_1^k, r_2^k, r_3^k, r_4^k, r_5^k, r_6^k, r_7^k, r_8^k\}$
- $\mathbb{R}_{r_i^k}$: ancestor set of the role r_i^k. For example, $\mathbb{R}_{r_8^k} = \{r_r^k, r_1^k, r_2^k, r_4^k, r_5^k, r_6^k, r_7^k, r_8^k\}, \mathbb{R}_{r_5^k} = \{r_r^k, r_1^k, r_2^k, r_5^k\}$ and $\mathbb{R}_{r_6^k} = \{r_r^k, r_1^k, r_2^k, r_4^k, r_6^k\}$.

4.2. Bilinear Pairing

Let $\mathbb{G}_1$ and $\mathbb{G}_T$ be two cyclic multiplicative groups of order q. Let g be a generator of $\mathbb{G}_1$. The bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ has the following properties:

- *Bilinear*: $\hat{e}\left(g^a, g^b\right) = \hat{e}\left(g, g\right)^{ab}$, $\forall g \in \mathbb{G}_1$ and $\forall (a, b) \in \mathbb{Z}_q^*$
- *Non-degenerate*: $\hat{e}\left(g, g\right) \neq 1$
- *Computable*: $\hat{e}(g, g)$ is efficiently computable for all $g \in \mathbb{G}_1$

4.3. Group Key Distribution

In [Burmester and Desmedt 2005], Burmester *et al.* proposed a two round group key distribution scheme using the concept of Diffie-Hellman assumption. Their scheme works as follows:

Let $\mathbb{U} = \{\mathtt{ID}_1, \mathtt{ID}_2, ..., \mathtt{ID}_\mathtt{n}\}$ be the group of n users. Suppose the users are arranged into a cycle. To compute a group key among the users, each user $\mathtt{ID}_\mathtt{i} \in \mathbb{U}$ selects a random secret number $a_i \in \mathbb{Z}_q^*$ and broadcasts $x_i = g^{a_i}$ where g is a generator of group $\mathbb{G}_1$. Afterward, it publishes $X_i = (\frac{x_{i+1}}{x_{i-1}})^{a_i}$. Finally, each user $\mathtt{ID}_\mathtt{i}$ in the group computes a common key $\mathtt{CK} = g^{a_1 \cdot a_2 + a_2 \cdot a_3 + ... + a_n \cdot a_1}$ without knowing others' secrets and without disclosing the common key to any other unintended entities.

4.4. Decisional Bilinear Diffie-Hellman (DBDH)

Let $\mathbb{G}_1$ and $\mathbb{G}_T$ be two cyclic multiplicative groups of order q. Let g be a generator of $\mathbb{G}_1$ and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ be an efficiently computable non-degenerate bilinear map. The Decisional Bilinear Diffie-Hellman (DBDH) Assumption is defined as follows: No probabilistic polynomial time adversary is able to distinguish the tuples

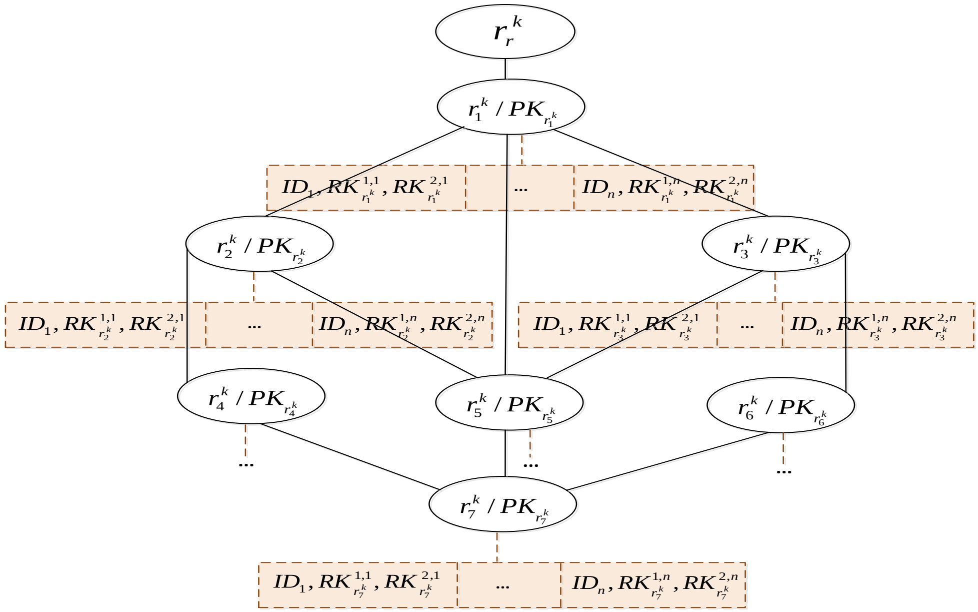

r_r^k
r_1^k / $PK_{r_1^k}$
$ID_1, RK^{1,1}_{r_1^k}, RK^{2,1}_{r_1^k}$
...
$ID_n, RK^{1,n}_{r_1^k}, RK^{2,n}_{r_1^k}$
r_2^k / $PK_{r_2^k}$
r_3^k / $PK_{r_3^k}$
$ID_1, RK^{1,1}_{r_2^k}, RK^{2,1}_{r_2^k}$
...
$ID_n, RK^{1,n}_{r_2^k}, RK^{2,n}_{r_2^k}$
$ID_1, RK^{1,1}_{r_3^k}, RK^{2,1}_{r_3^k}$
...
$ID_n, RK^{1,n}_{r_3^k}, RK^{2,n}_{r_3^k}$
r_4^k / $PK_{r_4^k}$
r_5^k / $PK_{r_5^k}$
r_6^k / $PK_{r_6^k}$
...
...
...
r_7^k / $PK_{r_7^k}$
$ID_1, RK^{1,1}_{r_7^k}, RK^{2,1}_{r_7^k}$
...
$ID_n, RK^{1,n}_{r_7^k}, RK^{2,n}_{r_7^k}$

Fig. 3: Sample Role Key Hierarchy (RKH)

$\left\langle g, g^a, g^b, g^c, Z = \hat{e}\left(g, g\right)^{abc}\right\rangle$ and $\left\langle g, g^a, g^b, g^c, Z = \hat{e}\left(g, g\right)^{z}\right\rangle$ with non-negligible advantage, where $(a, b, c, z) \in \mathbb{Z}_q^*$ are randomly chosen.

5. PROPOSED SCHEME

This section presents the proposed scheme in details. First, a brief overview of the proposed scheme is presented, followed by its main construction.

5.1. Overview

The main goal of the proposed scheme is to enable the owners to enforce RBAC access policies on the encrypted data so that only the users with the authorized roles can perform the keyword search along with efficient data decryption. To achieve this, the proposed scheme devises a novel RBE technique that enables only the users having authorized roles satisfying the specified RBAC access policy to delegate the keyword search capability to the public cloud without disclosing any sensitive information. To reduce decryption cost at the user side, the devised RBE technique also enables the authorized users to delegate computationally expensive cryptographic operations to the public cloud.

In the proposed scheme, each organization is allowed to maintain its own role hierarchy, and each role hierarchy is associated with a Role-Key Hierarchy (RKH). In Figure 3, a sample RKH is shown. Each node in a RKH represents a role, and each role (except the root role), say r_i^k, is associated with a role public key, say $\mathtt{PK}_{\mathtt{r}_{\mathtt{i}}^{\mathtt{k}}}$. In addition, each role (except the root role) is associated with a set of users who have that role, and the users are assigned with a unique pair of role-keys for each role they possess[8]. The role-keys are generated in such a way that the user can use them to compute trap-

[8] In our proposed scheme, the root role, i.e., r_r^k is not assigned to any users, and it is internally managed by the SA. As such, we do not consider any user set with the root role in the Figure 3.

doors to perform a keyword search over the ciphertexts, which are encrypted using a role public key of any descendent role. The same trapdoor can also be used to perform the outsourced decryption operation. This in turn enables the users to gain access to the actual plaintext data. This process is illustrated as follows. Let us assume that the owner wants to authorize all the users having access privileges for the role r_5^k to have access to data. The owner encrypts the data and the associated keywords using the role public key $\mathtt{PK}_{\mathtt{r}_5^\mathtt{k}}$. Any user who possesses any one of the roles in $\mathbb{R}_{r_5^k} = \{r_r^k, r_1^k, r_2^k, r_3^k, r_5^k\}$ can search and decrypt the encrypted data using their respective role-keys. That is, the user possesses a qualified role for accessing the ciphertext. Similarly, if the owner encrypts data and associated keywords using the role public keys $\mathtt{PK}_{\mathtt{r}_1^\mathtt{k}}$ and $\mathtt{PK}_{\mathtt{r}_4^\mathtt{k}}$, then any user who possesses roles in $\mathbb{R}_{r_1^k} = \{r_r^k, r_1^k\}$ and $\mathbb{R}_{r_4^k} = \{r_r^k, r_1^k, r_2^k, r_4^k\}$ respectively can perform keyword search and data decryption using their respective role-keys.

To support multi-organization data sharing, the proposed scheme takes advantage of an existing group key distribution protocol to generate a common master secret for all the participating organizations. This master secret is used for generating the system parameters, including public parameters and master secrets of each organizations. This allows a user to possess more than one role from different organizations. More details are given in the following subsection.

5.2. Construction

A detailed description of all the phases of the proposed scheme is presented as follows.

5.2.1. System Setup. In this phase, the system authority of each organization mutually publishes the system public parameter, and they generate their own master secrets. This phase consists of the SYSTEMSETUP algorithm which is defined next.

SYSTEMSETUP $\left((\mathtt{PP}, \{\mathtt{MS}_\mathtt{k}\}_{\forall \mathtt{k} \in \Phi}) \leftarrow 1^\Lambda\right)$. It chooses two cyclic multiplicative bilinear groups $\mathbb{G}_1$ and $\mathbb{G}_T$ of order q, where q is a large prime number. It also chooses a generator $g \in \mathbb{G}_1$, random numbers $\{\eta_k, \mu_k, \mathrm{x}_\mathrm{k}\}_{\forall k \in \Phi} \in \mathbb{Z}_q^*$ and two hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*, H_2 : \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$. Afterward, all the system authorities follow a group key generation protocol, as described in Section 4.3, to compute a shared secret g^y, where $y = y_1 \cdot y_2 + y_2 \cdot y_3 + ... + y_m \cdot y_1$ and m is the total number of system authorities. Afterward, it computes $Y = \hat{e}(g,g)^y$ and $h_1^k = g^{\eta_k}$, and then publishes the system public parameter $\mathtt{PP} = \left\langle \mathbb{G}_1, \mathbb{G}_T, g, \hat{e}, H_1, H_2, Y, \{h_1^k\}_{\forall k \in \Phi} \right\rangle$. Each system authority, say k^{th} system authority $\mathtt{SA}_\mathtt{k}$, keeps master secret $\mathtt{MS}_\mathtt{k} = \langle g^y, \eta_k, \mu_k, \mathrm{x}_\mathrm{k} \rangle$ in a secure place.

Remark 5.1. All the system authorities can check validity of Y by comparing $\hat{e}(g^y, g) \overset{?}{=} Y$. Also, any number of new system authorities can be added in the system at any time by sharing the existing group secret key, i.e., g^y.

5.2.2. Management of Roles. In this phase, a system authority generates the role related parameters. Suppose the system authority $\mathtt{SA}_\mathtt{k}$ wants to initialize a role hierarchy $\mathcal{H}$. The system authority $\mathtt{SA}_\mathtt{k}$ generates role secrets $\mathtt{RS}_{\mathtt{r}_\mathtt{i}^\mathtt{k}}$ and role public keys $\mathbb{PK}_{r_i^k}$ for each role r_i^k associated with $\mathcal{H}$. It also computes proxy re-encryption keys $\{\mathtt{PKey}_{\mathtt{r}_\mathtt{i}^\mathtt{k}}^{\mathtt{r}_\mathtt{x}^\mathtt{k}}\}_{r_x^k \in \mathbb{R}_{r_i^k} \setminus \{r_i^k\}}$ for each role r_i^k (except the root role) associated with the role hierarchy $\mathcal{H}$. It stores the role public keys in its public bulletin board and keeps the role secrets in a secure place. It also shares each role secret to its corresponding role-manager. That is, the role secret associated with r_i^k, i.e., $\mathtt{RS}_{\mathtt{r}_\mathtt{i}^\mathtt{k}}$ is shared with the role-manager which manages r_i^k, i.e., $\mathtt{RM}_{\mathtt{r}_\mathtt{i}^\mathtt{k}}$. Moreover, the proxy re-encryption keys are sent to the proxy-server (i.e., public cloud) using secure-channels. This phase consists of the MANAGEROLE algorithm which is defined next.

$\textsc{ManageRole}\ \Big(\Big(\mathtt{RP_k}, \{\mathbb{PK}_{r_i^k}\}_{\forall r_i^k \in \Psi_k}, \{\mathtt{RS_{r_i^k}}\}_{\forall r_i^k \in \Psi_k}, \left\{\left\{\mathtt{PKey}_{\mathtt{r_i^k}}^{\mathtt{r_w^k}}\right\}_{\forall r_w^k \in \mathbb{R}_{r_i^k} \setminus \{r_i^k\}}\right\}_{\forall r_i^k \in \Psi_k}\Big) \leftarrow \Big(\mathcal{H}, \mathtt{PP}\Big)\Big)$. It selects random numbers $\{t_{r_i^k}\}_{\forall r_i^k \in \Psi_k} \in \mathbb{Z}_q^*$. It computes role secrets $\mathtt{RS_{r_i^k}}$, role public key $\mathbb{PK}_{r_i^k} = \Big\langle \mathtt{PK_{r_i^k}}, r_i^k, \mathbb{R}_{r_i^k} \Big\rangle$ and proxy re-encryption key $\{\mathtt{PKey}_{\mathtt{r_i^k}}^{\mathtt{r_w^k}}\}_{\forall r_w^k \in \mathbb{R}_{r_i^k} \setminus \{r_i^k\}}$ for each role $r_i^k \in (\Psi_k \setminus \{r_r^k\})$, where

$$\mathtt{RS_{r_i^k}} = \prod_{\forall r_j^k \in \mathbb{R}_{r_i^k}} t_{r_j^k}$$

$$\mathtt{PK_{r_i^k}} = g^{\prod_{\forall r_j^k \in \mathbb{R}_{r_i^k}} t_{r_j^k}}$$

$$\mathtt{PKey}_{\mathtt{r_i^k}}^{\mathtt{r_w^k}} = \prod_{\forall r_j^k \in \mathbb{R}_{r_i^k} \setminus \{r_w^k\}} t_{r_j^k}$$

$\mathbb{R}_{r_i^k}$ is the set of ancestor roles of r_i^k and role secret parameter $\mathtt{RP_k} = \Big\langle \{t_{r_i^k}\}_{\forall r_i^k \in \Psi_k} \Big\rangle$. The system authority sends each secret role parameter and role secret associated with a role to the role-manager which is responsible of its management. For example, secret role parameter $t_{r_i^k}$ and role secret $\mathtt{RS_{r_i^k}}$ are shared with the role-manager $\mathtt{RM_{r_i^k}}$. Note that the root role is internally managed by the system authority. As such, no proxy re-encryption key, role secret key, role public key are generated for the root role.

5.2.3. Public Cloud Key Generation. In this phase, a system authority generates keys for the public cloud. Let the system authority $\mathtt{SA_k}$ wants to issue keys for the public cloud. It computes a private key $\mathtt{Priv_c^k}$, two public keys $(\mathtt{Pub_c^{1k}}, \mathtt{Pub_c^{2k}})$ and sends the private key $\mathtt{Priv_c^k}$ to the public cloud using a secure-channel. It stores both the public keys $(\mathtt{Pub_c^{1k}}, \mathtt{Pub_c^{2k}})$ in its public bulletin board. This phase consists of the PUBCLOUDKEYGEN algorithm which is defined next.

PUBCLOUDKEYGEN $\big((\mathtt{Priv_c^k}, \mathtt{Pub_c^{1k}}, \mathtt{Pub_c^{2k}}) \leftarrow (\mathtt{PP}, \mathtt{MS_k}, \mathtt{ID_c})\big)$. It computes a private key $\mathtt{Priv_c^k}$ and two public keys $(\mathtt{Pub_c^{1k}}, \mathtt{Pub_c^{2k}})$ for the public cloud as follows:

$$\mathtt{Priv_c^k} = H_2\Big(\,(g^y)^{\frac{H_1(\mathtt{ID_c})}{\mathtt{x_k}}}\,\Big) = H_2\Big(g^{\frac{y \cdot H_1(\mathtt{ID_c})}{\mathtt{x_k}}}\Big)$$

$$\mathtt{Pub_c^{1k}} = g^{\mu_k \cdot \mathtt{Priv_c^k}}$$

$$\mathtt{Pub_c^{2k}} = g^{\mathtt{x_k} \cdot \mathtt{Priv_c^k}}$$

5.2.4. New User Enrolment. A system authority initiates this phase when a new legitimate user, say $\mathtt{ID_u}$, wants to join an organization, say k^{th} organization. The system authority $\mathtt{SA_k}$ generates a secret key $\mathtt{SK_{ID_u}^k}$ and public key $\mathtt{Pub_{ID_u}^k}$ for the user $\mathtt{ID_u}$. It also generates a user secret $\mathtt{US_{ID_u}}$ which is shared with all the role-managers under its control. $\mathtt{SA_k}$ sends the secret key $\mathtt{SK_{ID_u}^k}$ to the user $\mathtt{ID_u}$ using a secure-channel and keeps the public key $\mathtt{Pub_{ID_u}^k}$ in its public bulletin board. This phase comprises the USERPRIVKEYGEN algorithm which is defined next.

USERPRIVKEYGEN $\left(\left(\mathtt{SK}^{\mathtt{k}}_{\mathtt{ID_u}}, \mathtt{Pub}^{\mathtt{k}}_{\mathtt{ID_u}}, \mathtt{US}_{\mathtt{ID_u}}\right) \leftarrow \left(\mathtt{MS_k}, \mathtt{PP}, \mathtt{ID_u}\right) \right)$. It issues a pair of secret key $\mathtt{SK}^{\mathtt{k}}_{\mathtt{ID_u}} = \langle \mathtt{Priv}_{\mathtt{ID_u}}, \mathtt{Priv}^{\mathtt{k}}_{\mathtt{ID_u}} \rangle$, public key $\mathtt{Pub}^{\mathtt{k}}_{\mathtt{ID_u}}$, and a user secret $\mathtt{US}_{\mathtt{ID_u}}$ as follows:

$$\begin{aligned}
\mathtt{Priv}_{\mathtt{ID_u}} =& H_2\left((g^y)^{H_1(\mathtt{ID_u})} \right) = H_2\left(g^{y \cdot H_1(\mathtt{ID_u})} \right) \\
\mathtt{Priv}^{\mathtt{k}}_{\mathtt{ID_u}} =& (g^y)^{\frac{\mathtt{Priv}_{\mathtt{ID_u}}}{\eta_k}} \cdot g^{\frac{\mathtt{x_k}}{\eta_k}} = g^{\frac{y \cdot \mathtt{Priv}_{\mathtt{ID_u}} + \mathtt{x_k}}{\eta_k}} \\
\mathtt{Pub}^{\mathtt{k}}_{\mathtt{ID_u}} =& g^{\frac{H_2\left(\mathtt{Priv}^{\mathtt{k}}_{\mathtt{ID_u}}\right)}{\mathtt{Priv}_{\mathtt{ID_u}}}} \\
\mathtt{US}_{\mathtt{ID_u}} =& (g^y)^{\mathtt{Priv}_{\mathtt{ID_u}}} \cdot g^{\mu_k} = g^{y \cdot \mathtt{Priv}_{\mathtt{ID_u}} + \mu_k}
\end{aligned}$$

Note that all the system authorities compute the same private key $\mathtt{Priv}_{\mathtt{ID_u}}$ for the user $\mathtt{ID_u}$. Hence, the user $\mathtt{ID_u}$ needs to keep only one copy of it.

5.2.5. Role Assignment. In this phase, a role-manager assigns roles to a legitimate user. Suppose the role-manager $\mathtt{RM}_{\mathtt{r^k_x}}$ wants to assign a role r^k_x to the user $\mathtt{ID_u}$. To do so, $\mathtt{RM}_{\mathtt{r^k_x}}$ computes two role-keys $(\mathtt{RK}^{1,\mathtt{u}}_{\mathtt{r^k_x}}, \mathtt{RK}^{2,\mathtt{u}}_{\mathtt{r^k_x}})$ for the user $\mathtt{ID_u}$ and sends the role-keys to the user $\mathtt{ID_u}$ using a secure-channel. This phase comprises the USERROLEKEYGEN algorithm which is described next.

USERROLEKEYGEN $((\mathtt{RK}^{1,\mathtt{u}}_{\mathtt{r^k_x}}, \mathtt{RK}^{2,\mathtt{u}}_{\mathtt{r^k_x}}) \leftarrow (\mathtt{PP}, \mathtt{US}_{\mathtt{ID_u}}, \mathtt{RS}_{\mathtt{r^k_x}}, t_{r^k_x}))$. Let's say, user $\mathtt{ID_u}$ is assigned with the role r^k_x. $\mathtt{RM}_{\mathtt{r^k_x}}$ computes the role-keys $\mathtt{RK}^{1,\mathtt{u}}_{\mathtt{r^k_x}}$ and $\mathtt{RK}^{2,\mathtt{u}}_{\mathtt{r^k_x}}$ as follows:

$$\begin{aligned}
\mathtt{RK}^{1,\mathtt{u}}_{\mathtt{r^k_x}} &= (\mathtt{US}_{\mathtt{ID_u}})^{\frac{1}{\mathtt{RS}_{\mathtt{r^k_x}}}} = g^{\frac{y \cdot \mathtt{Priv}_{\mathtt{ID_u}} + \mu_k}{\prod_{r^k_j \in \mathbb{R}_{r^k_x}} t_{r^k_j}}} \\
\mathtt{RK}^{2,\mathtt{u}}_{\mathtt{r^k_x}} &= (\mathtt{US}_{\mathtt{ID_u}})^{\frac{1}{t_{r^k_x}}} = g^{\frac{y \cdot \mathtt{Priv}_{\mathtt{ID_u}} + \mu_k}{t_{r^k_x}}}
\end{aligned}$$

5.2.6. Data Encryption. In this phase, the owner encrypts the plaintext data and then outsources the encrypted data to the cloud storage servers. The owner first encrypts the plaintext data using a random symmetric key by following a secure symmetric key encryption algorithm (e.g., Advanced Encryption Standards). The owner then chooses a set of keywords associated with the actual plaintext data and encrypts the chosen keywords along with the symmetric key using our proposed ENC algorithm. Finally, the owner combines both ciphertexts (i.e., symmetric key and actual plaintext data components) into one archive and outsources the archive file to the public cloud. The ENC algorithm is defined as follows:

ENC $\Big(\mathbb{CT} \leftarrow (\mathtt{PP}, \mathtt{Pub}^{\mathtt{1k}}_{\mathtt{c}}, \mathtt{Pub}^{\mathtt{2k}}_{\mathtt{c}}, \mathtt{M}, \mathbb{W}, \Gamma, \Gamma_\Phi)\Big)$. Let an owner of the k^{th} organization wants to share a plaintext message $\mathtt{M}$ with the users who possess access rights for the roles in Γ. Let w be a keyword from the keyword space $\mathbb{W}$. First, the owner chooses a random number $\mathtt{K} \in \mathbb{G}_T$ and encrypts the plaintext message $\mathtt{M}$ using $\mathtt{K}$ by following a symmetric key encryption algorithm. Afterward, the owner encrypts the random number $\mathtt{K}$ along with the keyword w using the role public parameters of the roles in Γ.

The owner chooses random numbers $(\{d_{r^k_i}, d'_{r^k_i}\}_{\forall r^k_i \in \Gamma}) \in \mathbb{Z}^*_q$, where $d_i = \sum_{r^k_i \in \Gamma} d_{r^k_i}$, $d_j = \sum_{r^k_i \in \Gamma} d'_{r^k_i}$ and $d = d_i + d_j$. The owner also computes $\{d_k = \sum_{\forall r^k_i \in (\Gamma \cap \Psi_k)} d_{r^k_i}\}_{\forall k \in \Gamma_\Phi}$ and $\{d'_k = \sum_{\forall r^k_i \in (\Gamma \cap \Psi_k)} d'_{r^k_i}\}_{\forall k \in \Gamma_\Phi}$. Finally, the owner generates a ciphertext $\mathbb{CT} = \Big\langle \mathtt{Enc_K(M)}, C_1, C_2, C_3, \{C_{4k}, C'_{4k}\}_{\forall k \in \Gamma_\Phi}, \{C_{r^k_i}, C'_{r^k_i}\}_{\forall r^k_i \in \Gamma}, \Gamma, \Gamma_\Phi \Big\rangle$ for the plaintext message

$\mathtt{M}$, where:

$$
\begin{aligned}
C_1 =&\mathtt{K} \cdot Y^d = \mathtt{K} \cdot \hat{e}(g,g)^{y \cdot d} \\
C_2 =&(h_1^k)^{d_j} = g^{\eta_k \cdot d_j} \\
C_3 =&(\mathtt{Pub}_{\mathtt{c}}^{\mathtt{2k}})^{d_j} = g^{\mathtt{x_k} \cdot \mathtt{Priv}_{\mathtt{c}}^{\mathtt{k}} \cdot d_j} \\
C_{4k} =& \left(\mathtt{Pub}_{\mathtt{c}}^{\mathtt{1k}}\right)^{d_k} = g^{\mu_k \cdot \mathtt{Priv}_{\mathtt{c}}^{\mathtt{k}} \cdot d_k} \\
C'_{4k} =& \left(\mathtt{Pub}_{\mathtt{c}}^{\mathtt{1k}}\right)^{d'_k} = g^{\mu_k \cdot \mathtt{Priv}_{\mathtt{c}}^{\mathtt{k}} \cdot d'_k} \\
C_{r_i^k} =& \left(\mathtt{PK}_{\mathtt{r_i^k}}\right)^{d_{r_i^k} \cdot H_1(w)} = g^{H_1(w) \cdot d_{r_i^k} \prod_{r_j^k \in \mathbb{R}_{r_i^k}} t_{r_j^k}} \\
C'_{r_i^k} =& \left(\mathtt{PK}_{\mathtt{r_i^k}}\right)^{d'_{r_i^k} \cdot H_1(w)} = g^{H_1(w) \cdot d'_{r_i^k} \prod_{r_j^k \in \mathbb{R}_{r_i^k}} t_{r_j^k}}
\end{aligned}
$$

Note that the data owner embeds the hashed value of the keyword, i.e, $H_1(w)$ for some roles in Γ only (this fixed position can be seen as part of the public parameter).

5.2.7. Trapdoor Generation. In this phase, a user generates trapdoor $\mathtt{Trap}$ using his/her secret keys and the keywords of his/her choice for delegating keyword search capabilities to the public cloud. The user sends the trapdoor $\mathtt{Trap}$ along with the associated roles to the public cloud using a secure-channel. This phase comprises the TRAPGEN algorithm which is described next.

TRAPGEN $((\mathtt{Trap}, v) \leftarrow (\{\mathtt{RK}_{\mathtt{r_x^k}}^{\mathtt{1,u}}, \mathtt{RK}_{\mathtt{r_x^k}}^{\mathtt{2,u}}\}_{\forall r_x^k \in \mathbb{S}_{\mathtt{ID_u}}}, \mathtt{SK}_{\mathtt{ID_u}}^{\mathtt{k}}, \mathbb{S}_{\mathtt{ID_u}}, w))$. Suppose the user $\mathtt{ID_u}$ who possesses roles $\mathbb{S}_{\mathtt{ID_u}}$ wants to access the ciphertexts associated with the keyword w of the k^{th} organization. User $\mathtt{ID_u}$ chooses a random secret $v \in \mathbb{Z}_q^*$, current timestamp ts, and then he computes a trapdoor $\mathtt{Trap} = \left\langle tr_1, tr_2, tr_3, tr_4, \{tr_{r_x^k}^1, tr_{r_x^k}^2\}_{\forall r_x^k \in \mathbb{S}_{\mathtt{ID_u}}}, \mathbb{S}_{\mathtt{ID_u}}, ts \right\rangle$, where:

$$
\begin{aligned}
tr_1 =& \left[\frac{\mathtt{Priv}_{\mathtt{ID_u}} + ts}{H_2\left(\mathtt{Priv}_{\mathtt{ID_u}}^{\mathtt{k}}\right)}\right] v = \frac{\left[\mathtt{Priv}_{\mathtt{ID_u}} + ts\right] v}{H_2\left(\mathtt{Priv}_{\mathtt{ID_u}}^{\mathtt{k}}\right)} \\
tr_2 =& \left(\mathtt{Priv}_{\mathtt{ID_u}}^{\mathtt{k}}\right)^v = g^{\frac{\left[y \cdot \mathtt{Priv}_{\mathtt{ID_u}} + \mathtt{x_k}\right] \cdot v}{\eta_k}} \\
tr_3 =&g^{\frac{v}{\mathtt{Priv}_{\mathtt{ID_u}}}} \\
tr_4 =&g^v \\
tr_{r_x^k}^1 =& \left(\mathtt{RK}_{\mathtt{r_x^k}}^{\mathtt{1,u}}\right)^{\frac{v}{H_1(w)}} = g^{\frac{\left[y \cdot \mathtt{Priv}_{\mathtt{ID_u}} + \mu_k\right] v}{H_1(w) \cdot \prod_{r_j^k \in \mathbb{R}_{r_x^k}} t_{r_j^k}}} \\
tr_{r_x^k}^2 =& \left(\mathtt{RK}_{\mathtt{r_x^k}}^{\mathtt{2,u}}\right)^{\frac{v}{H_1(w)}} = g^{\frac{\left[y \cdot \mathtt{Priv}_{\mathtt{ID_u}} + \mu_k\right] v}{H_1(w) \cdot t_{r_x^k}}}
\end{aligned}
$$

The user $\mathtt{ID_u}$ keeps the random secret v in a secure place for decryption of the ciphertexts in Section 5.2.9.

5.2.8. Data Search. In this phase, the public cloud performs a keyword search operation on the ciphertexts using the trapdoor received from the requested user $\mathtt{ID_u}$. This phase consists of the AUTHENTICATION, KEYSEARCH and PARTIALDEC algorithms. In the AUTHENTICATION algorithm, the public cloud authenticates the user and checks freshness of the keyword search request. In the KEYSEARCH algorithm, the public cloud performs all the search related operation for finding the ciphertexts which have a matching keyword with the trapdoor received from the user $\mathtt{ID_u}$. This will be done if and only if the user is legitimate and the keyword search request is valid. In the PARTIALDEC algorithm, the public cloud partially decrypts the ciphertexts and finally sends the partially decrypted ciphertexts to the user $\mathtt{ID_u}$. The details of these algorithms are given next.

AUTHENTICATION $\left(\left(V_3^1 / \perp\right) \leftarrow \left(\{\mathtt{Priv}_\mathtt{c}^\mathtt{k}\}_{\forall \mathtt{k} \in \Gamma_\Phi}, \mathtt{Trap}, \mathtt{Pub}^\mathtt{k}_{\mathtt{ID_u}}, \mathtt{ID_u}, ts' \right) \right)$. Before performing computationally expensive operations, the public cloud first authenticates the requesting user. During the authentication process, the public cloud also checks the freshness of search request by comparing the timestamp ts associated with the trapdoor to its own current timestamp ts' for preventing replay attacks. If the authentication fails or if the timestamp associated with the trapdoor represents a past time, the public cloud aborts the connection, i.e., returns $\perp$. Otherwise, it performs keyword search operations defined in the KEYSEARCH algorithm. To authenticate the user and check the freshness of the request, the public cloud computes U', V_1^1, V_1^2 and V_1^3, based on its known $\{\mathtt{Priv}_\mathtt{c}^\mathtt{k}\}_{\forall k \in \Gamma_\Phi})$ keys, where:

$$
\begin{aligned}
U' &= \prod_{\forall k \in \Gamma_\Phi} \left(C'_{4k}\right)^{\frac{1}{\mathtt{Priv}_\mathtt{c}^\mathtt{k}}} \\
&= \prod_{\forall k \in \Gamma_\Phi} \left(g^{\mu_k \cdot \mathtt{Priv}_\mathtt{c}^\mathtt{k} \cdot d'_k}\right)^{\frac{1}{\mathtt{Priv}_\mathtt{c}^\mathtt{k}}} \\
&= g^{\sum_{\forall k \in \Gamma_\Phi} \mu_k \cdot d'_k} \\
V_1^1 &= \hat{e}\left(\left(\mathtt{Pub}^\mathtt{k}_{\mathtt{ID_u}}\right)^{tr_1}, U'\right) \\
&= \hat{e}\left(\left(g^{\frac{H_2\left(\mathtt{Priv}^\mathtt{k}_{\mathtt{ID_u}}\right)}{\mathtt{Priv}_{\mathtt{ID_u}}}}\right)^{\frac{\left[\mathtt{Priv}_{\mathtt{ID_u}} + ts\right] v}{H_2\left(\mathtt{Priv}^\mathtt{k}_{\mathtt{ID_u}}\right)}}, g^{\sum_{\forall k \in \Gamma_\Phi} \mu_k \cdot d'_k}\right) \\
&= \hat{e}\left(g^v, g^{\sum_{\forall k \in \Gamma_\Phi} \mu_k \cdot d'_k}\right) \cdot \hat{e}\left(g^{\frac{v \cdot ts}{\mathtt{Priv}_{\mathtt{ID_u}}}}, g^{\sum_{\forall k \in \Gamma_\Phi} \mu_k \cdot d'_k}\right) \\
&= \hat{e}\left(g, g\right)^{v \sum_{\forall k \in \Gamma_\Phi} \mu_k \cdot d'_k} \cdot \hat{e}\left(g, g\right)^{\frac{v \cdot ts \sum_{\forall k \in \Gamma_\Phi} \mu_k \cdot d'_k}{\mathtt{Priv}_{\mathtt{ID_u}}}} \\
V_1^2 &= \hat{e}\left((tr_3)^{ts}, U'\right) \\
&= \hat{e}\left(g^{\frac{v \cdot ts}{\mathtt{Priv}_{\mathtt{ID_u}}}}, g^{\sum_{\forall k \in \Gamma_\Phi} \mu_k \cdot d'_k}\right) \\
&= \hat{e}\left(g, g\right)^{\frac{v \cdot ts \sum_{\forall k \in \Gamma_\Phi} \mu_k \cdot d'_k}{\mathtt{Priv}_{\mathtt{ID_u}}}} \\
V_1^3 &= \hat{e}\left(tr_4, U'\right) \\
&= \hat{e}\left(g^v, g^{\sum_{\forall k \in \Gamma_\Phi} \mu_k \cdot d'_k}\right) \\
&= \hat{e}\left(g, g\right)^{v \sum_{\forall k \in \Gamma_\Phi} \mu_k \cdot d'_k}
\end{aligned}
$$

Now, the public cloud checks whether $V_1^1 \overset{?}{=} V_1^2 \cdot V_1^3$. If the equation holds, the public cloud performs the operations defined in the KEYSEARCH algorithm. Otherwise, it aborts the connection.

Proof of consistency:

$$
V_1^1 = \hat{e}\left(g, g\right)^{v \sum_{\forall k \in \Gamma_\Phi} \mu_k \cdot d'_k} \cdot \hat{e}\left(g, g\right)^{\frac{v \cdot ts \sum_{\forall k \in \Gamma_\Phi} \mu_k \cdot d'_k}{\mathtt{Priv}_{\mathtt{ID_u}}}} = V_1^3 \cdot V_1^2
$$

KEYSEARCH $((\mathbb{CT}/\ \perp) \leftarrow (\mathbb{CT}, \mathtt{Trap}, V_1^3))$. Suppose the user $\mathtt{ID_u}$ possesses a role set $\mathbb{S}_{\mathtt{ID_u}}$ and wants to access the k^{th} organization's data. Suppose $\mathbb{CT} = \big\langle \mathtt{Enc_K(M)}, C_1, C_2, C_3, \{C_{4k}, C'_{4k}\}_{\forall k \in \Gamma_\Phi}, \{C_{r_i^k}, C'_{r_i^k}\}_{\forall r_i^k \in \Gamma}, \Gamma,$ $\Gamma_\Phi \big\rangle$ is the ciphertext of the k^{th} organization on which the public cloud wants to perform the keyword search operation, where for all $r_i^k \in \Gamma$, there is at least one $r_x^k \in \mathbb{S}_{\mathtt{ID_u}}$ such that $r_x^k \in \mathbb{R}_{r_i^k}$.

The public cloud computes $V^1_{r_x^k}$ and V_2. While computing $V^1_{r_x^k}$, two cases are considered which are as follows:

Case 1: if $r_x^k == r_i^k$, then

$$\begin{aligned} V^1_{r_x^k} =& \hat{e}\left(tr^1_{r_x^k}, C'_{r_i^k}\right) \\ =& \hat{e}\left(g^{\frac{[y \cdot \mathtt{Priv_{ID_u}} + \mu_k]v}{H_1(w) \cdot \prod_{r_j^k \in \mathbb{R}_{r_x^k}} t_{r_j^k}}}, g^{H_1(w) \cdot d'_{r_i^k} \prod_{r_j^k \in \mathbb{R}_{r_i^k}} t_{r_j^k}}\right) \\ =& \hat{e}\left(g, g\right)^{[y \cdot \mathtt{Priv_{ID_u}} + \mu_k] \cdot v \cdot d'_{r_i^k}}, \left(\text{as } \mathbb{R}_{r_x^k} = \mathbb{R}_{r_i^k}\right) \\ =& \hat{e}\left(g, g\right)^{[y \cdot \mathtt{Priv_{ID_u}}]v \cdot d'_{r_i^k}} \cdot \hat{e}\left(g, g\right)^{\mu_k \cdot v \cdot d'_{r_i^k}} \end{aligned}$$

Otherwise, Case 2: if $r_x^k \in \left(\mathbb{R}_{r_i^k} \setminus \{r_i^k\}\right)$ (let $\gamma = [y \cdot \mathtt{Priv_{ID_u}} + \mu_k]$)

$$\begin{aligned} V^1_{r_x^k} =& \hat{e}\left(\left(tr^2_{r_x^k}\right)^{\mathtt{PKey}^{\mathtt{r_x^k}}_{\mathtt{r_i^k}}}, C'_{r_i^k}\right) \\ =& \hat{e}\left(g^{\frac{\gamma \cdot v}{H_1(w) \cdot t_{r_x^k}} \cdot \frac{1}{\prod_{\forall r_j^k \in \mathbb{R}_{r_i^k} \setminus \{r_x^k\}} t_{r_j^k}}}, g^{H_1(w) \cdot d'_{r_i^k} \prod_{r_j^k \in \mathbb{R}_{r_i^k}} t_{r_j^k}}\right) \\ =& \hat{e}\left(g^{\frac{\gamma \cdot v}{\prod_{\forall r_j^k \in \mathbb{R}_{r_i^k}} t_{r_j^k}}}, g^{d'_{r_i^k} \prod_{r_j^k \in \mathbb{R}_{r_i^k}} t_{r_j^k}}\right) \\ =& \hat{e}\left(g, g\right)^{\gamma \cdot v \cdot d'_{r_i^k}} \\ =& \hat{e}\left(g, g\right)^{[y \cdot \mathtt{Priv_{ID_u}} + \mu_k]v \cdot d'_{r_i^k}} \\ =& \hat{e}\left(g, g\right)^{[y \cdot \mathtt{Priv_{ID_u}}]v \cdot d'_{r_i^k}} \cdot \hat{e}\left(g, g\right)^{\mu_k \cdot v \cdot d'_{r_i^k}} \\ V_2 =& \prod V^1_{r_x^k} \\ =& \hat{e}\left(g, g\right)^{[y \cdot \mathtt{Priv_{ID_u}}]v \cdot \sum d'_{r_i^k}} \cdot \hat{e}\left(g, g\right)^{v \sum \mu_k \cdot d'_{r_i^k}} \\ =& \hat{e}\left(g, g\right)^{[y \cdot \mathtt{Priv_{ID_u}}]v \cdot d_j} \cdot \hat{e}\left(g, g\right)^{v \sum \mu_k \cdot d'_{r_i^k}} \end{aligned}$$

Now, the public cloud computes V_3, where

$$V_3 = \frac{V_2}{V_1^3} = \frac{\hat{e}\left(g, g\right)^{[y \cdot \mathtt{Priv_{ID_u}}]v \cdot d_j} \cdot \hat{e}\left(g, g\right)^{v \sum \mu_k \cdot d'_{r_i^k}}}{\hat{e}\left(g, g\right)^{v \sum \mu_k \cdot d'_k}} = \hat{e}\left(g, g\right)^{y \cdot \mathtt{Priv_{ID_u}} \cdot d_j \cdot v}$$

Note that $\hat{e}\left(g, g\right)^{v \sum \mu_k \cdot d_{r_i^k}} = \hat{e}\left(g, g\right)^{v \sum \mu_k \cdot d'_k}$ (Please refer Section 5.2.6).

Afterward, the public cloud computes V_4, V_5 and V_6, where

$$
\begin{aligned}
V_4 =&\hat{e}\left(tr_2, C_2\right)\\
=&\hat{e}\left(g^{\frac{[y\cdot\mathtt{Priv_{ID_u}}+\mathtt{x_k}]\cdot v}{\eta_k}}, g^{\eta_k\cdot d_j}\right)\\
=&\hat{e}\left(g,g\right)^{[y\cdot\mathtt{Priv_{ID_u}}+\mathtt{x_k}]\cdot v\cdot d_j}\\
=&\hat{e}\left(g,g\right)^{y\cdot\mathtt{Priv_{ID_u}}\cdot v\cdot d_j}\cdot\hat{e}\left(g,g\right)^{\mathtt{x_k}\cdot v\cdot d_j}\\
V_5 =&\hat{e}\left(\left(tr_4\right)^{\frac{1}{\mathtt{Priv_c^k}}}, C_3\right)=\hat{e}\left(g^{\frac{v}{\mathtt{Priv_c^k}}}, g^{\mathtt{x_k}\cdot\mathtt{Priv_c^k}\cdot d_j}\right)=\hat{e}\left(g,g\right)^{\mathtt{x_k}\cdot v\cdot d_j}\\
V_6 =&\frac{V_1}{V_2}=\frac{\hat{e}\left(g,g\right)^{y\cdot\mathtt{Priv_{ID_u}}\cdot v\cdot d_j}\cdot\hat{e}\left(g,g\right)^{\mathtt{x_k}\cdot v\cdot d_j}}{\hat{e}\left(g,g\right)^{\mathtt{x_k}\cdot v\cdot d_j}}=\hat{e}\left(g,g\right)^{y\cdot\mathtt{Priv_{ID_u}}\cdot d_j\cdot v}
\end{aligned}
$$

Finally, the public cloud compares V_3 and V_6. If both are equal then it performs the operations defined in the PARTIALDEC algorithm (described in Section 5.2.8). Otherwise, it aborts all the operations and outputs $\perp$, which means that the ciphertext does not have the desired keyword.

PARTIALDEC $\left(\mathbb{CT}'\leftarrow(\mathbb{CT},\mathtt{Trap},\{\mathtt{Priv_c^k}\}_{\forall k\in\Gamma_\Phi},\mathbb{S}_{\mathtt{ID_u}})\right)$. In this algorithm, the public cloud partially decrypts all the ciphertexts returned by the KEYSEARCH algorithm. Suppose ciphertext $\mathbb{CT}=\left\langle\mathtt{Enc_K(M)},C_1,C_2,C_3,\{C_{4k},C'_{4k}\}_{\forall k\in\Gamma_\Phi},\{C_{r_i^k},C'_{r_i^k}\}_{\forall r_i^k\in\Gamma},\Gamma,\Gamma_\Phi\right\rangle$ has a matching keyword with the trapdoor Trap. To partially decrypt the ciphertext $\mathbb{CT}$, the public cloud first computes $V^7_{r_x^k}$ and V_7. Similar to $V^1_{r_x^k}$, the computation procedure considers the two following cases to compute $V^7_{r_x^k}$:

Case 1: if $r_x^k == r_i^k$, then

$$
\begin{aligned}
V^7_{r_x^k} =&\hat{e}\left(tr^1_{r_x^k}, C_{r_i^k}\right)\\
=&\hat{e}\left(g^{\frac{[y\cdot\mathtt{Priv_{ID_u}}+\mu_k]v}{H_1(w)\cdot\prod_{r_j^k\in\mathbb{R}_{r_x^k}}t_{r_j^k}}}, g^{H_1(w)\cdot d_{r_i^k}\prod_{r_j^k\in\mathbb{R}_{r_i^k}}t_{r_j^k}}\right)\\
=&\hat{e}\left(g,g\right)^{[y\cdot\mathtt{Priv_{ID_u}}+\mu_k]\cdot v\cdot d_{r_i^k}}, \left(\text{as } \mathbb{R}_{r_x^k}=\mathbb{R}_{r_i^k}\right)\\
=&\hat{e}\left(g,g\right)^{[y\cdot\mathtt{Priv_{ID_u}}]v\cdot d_{r_i^k}}\cdot\hat{e}\left(g,g\right)^{\mu_k\cdot v\cdot d_{r_i^k}}
\end{aligned}
$$

Otherwise, Case 2: if $r_x^k \in \left(\mathbb{R}_{r_i^k} \setminus \{r_i^k\}\right)$ (let $\gamma = [y \cdot \mathtt{Priv}_{\mathtt{ID_u}} + \mu_k]$)

$$\begin{aligned}
V_{r_x^k}^7 =& \hat{e}\left(\left(tr_{r_x^k}^2\right)^{\mathtt{PKey}_{\mathtt{r_i^k}}^{\mathtt{r_x^k}}}, C_{r_i^k}\right)\\
=& \hat{e}\left(g^{\frac{\gamma \cdot v}{H_1(w) \cdot t_{r_x^k}} \cdot \frac{1}{\prod_{\forall r_j^k \in \mathbb{R}_{r_i^k} \setminus \{r_x^k\}} t_{r_j^k}}}, g^{H_1(w) \cdot d_{r_i^k} \prod_{r_j^k \in \mathbb{R}_{r_i^k}} t_{r_j^k}}\right)\\
=& \hat{e}\left(g^{\frac{\gamma \cdot v}{\prod_{\forall r_j^k \in \mathbb{R}_{r_i^k}} t_{r_j^k}}}, g^{d_{r_i^k} \prod_{r_j^k \in \mathbb{R}_{r_i^k}} t_{r_j^k}}\right)\\
=& \hat{e}\left(g, g\right)^{\gamma \cdot v \cdot d_{r_i^k}}\\
=& \hat{e}\left(g, g\right)^{[y \cdot \mathtt{Priv}_{\mathtt{ID_u}} + \mu_k] v \cdot d_{r_i^k}}\\
=& \hat{e}\left(g, g\right)^{[y \cdot \mathtt{Priv}_{\mathtt{ID_u}}] v \cdot d_{r_i^k}} \cdot \hat{e}\left(g, g\right)^{\mu_k \cdot v \cdot d_{r_i^k}}
\end{aligned}$$

$$\begin{aligned}
V_7 = \prod V_{r_x^k}^7 =& \hat{e}\left(g, g\right)^{[y \cdot \mathtt{Priv}_{\mathtt{ID_u}}] v \cdot \sum d_{r_i^k}} \cdot \hat{e}\left(g, g\right)^{v \sum \mu_k \cdot d_{r_i^k}}\\
=& \hat{e}\left(g, g\right)^{[y \cdot \mathtt{Priv}_{\mathtt{ID_u}}] v \cdot d_i} \cdot \hat{e}\left(g, g\right)^{v \sum \mu_k \cdot d_{r_i^k}}
\end{aligned}$$

The public cloud knowing its private key $\{\mathtt{Priv}_{\mathtt{c}}^{\mathtt{k}}\}_{\forall k \in \Gamma_\Phi}$ computes U and V_8, as follows:

$$\begin{aligned}
U = \prod_{\forall k \in \Gamma_\Phi} \left(C_{4k}\right)^{\frac{1}{\mathtt{Priv}_{\mathtt{c}}^{\mathtt{k}}}} = \prod_{\forall k \in \Gamma_\Phi} \left(g^{\mu_k \cdot \mathtt{Priv}_{\mathtt{c}}^{\mathtt{k}} \cdot d_k}\right)^{\frac{1}{\mathtt{Priv}_{\mathtt{c}}^{\mathtt{k}}}} = g^{\sum_{\forall k \in \Gamma_\Phi} \mu_k \cdot d_k}\\
V_8 =& \hat{e}\left(tr_4, U\right) = \hat{e}\left(g^v, g^{\sum_{\forall k \in \Gamma_\Phi} \mu_k \cdot d_k}\right) = \hat{e}\left(g, g\right)^{v \sum_{\forall k \in \Gamma_\Phi} \mu_k \cdot d_k}
\end{aligned}$$

Now, the public cloud computes V_9, where:

$$V_9 = \frac{V_7}{V_8} = \frac{\hat{e}\left(g, g\right)^{[y \cdot \mathtt{Priv}_{\mathtt{ID_u}}] v \cdot d_i} \cdot \hat{e}\left(g, g\right)^{v \sum \mu_k \cdot d_{r_i^k}}}{\hat{e}\left(g, g\right)^{v \sum \mu_k \cdot d_k}} = \hat{e}\left(g, g\right)^{y \cdot \mathtt{Priv}_{\mathtt{ID_u}} \cdot d_i \cdot v}$$

Note that $\hat{e}\left(g, g\right)^{v \sum \mu_k \cdot d_{r_i^k}} = \hat{e}\left(g, g\right)^{v \sum \mu_k \cdot d_k}$ (Please refer Section 5.2.6).

The public cloud computes V_9, where:

$$\begin{aligned}
V_{10} =& V_6 \cdot V_9\\
=& \hat{e}\left(g, g\right)^{y \cdot \mathtt{Priv}_{\mathtt{ID_u}} \cdot d_j \cdot v} \cdot \hat{e}\left(g, g\right)^{y \cdot \mathtt{Priv}_{\mathtt{ID_u}} \cdot d_i \cdot v}\\
=& \hat{e}\left(g, g\right)^{y \cdot \mathtt{Priv}_{\mathtt{ID_u}} \cdot v [d_j + d_i]}\\
=& \hat{e}\left(g, g\right)^{y \cdot \mathtt{Priv}_{\mathtt{ID_u}} \cdot v \cdot d}
\end{aligned}$$

Finally, the public cloud sends the partially decrypted ciphertext $\mathbb{CT}' = \langle \mathtt{Enc_K(M)}, C_1, V_{10} \rangle$ to the user $\mathtt{ID_u}$.

5.2.9. Decryption. In this phase, the user $\mathtt{ID_u}$ decrypts the received partially decrypted ciphertext $\mathbb{CT}'$ using his/her private key $\mathtt{Priv}_{\mathtt{ID_u}}$ and random secret v. This phase comprises the FULLDEC algorithm which is described next.

FULLDEC($\mathtt{M} \leftarrow (\mathbb{CT}', \mathtt{Priv}_{\mathtt{ID_u}}, v)$). It computes $\mathtt{K}$ from the ciphertext $\mathbb{CT}'$ using his/her secret keys, $\mathtt{priv}_{\mathtt{ID_u}}$ and v.

$$\mathtt{K} = \frac{C_1}{(V_{10})^{\frac{1}{\mathtt{Priv}_{\mathtt{ID_u}} \cdot v}}} = \frac{\mathtt{K} \cdot \hat{e}\,(g,g)^{y \cdot d}}{\left(\hat{e}\,(g,g)^{y \cdot \mathtt{Priv}_{\mathtt{ID_u}} \cdot v \cdot d}\right)^{\frac{1}{\mathtt{Priv}_{\mathtt{ID_u}} \cdot v}}} = \frac{\mathtt{K} \cdot \hat{e}\,(g,g)^{y \cdot d}}{\hat{e}\,(g,g)^{y \cdot d}}$$

Finally, user $\mathtt{ID_u}$ gets the actual plaintext data by decrypting $\mathtt{Enc_K(M)}$ using $\mathtt{K}$ and removes the random secret v from his/her database.

5.3. Conjunctive Keyword Search

Many times a user wants to perform multiple keyword search using a single search request instead of sending multiple single keyword search requests. This property is called the *Conjunctive Keyword Search*. The proposed scheme can provide conjunctive keyword search with the following modifications. The owner computes modified ciphertext components $C_{r_i^k} = \left(\mathtt{PK}_{\mathtt{r_i^k}}\right)^{d_{r_i^k} \cdot \prod H_1(w_i)} = g^{d_{r_i^k} \prod H_1(w_i) \cdot \prod_{r_j^k \in \mathbb{R}_{r_i^k}} t_{r_j^k}}$ and $C'_{r_i^k} = \left(\mathtt{PK}_{\mathtt{r_i^k}}\right)^{d'_{r_i^k} \cdot \prod H_1(w_i)} = g^{d'_{r_i^k} \prod H_1(w_i) \cdot \prod_{r_j^k \in \mathbb{R}_{r_i^k}} t_{r_j^k}}$. Similarly, a user computes trapdoor components $tr^1_{r_x^k} = \left(\mathtt{RK}^{1,\mathtt{u}}_{\mathtt{r_x^k}}\right)^{\frac{v}{\prod H_1(w_i)}}$ and $tr^2_{r_x^k} = \left(\mathtt{RK}^{2,\mathtt{u}}_{\mathtt{r_x^k}}\right)^{\frac{v}{\prod H_1(w_i)}}$. It can be observed that, our conjunctive keyword search mechanism does not introduce any additional overhead in the system.

5.4. Revocation

In the proposed scheme, a SA can revoke a user in two ways, namely *complete user revocation* and *role-level revocation*. The former revocation method means that the user can no longer access any data belonging to that organization. The later revocation method represents that if one or more roles of a user is revoked, the user can still access data with his/her non-revoked roles if they are qualified enough according to the RBAC access policy.

The complete user revocation is achieved by revoking the public key $\mathtt{Pub}^{\mathtt{k}}_{\mathtt{ID_u}}$ of the user, so that the public cloud do not use it during the authentication process in the AUTHENTICATION algorithm defined in Section 5.2.8. To do that, SA removes the pubic key $\mathtt{Pub}^{\mathtt{k}}_{\mathtt{ID_u}}$ of the revoked user $\mathtt{ID_u}$ from its public bulletin board, which can be done easily.

For the role-level revocation, the SA updates all the parameters related with the revoked role. Suppose the SA wants to revoke a role r_i^k from one or more users. To do that, the SA first chooses a fresh random number $t'_{r_i^k} \in \mathbb{Z}_q^*$ and updates all the parameters related with the revoked role r_i^k. The SA computes updated public keys $\left(\mathtt{PK}_{\mathtt{r_j^k}}\right)^{\frac{t'_{r_i^k}}{t_{r_i^k}}}$, role secrets $\left(\mathtt{RS}_{\mathtt{r_j^k}} \cdot \frac{t'_{r_i^k}}{t_{r_i^k}}\right)$ and proxy re-encryption keys $\left(\mathtt{PKey}^{\mathtt{r_w^k}}_{\mathtt{r_j^k}} \cdot \frac{t'_{r_i^k}}{t_{r_i^k}}\right)$ related with the revoked role r_i^k (i.e., for all r_j^k such that $r_i^k \in \mathbb{R}_{r_j^k}$), where $t_{r_i^k}$ is the previously chosen random number associated with r_i^k. The SA then sends the $\frac{t'_{r_i^k}}{t_{r_i^k}}$ to the public cloud for re-encryption of the stored ciphertexts associated with the revoked role r_i^k. It also sends $\frac{t'_{r_i^k}}{t_{r_i^k}}$ to the corresponding role-managers for updating the role-keys associated with the revoked role r_i^k.

The public cloud re-encrypts the ciphertext components $\left(C_{r_j^k}\right)^{\frac{t'_{r_i^k}}{t_{r_i^k}}}$ and $\left(C'_{r_j^k}\right)^{\frac{t'_{r_i^k}}{t_{r_i^k}}}$ for all r_j^k such that $r_i^k \in \mathbb{R}_{r_j^k}$. This is essential to prevent the revoked users from accessing the data using the revoked role (i.e., *Backward Secrecy*).

Moreover, to enable the other non-revoked users for accessing the re-encrypted ciphertexts, the concerned role-managers need to send updated role-keys to the non-revoked users (i.e., *Forward Secrecy*). The updated role-keys are computed as follows: i) $\left(\mathtt{RK}^{\mathtt{1,u}}_{\mathtt{r_i^k}}\right)^{\frac{t_{r_i^k}}{t'_{r_i^k}}}$ for all the non-revoked users who possess r_i^k and ii) $\left(\mathtt{RK}^{\mathtt{2,u}}_{\mathtt{r_j^k}}\right)^{\frac{t_{r_i^k}}{t'_{r_i^k}}}$ for all the non-revoked users who possess r_j^k, such that $r_i^k \in \mathbb{R}_{r_j^k}$.

6. ANALYSIS

This section first presents security analysis of the proposed scheme, followed by its performance analysis. In the security analysis, we demonstrate that the proposed scheme is secure against chosen plaintext and chosen keyword attacks. In the performance analysis, we present a comprehensive performance analysis of the proposed scheme along with its experimental results.

6.1. Security Analysis

6.1.1. Security against Chosen Plaintext Attack. CPA security of the proposed scheme can be defined by the following theorem and proof.

THEOREM 6.1. *If a probabilistic-polynomial time (PPT) adversary $\mathcal{A}_1$ wins the CPA security game as defined in Section 3.5.1 with a non-negligible advantage ϵ, then a PPT simulator $\mathcal{B}$ can be constructed to break the DBDH assumption with non-negligible advantage $\frac{\epsilon}{2}$.*

PROOF. In this proof, we show that a simulator $\mathcal{B}$ can be constructed to help an adversary $\mathcal{A}_1$ to gain advantage $\frac{\epsilon}{2}$ against our proposed scheme.

The DBDH challenger $\mathcal{C}$ chooses random numbers $(a, b, c, z) \in \mathbb{Z}_q^*$ and flips a binary random coin l. It sets $Z = \hat{e}(g, g)^{abc}$ if $l = 0$ and $Z = \hat{e}(g, g)^z$ otherwise. Afterwards, challenger $\mathcal{C}$ sends $A = g^a, B = g^b, C = g^c$ and Z to the simulator $\mathcal{B}$, and it asks the simulator $\mathcal{B}$ to output l. Now simulator $\mathcal{B}$ acts as a challenger in the rest of the security game.

In the following game, simulator $\mathcal{B}$ interacts with the adversary $\mathcal{A}_1$ as follows:

INIT Adversary $\mathcal{A}_1$ sends a challenged role set Γ^*, a keyword w and two identities $(\mathtt{ID}^*_{\mathtt{u}}, \mathtt{ID}^*_{\mathtt{c}})$ to the simulator $\mathcal{B}$.

SETUP Simulator $\mathcal{B}$ chooses random numbers $\{\zeta_k, \vartheta_k, \varrho_k\}_{\forall k \in \Phi} \in \mathbb{Z}_q^*$. It also chooses random numbers $\{\alpha_{r_i^k}\}_{\forall i \in \Psi_k, \forall k \in \Phi} \in \mathbb{Z}_q^*$. Simulator $\mathcal{B}$ computes $Y = \hat{e}(g, g)^{ab} = \hat{e}(A, B), \{h_1^k = g^{b \cdot \zeta_k} = B^{\zeta_k}\}_{\forall k \in \Phi}$. Simulator $\mathcal{B}$ also computes $\mathtt{PK}_{\mathtt{r_i^k}} = g^{b \prod_{\forall r_j^k \in \mathbb{R}_{r_i^k}} \alpha_{r_j^k}} = B^{\prod_{\forall r_j^k \in \mathbb{R}_{r_i^k}} \alpha_{r_j^k}}$ for all $r_i^k \in \Psi_k$, where $1 \leq k \leq m$. Moreover, simulator $\mathcal{B}$ computes $\left\{\left\{\mathtt{PKey}^{\mathtt{r_w^k}}_{\mathtt{r_i^k}} = \prod_{\forall r_j^k \in \mathbb{R}_{r_i^k} \setminus \{r_w^k\}} \alpha_{r_j^k}\right\}_{\forall r_w^k \in \mathbb{R}_{r_i^k} \setminus \{r_i^k\}}\right\}_{\forall r_i^k \in \Psi_k}$ where $1 \leq k \leq m$.

Simulator $\mathcal{B}$ also chooses a random number $\mathtt{s}_{\mathtt{ID}_\mathtt{u}^*} \in \mathbb{Z}_q^*$ and computes $h_{id_c^*} = H_1(\mathtt{ID}_\mathtt{c}^*)$. It then computes $\{\mathtt{Priv}_\mathtt{c}^\mathtt{k}, \mathtt{Pub}_\mathtt{c}^\mathtt{1k}, \mathtt{Pub}_\mathtt{c}^\mathtt{2k}, \mathtt{Priv}_{\mathtt{ID}_\mathtt{u}}^\mathtt{k}, \mathtt{Pub}_{\mathtt{ID}_\mathtt{u}}^\mathtt{k}\}_{\forall k \in \Phi}$ and $\mathtt{Priv}_{\mathtt{ID}_\mathtt{u}}$, where

$$\begin{aligned}
\mathtt{Priv}_\mathtt{c}^\mathtt{k} =& H_2\big(g^{\frac{a \cdot b \cdot h_{id_c^*}}{b \cdot \varrho_k}}\big) = H_2\big(A^{\frac{h_{id_c^*}}{\varrho_k}}\big)\\
\mathtt{Pub}_\mathtt{c}^\mathtt{1k} =& g^{b \cdot \vartheta_k \cdot \mathtt{Priv}_\mathtt{c}^\mathtt{k}} = B^{\vartheta_k \cdot H_2\left(A^{\frac{h_{id_c^*}}{\varrho_k}}\right)}\\
\mathtt{Pub}_\mathtt{c}^\mathtt{2k} =& g^{b \cdot \varrho_k \cdot \mathtt{Priv}_\mathtt{c}^\mathtt{k}} = B^{\varrho_k \cdot H_2\left(A^{\frac{h_{id_c^*}}{\varrho_k}}\right)}\\
\mathtt{Priv}_{\mathtt{ID}_\mathtt{u}}^\mathtt{k} =& g^{\frac{a \cdot b \cdot \mathtt{s}_{\mathtt{ID}_\mathtt{u}^*} + b \cdot \varrho_k}{b \cdot \zeta_k}} = A^{\frac{\mathtt{s}_{\mathtt{ID}_\mathtt{u}^*}}{\zeta_k}} \cdot g^{\frac{\varrho_k}{\zeta_k}}\\
\mathtt{Pub}_{\mathtt{ID}_\mathtt{u}}^\mathtt{k} =& g^{\frac{H_2\left(\mathtt{Priv}_{\mathtt{ID}_\mathtt{u}}^\mathtt{k}\right)}{\mathtt{s}_{\mathtt{ID}_\mathtt{u}^*}}} = g^{\frac{H_2\left(A^{\frac{\mathtt{s}_{\mathtt{ID}_\mathtt{u}^*}}{\zeta_k}} \cdot g^{\frac{\varrho_k}{\zeta_k}}\right)}{\mathtt{s}_{\mathtt{ID}_\mathtt{u}^*}}}\\
\mathtt{Priv}_{\mathtt{ID}_\mathtt{u}} =& \mathtt{s}_{\mathtt{ID}_\mathtt{u}^*}
\end{aligned}$$

Finally, simulator $\mathcal{B}$ sends the following parameters to the adversary $\mathcal{A}_1$: $\big\langle q, \mathbb{G}_1, \mathbb{G}_T, \hat{e}, H_1, H_2, Y, \{h_1^k\}_{\forall k \in \Phi}, \{\{\mathtt{PK}_{\mathtt{r}_\mathtt{i}^\mathtt{k}}\}_{\forall r_i^k \in \Psi_k}\}_{\forall k \in \Phi}, \Big\{\Big\{\mathtt{PKey}_{\mathtt{r}_\mathtt{i}^\mathtt{k}}^{\mathtt{r}_\mathtt{w}^\mathtt{k}}\Big\}_{\forall r_w^k \in \mathbb{R}_{r_i^k} \setminus \{r_i^k\}}\Big\}_{\forall r_i^k \in \Psi_k, \forall k \in \Phi}\big\rangle$. Simulator $\mathcal{B}$ also sends $\{\mathtt{Priv}_\mathtt{c}^\mathtt{k}, \mathtt{Pub}_\mathtt{c}^\mathtt{1k}, \mathtt{Pub}_\mathtt{c}^\mathtt{2k}, \mathtt{Priv}_{\mathtt{ID}_\mathtt{u}}^\mathtt{k}, \mathtt{Pub}_{\mathtt{ID}_\mathtt{u}}^\mathtt{k}\}_{\forall k \in \Phi}$ and $\mathtt{Priv}_{\mathtt{ID}_\mathtt{u}}$ to the adversary $\mathcal{A}$. Note that simulator $\mathcal{B}$ sends a random number $\mathtt{s}_{\mathtt{ID}_\mathtt{u}^*}$ as $\mathtt{Priv}_{\mathtt{ID}_\mathtt{u}}$ to the adversary $\mathcal{A}$. As the simulator $\mathcal{B}$ chooses $\mathtt{s}_{\mathtt{ID}_\mathtt{u}^*}$ in the SETUP and sends it to the adversary $\mathcal{A}$, the simulated game remains the same as the original scheme.

PHASE 1 Adversary sends a challenged role set $\mathbb{S}^*$ to the simulator $\mathcal{B}$ for role-keys. Simulator $\mathcal{B}$ computes $\{\mathtt{RK}_{\mathtt{r}_\mathtt{x}^\mathtt{k}}^{\mathtt{1,u}}, \mathtt{RK}_{\mathtt{r}_\mathtt{x}^\mathtt{k}}^{\mathtt{2,u}}\}_{\forall r_x^k \in \mathbb{S}^*}$ as follows:

For all $r_x^k \in \mathbb{S}^*$, simulator $\mathcal{B}$ computes

$$\begin{aligned}
\mathtt{RK}_{\mathtt{r}_\mathtt{x}^\mathtt{k}}^{\mathtt{1,u}} =& g^{\frac{a \cdot b \cdot \mathtt{Priv}_{\mathtt{ID}_\mathtt{u}} + b \cdot \vartheta_k}{b \prod_{\forall r_j^k \in \mathbb{R}_{r_x^k}} \alpha_{r_j^k}}} = A^{\frac{H_2\left(A^{h_{id_u^*}}\right)}{\prod_{\forall r_j^k \in \mathbb{R}_{r_x^k}} \alpha_{r_j^k}}} \cdot g^{\frac{\vartheta_k}{\prod_{\forall r_j^k \in \mathbb{R}_{r_x^k}} \alpha_{r_j^k}}}\\
\mathtt{RK}_{\mathtt{r}_\mathtt{x}^\mathtt{k}}^{\mathtt{2,u}} =& g^{\frac{a \cdot b \cdot \mathtt{Priv}_{\mathtt{ID}_\mathtt{u}} + b \cdot \vartheta_k}{b \cdot \alpha_{r_x^k}}} = A^{\frac{H_2\left(A^{h_{id_u^*}}\right)}{\alpha_{r_x^k}}} \cdot g^{\frac{\vartheta_k}{\alpha_{r_x^k}}}
\end{aligned}$$

Finally, simulator $\mathcal{B}$ sends $\{\mathtt{RK}_{\mathtt{r}_\mathtt{x}^\mathtt{k}}^{\mathtt{1,u}}, \mathtt{RK}_{\mathtt{r}_\mathtt{x}^\mathtt{k}}^{\mathtt{2,u}}\}_{\forall r_x^k \in \mathbb{S}^*}$ to the adversary $\mathcal{A}_1$. Note that distribution of the role-keys for $\mathbb{S}^*$ is identical to the original scheme.

CHALLENGE When adversary $\mathcal{A}_1$ decides that **PHASE 1** is over, it submits two equal length messages $\mathtt{K}_0$ and $\mathtt{K}_1$ to the simulator $\mathcal{B}$. Simulator $\mathcal{B}$ flips a random binary coin ω and encrypts $\mathtt{K}_\omega$ with the challenged role set Γ^*.

Simulator $\mathcal{B}$ first computes $h_w = H_1(w)$ and chooses five polynomials $q_1(x), q_2(x), q_3(x), q_4(x)$ and $q_5(x)$ of degree $2, |\Gamma_\Phi^*|, |\Gamma_\Phi^*|, |\Gamma^*|$ and $|\Gamma^*|$ respectively, where Γ_Φ^* represents the set of system authorities associated with Γ^*, as follows:

- $q_1(x)$: Simulator $\mathcal{B}$ implicitly sets $q_1(0) = c$ and randomly chooses the rest of the points to define the polynomial $q_1(x)$ completely. Note that $q_1(1)$ and $q_1(2)$ values implicitly represent d_i and d_j of our original scheme respectively.
- $q_2(x)$: Simulator $\mathcal{B}$ sets $q_2(0) = q_1(1)$ and randomly chooses the rest of the points to define $q_2(x)$ completely.
- $q_3(x)$: Simulator $\mathcal{B}$ sets $q_3(0) = q_1(2)$ and randomly chooses the rest of the points to defined $q_3(x)$ completely.

- $q_4(x)$: Simulator $\mathcal{B}$ sets $q_4(0) = q_1(1)$ and randomly chooses the rest of the points to define $q_4(x)$ completely.
- $q_5(x)$: Simulator $\mathcal{B}$ sets $q_5(0) = q_1(2)$ and randomly chooses the rest of the points to define $q_5(x)$ completely.

Now, simulator $\mathcal{B}$ computes a challenged ciphertext $\mathbb{CT}_\omega = \langle C_1, C_2, C_3, \{C_{4k}, C'_{4k}\}_{\forall k \in \Gamma^*_\Phi}, \{C_{r_i^k}, C'_{r_i^k}\}_{\forall r_i^k \in \Gamma^*} \rangle$, where

$$\begin{aligned}
C_1 =& \mathtt{K}_\omega \cdot Z \\
C_2 =& g^{b \cdot \zeta_k \cdot q_1(2)} = B^{\zeta_k \cdot q_1(2)} \\
C_3 =& g^{b \cdot \varrho_k \cdot \mathtt{Priv}_\mathtt{c}^\mathtt{k} \cdot q_1(2)} = B^{\varrho_k \cdot H_2\left(A^{\frac{h_{id_c^*}}{\varrho_k}}\right) \cdot q_1(2)} \\
C_{4k} =& g^{b \cdot \vartheta_k \cdot \mathtt{Priv}_\mathtt{c}^\mathtt{k} \cdot q_2(i)} = B^{\vartheta_k \cdot H_2\left(A^{\frac{h_{id_c^*}}{\varrho_k}}\right) \cdot q_2(i)},\ 1 \le i \le |\Gamma^*_\Phi| \\
C'_{4k} =& g^{b \cdot \vartheta_k \cdot \mathtt{Priv}_\mathtt{c}^\mathtt{k} \cdot q_3(i)} = B^{\vartheta_k \cdot H_2\left(A^{\frac{h_{id_c^*}}{\varrho_k}}\right) \cdot q_3(i)},\ 1 \le i \le |\Gamma^*_\Phi| \\
C_{r_i^k} =& g^{h_w \cdot b \cdot q_4(i) \prod_{\forall r_j^k \in \mathbb{R}_{r_i^k}} \alpha_{r_j^k}},\ 1 \le i \le |\Gamma^*| = B^{h_w \cdot q_4(i) \prod_{\forall r_j^k \in \mathbb{R}_{r_i^k}} \alpha_{r_j^k}} \\
C'_{r_i^k} =& g^{h_w \cdot b \cdot q_5(i) \prod_{\forall r_j^k \in \mathbb{R}_{r_i^k}} \alpha_{r_j^k}},\ 1 \le i \le |\Gamma^*| = B^{h_w \cdot q_5(i) \prod_{\forall r_j^k \in \mathbb{R}_{r_i^k}} \alpha_{r_j^k}}
\end{aligned}$$

Note that c (implicitly) can be recovered using the Lagrange's polynomial interpolation from the values $q_1(1)$ and $q_1(2)$, and $q_1(1)$, $q_1(2)$ can be recovered from the polynomials $q_4(x)$ and $q_5(x)$ if and only if the entity (i.e., adversary $\mathcal{A}_1$) possesses a qualified set of roles. Hence, the distribution of the ciphertext $\mathbb{CT}_\omega$ for Γ^* is identical to the original scheme.

PHASE 2 Same as **PHASE 1**

GUESS The adversary $\mathcal{A}_1$ guesses a bit ω' which is sent to simulator $\mathcal{B}$. If $\omega' = \omega$ then the adversary $\mathcal{A}_1$ wins CPA game; otherwise it fails. If $\omega' = \omega$, simulator $\mathcal{B}$ answers "DBDH" in the game (i.e. outputs $l = 0$); otherwise $\mathcal{B}$ answers "random" (i.e. outputs $l = 1$).

If $Z = \hat{e}(g, g)^z$; then C_1 is completely random from the view of the adversary $\mathcal{A}_1$. So, the received ciphertext $\mathbb{CT}_\omega$ is not compliant to the game (i.e. invalid ciphertext). Therefore, the adversary $\mathcal{A}_1$ chooses ω' randomly. Hence, the probability of the adversary $\mathcal{A}_1$ for outputting $\omega' = \omega$ is $\frac{1}{2}$.

If $Z = \hat{e}(g, g)^{abc}$, then adversary $\mathcal{A}_1$ receives a valid ciphertext. The adversary $\mathcal{A}_1$ wins the CPA game with non-negligible advantage ϵ (according to Theorem 6.1). As such, the probability of outputting $\omega' = \omega$ for the adversary $\mathcal{A}_1$ is $\frac{1}{2} + \epsilon$, where probability ϵ is for guessing that the received ciphertext is valid and probability $\frac{1}{2}$ is for guessing whether the valid encrypted message C_1 is related to $\mathtt{K}_0$ or $\mathtt{K}_1$.

Therefore, the overall advantage $Adv_{\mathcal{A}_1}^{IND-CPA}$ of the simulator $\mathcal{B}$ is $\frac{1}{2}(\frac{1}{2} + \epsilon + \frac{1}{2}) - \frac{1}{2} = \frac{\epsilon}{2}$. □

6.1.2. Security against Chosen Keyword Attack. Chosen keyword attack (CKA) security of the proposed scheme can be defined by the following theorem and proof.

THEOREM 6.2. *If a PPT adversary $\mathcal{A}_2$ wins the CKA security game defined in Section 3.5.2 with a non-negligible advantage ϵ, then a PPT simulator $\mathcal{B}$ can be constructed to break DBDH assumption with non-negligible advantage $\frac{\epsilon}{2}$.*

PROOF. In this proof, we show that a simulator $\mathcal{B}$ can be constructed to help an adversary $\mathcal{A}_2$ to gain advantage $\frac{\epsilon}{2}$ against our proposed scheme.

The DBDH challenger $\mathcal{C}$ chooses random numbers $(a, b, c, z) \in \mathbb{Z}_q^*$ and flips a binary random coin l. It sets $Z = \hat{e}\left(g, g\right)^{abc}$ if $l = 0$ and $Z = \hat{e}\left(g, g\right)^{z}$ otherwise. Afterwards, challenger $\mathcal{C}$ sends $A = g^a, B = g^b, C = g^c$ and Z to the simulator $\mathcal{B}$, and it asks the simulator $\mathcal{B}$ to output l. Now simulator $\mathcal{B}$ acts as a challenger in the rest of the security game.

In the following game simulator $\mathcal{B}$ interacts with the adversary $\mathcal{A}_2$ as follows:

INIT Adversary $\mathcal{A}_2$ sends a challenged role set Γ^* and two identities $(\mathtt{ID}_\mathtt{c}^*, \mathtt{ID}_\mathtt{u}^*)$ to the simulator $\mathcal{B}$.

SETUP Simulator $\mathcal{B}$ chooses random numbers $\{\zeta_k, \vartheta_k, \varrho_k\}_{\forall k \in \Phi}$. It also chooses random numbers $\{\alpha_{r_i^k}\}_{\forall i \in \Psi_k, \forall k \in \Phi} \in \mathbb{Z}_q^*$. Simulator $\mathcal{B}$ computes $Y = \hat{e}\left(g, g\right)^{ab} = \hat{e}\left(A, B\right), \{h_1^k = g^{b \cdot \zeta_k} = B^{\zeta_k}\}_{\forall k \in \Phi}$. It also computes $\mathtt{PK}_{\mathtt{r}_\mathtt{i}^\mathtt{k}} = g^{b \prod_{\forall r_j^k \in \mathbb{R}_{r_i^k}} \alpha_{r_j^k}} = B^{\prod_{\forall r_j^k \in \mathbb{R}_{r_i^k}} \alpha_{r_j^k}}$ for all $r_i^k \in \Psi_k$, where $1 \leq k \leq m$. Moreover, simulator $\mathcal{B}$ computes $\left\{ \left\{ \mathtt{PKey}_{\mathtt{r}_\mathtt{i}^\mathtt{k}}^{\mathtt{r}_\mathtt{w}^\mathtt{k}} = \prod_{\forall r_j^k \in \mathbb{R}_{r_i^k} \setminus \{r_w^k\}} \alpha_{r_j^k} \right\}_{\forall r_w^k \in \mathbb{R}_{r_i^k} \setminus \{r_i^k\}} \right\}_{\forall r_i^k \in \Psi_k}$ where $1 \leq k \leq m$.

Moreover, simulator $\mathcal{B}$ chooses a random number $\mathtt{s}_{\mathtt{ID}_\mathtt{u}^*} \in \mathbb{Z}_q^*$ and computes $h_{id_c^*} = H_1(\mathtt{ID}_\mathtt{c}^*)$. It then computes $\{\mathtt{Priv}_\mathtt{c}^\mathtt{k}, \mathtt{Pub}_\mathtt{c}^\mathtt{1k}, \mathtt{Pub}_\mathtt{c}^\mathtt{2k}, \mathtt{Pub}_{\mathtt{ID}_\mathtt{u}}^\mathtt{k}\}_{\forall k \in \Phi}$, where

$$\begin{aligned}
\mathtt{Priv}_\mathtt{c}^\mathtt{k} =& H_2\big(g^{\frac{a \cdot b \cdot h_{id_c^*}}{b \cdot \varrho_k}}\big) = H_2\big(A^{\frac{h_{id_c^*}}{\varrho_k}}\big) \\
\mathtt{Pub}_\mathtt{c}^\mathtt{1k} =& g^{b \cdot \vartheta_k \cdot \mathtt{Priv}_\mathtt{c}^\mathtt{k}} = B^{\vartheta_k \cdot H_2\left(A^{\frac{h_{id_c^*}}{\varrho_k}}\right)} \\
\mathtt{Pub}_\mathtt{c}^\mathtt{2k} =& g^{b \cdot \varrho_k \cdot \mathtt{Priv}_\mathtt{c}^\mathtt{k}} = B^{\varrho_k \cdot H_2\left(A^{\frac{h_{id_c^*}}{\varrho_k}}\right)} \\
\mathtt{Pub}_{\mathtt{ID}_\mathtt{u}}^\mathtt{k} =& g^{\frac{H_2\left(g^{\frac{a \cdot b \cdot \mathtt{s}_{\mathtt{ID}_\mathtt{u}^*} + b \cdot \varrho_k}{b \cdot \zeta_k}}\right)}{\mathtt{s}_{\mathtt{ID}_\mathtt{u}^*}}} = g^{\frac{H_2\left(A^{\frac{\mathtt{s}_{\mathtt{ID}_\mathtt{u}^*}}{\zeta_k}} \cdot g^{\frac{\varrho_k}{\zeta_k}}\right)}{\mathtt{s}_{\mathtt{ID}_\mathtt{u}^*}}}
\end{aligned}$$

Simulator $\mathcal{B}$ sends the following parameters to the adversary $\mathcal{A}_2$: $\big\langle q, \mathbb{G}_1, \mathbb{G}_T, \hat{e}, H_1, H_2, Y, \{h_1^k\}_{\forall k \in \Phi}, \{\{\mathtt{PK}_{\mathtt{r}_\mathtt{i}^\mathtt{k}}\}_{\forall r_i^k \in \Psi_k}\}_{\forall k \in \Phi},$ $\left\{ \left\{ \mathtt{PKey}_{\mathtt{r}_\mathtt{i}^\mathtt{k}}^{\mathtt{r}_\mathtt{w}^\mathtt{k}} \right\}_{\forall r_w^k \in \mathbb{R}_{r_i^k} \setminus \{r_i^k\}} \right\}_{\forall r_i^k \in \Psi_k, \forall k \in \Phi}, \{\mathtt{Pub}_\mathtt{c}^\mathtt{1k}, \mathtt{Pub}_\mathtt{c}^\mathtt{2k},$ $\mathtt{Pub}_{\mathtt{ID}_\mathtt{u}}^\mathtt{k}\}_{\forall k \in \Phi} \big\rangle$. Simulator $\mathcal{B}$ also sends private keys $\{\mathtt{Priv}_\mathtt{c}^\mathtt{k}\}_{\forall k \in \Phi}$ and $\mathtt{Priv}_{\mathtt{ID}_\mathtt{u}} = \mathtt{s}_{\mathtt{ID}_\mathtt{u}^*}$ to the adversary $\mathcal{A}_2$.

PHASE 1 Adversary $\mathcal{A}_2$ sends a set of roles $\mathbb{S}^*$ and a keyword w to the simulator $\mathcal{B}$ for the trapdoor. Simulator $\mathcal{B}$ chooses random numbers $(\mathbb{v}, \mathbb{ts}) \in \mathbb{Z}_q^*$. It computes $h_w = H_1(w)$. Simulator $\mathcal{B}$ computes the trapdoor $\mathtt{Trap} = \big\langle tr_1, tr_2, tr_3, tr_4, \{tr_{r_x^k}^1, tr_{r_x^k}^2\}_{\forall r_x^k \in \mathbb{S}^*} \big\rangle,$

where

$$\begin{aligned} tr_1 =& \frac{\mathtt{s}_{\mathtt{ID}_{\mathtt{u}}^*} + \mathbb{ts}}{H_2\left(g^{\frac{a \cdot b \cdot \mathtt{s}_{\mathtt{ID}_{\mathtt{u}}^*} + b \cdot \varrho_k}{b \cdot \zeta_k}}\right)} \cdot \mathbb{v} = \frac{\left[\mathtt{s}_{\mathtt{ID}_{\mathtt{u}}^*} + \mathbb{ts}\right] \mathbb{v}}{H_2\left(A^{\frac{\mathtt{s}_{\mathtt{ID}_{\mathtt{u}}^*}}{\zeta_k}} \cdot g^{\frac{\varrho_k}{\zeta_k}}\right)} \\ tr_2 =& \left(g^{\frac{a \cdot b \cdot \mathtt{s}_{\mathtt{ID}_{\mathtt{u}}^*} + b \cdot \varrho_k}{b \cdot \zeta_k}}\right)^{\mathbb{v}} = A^{\frac{\mathtt{s}_{\mathtt{ID}_{\mathtt{u}}^*} \cdot \mathbb{v}}{\zeta_k}} \cdot g^{\frac{\varrho_k \cdot \mathbb{v}}{\zeta_k}} \\ tr_3 =& g^{\frac{\mathbb{v}}{\mathtt{s}_{\mathtt{ID}_{\mathtt{u}}^*}}} = g^{\frac{\mathbb{v}}{\mathtt{s}_{\mathtt{ID}_{\mathtt{u}}^*}}} \\ tr_4 =& g^{\mathbb{v}} \end{aligned}$$

For all $r_x^k \in \mathbb{S}^*$,

$$\begin{aligned} tr^1_{r_x^k} &= \left(g^{\frac{a \cdot b \cdot \mathtt{s}_{\mathtt{ID}_{\mathtt{u}}^*} + b \cdot \vartheta_k}{b \prod_{\forall r_j^k \in \mathbb{R}_{r_x^k}} \alpha_{r_j^k}}}\right)^{\frac{\mathbb{v}}{h_w}} = A^{\frac{\mathtt{s}_{\mathtt{ID}_{\mathtt{u}}^*} \cdot \mathbb{v}}{h_w \prod_{\forall r_j^k \in \mathbb{R}_{r_x^k}} \alpha_{r_j^k}}} \cdot g^{\frac{\vartheta_k \cdot \mathbb{v}}{h_w \prod_{\forall r_j^k \in \mathbb{R}_{r_x^k}} \alpha_{r_j^k}}} \\ tr^2_{r_x^k} &= \left(g^{\frac{a \cdot b \cdot \mathtt{s}_{\mathtt{ID}_{\mathtt{u}}^*} + b \cdot \vartheta_k}{b \cdot \alpha_{r_x^k}}}\right)^{\frac{\mathbb{v}}{h_w}} = A^{\frac{\mathtt{s}_{\mathtt{ID}_{\mathtt{u}}^*} \cdot \mathbb{v}}{h_w \cdot \alpha_{r_x^k}}} \cdot g^{\frac{\vartheta_k \cdot \mathbb{v}}{h_w \cdot \alpha_{r_x^k}}} \end{aligned}$$

Finally, simulator $\mathcal{B}$ sends trapdoor $\mathtt{Trap}$ to the adversary $\mathcal{A}_2$.

CHALLENGE When adversary $\mathcal{A}_2$ decides that **PHASE 1** is over, it submits two equal length keywords w_0 and w_1 to the simulator $\mathcal{B}$. Simulator $\mathcal{B}$ flips a random binary coin ω and encrypts w_ω with the challenged role set Γ^*.

Simulator $\mathcal{B}$ first computes $h_{w_\omega} = H_1(w_\omega)$. It then chooses a random element $\mathtt{K} \in \mathbb{G}_T$ and five polynomials $q_1(x), q_2(x), q_3(x), q_4(x)$ and $q_5(x)$ of degree $2, |\Gamma^*_\Phi|, |\Gamma^*_\Phi|, |\Gamma^*|$ and $|\Gamma^*|$ respectively as follows:

- $q_1(x)$: Simulator $\mathcal{B}$ implicitly sets $q_1(0) = c$ and randomly chooses the rest of the points to define the polynomial $q_1(x)$ completely. Note that $q_1(1)$ and $q_1(2)$ implicitly represent d_i and d_j of our original scheme respectively.
- $q_2(x)$: Simulator $\mathcal{B}$ sets $q_2(0) = q_1(1)$ and randomly chooses the rest of the points to define $q_2(x)$ completely.
- $q_3(x)$: Simulator $\mathcal{B}$ sets $q_3(0) = q_1(2)$ and randomly chooses the rest of the points to defined $q_3(x)$ completely.
- $q_4(x)$: Simulator $\mathcal{B}$ sets $q_4(0) = q_1(1)$ and randomly chooses the rest of the points to define $q_4(x)$ completely.
- $q_5(x)$: Simulator $\mathcal{B}$ sets $q_5(0) = q_1(2)$ and randomly chooses the rest of the points to define $q_5(x)$ completely.

Table II: NOTATIONS

Notation	Description
$\lvert\Gamma\rvert$	Total number of roles associated with a ciphertext
$\lvert\Gamma_\Phi\rvert$	Total number of SAs associated with Γ (i.e., ciphertext)
$\lvert\mathbb{S}_{\mathtt{ID_u}}\rvert$	Total number of roles associated with a trapdoor
n_c	Total number of ciphertext associated with a revoked role
n_u	Total number users associated with a revoked role
n_s	Total number SA associated with a user

Now, simulator $\mathcal{B}$ computes a challenged ciphertext $\mathbb{CT}_\omega = \langle C_1, C_2, C_3, \{C_{4k}, C'_{4k}\}_{\forall k \in \Gamma^*_\Phi}, \{C_{r_i^k}, C'_{r_i^k}\}_{\forall r_i^k \in \Gamma^*} \rangle$, where

$$\begin{aligned}
C_1 =& \mathtt{K} \cdot Z \\
C_2 =& g^{b \cdot \zeta_k \cdot q_1(2)} = B^{\zeta_k \cdot q_1(2)} \\
C_3 =& g^{b \cdot \varrho_k \cdot H_2\left(g^{\frac{a \cdot b \cdot h_{id_c^*}}{b \cdot \varrho_k}}\right) \cdot q_1(2)} = B^{\varrho_k \cdot H_2\left(A^{\frac{h_{id_c^*}}{\varrho_k}}\right) \cdot q_1(2)} \\
C_{4k} =& \left\{ g^{b \cdot \vartheta_k \cdot H_2\left(g^{\frac{a \cdot b \cdot h_{id_c^*}}{b \cdot \varrho_k}}\right) \cdot q_2(i)} = B^{\vartheta_k \cdot H_2\left(A^{\frac{h_{id_c^*}}{\varrho_k}}\right) \cdot q_2(i)} \right\},\ 1 \le i \le |\Gamma^*_\Phi| \\
C'_{4k} =& \left\{ g^{b \cdot \vartheta_k \cdot H_2\left(g^{\frac{a \cdot b \cdot h_{id_c^*}}{b \cdot \varrho_k}}\right) \cdot q_3(i)} = B^{\vartheta_k \cdot H_2\left(A^{\frac{h_{id_c^*}}{\varrho_k}}\right) \cdot q_3(i)} \right\},\ 1 \le i \le |\Gamma^*_\Phi| \\
C_{r_i^k} =& \left\{ g^{h_{w_\omega} \cdot b \cdot q_4(i) \prod_{\forall r_j^k \in \mathbb{R}_{r_i^k}} \alpha_{r_j^k}} = B^{h_{w_\omega} \cdot q_4(i) \prod_{\forall r_j^k \in \mathbb{R}_{r_i^k}} \alpha_{r_j^k}} \right\},\ 1 \le i \le |\Gamma^*| \\
C'_{r_i^k} =& \left\{ g^{h_{w_\omega} \cdot b \cdot q_5(i) \prod_{\forall r_j^k \in \mathbb{R}_{r_i^k}} \alpha_{r_j^k}} = B^{h_{w_\omega} \cdot q_5(i) \prod_{\forall r_j^k \in \mathbb{R}_{r_i^k}} \alpha_{r_j^k}} \right\},\ 1 \le i \le |\Gamma^*|
\end{aligned}$$

Similar with CPA proof 6.1.1, the distribution of the ciphertext $\mathbb{CT}_\omega$ for Γ^* is identical to the original scheme.

PHASE 2 Same as **PHASE 1**

GUESS The adversary $\mathcal{A}_2$ guesses a bit ω' and sends to the simulator $\mathcal{B}$. If $\omega' = \omega$ then the adversary $\mathcal{A}_2$ wins CPA game; otherwise it fails. If $\omega' = \omega$, simulator $\mathcal{B}$ answers "DBDH" in the game (i.e. outputs $l = 0$); otherwise $\mathcal{B}$ answers "random" (i.e. outputs $l = 1$).

If $Z = \hat{e}(g, g)^z$; then C_1 is completely random from the view of the adversary $\mathcal{A}_2$. So, the received ciphertext $\mathbb{CT}_\omega$ is not compliant to the game (i.e. invalid ciphertext). Therefore, the adversary $\mathcal{A}_2$ chooses ω' randomly. Hence, the probability of the adversary $\mathcal{A}_2$ for outputting $\omega' = \omega$ is $\frac{1}{2}$.

If $Z = \hat{e}(g, g)^{abc}$, then adversary $\mathcal{A}_2$ receives a valid ciphertext. The adversary $\mathcal{A}_2$ wins the CPA game with non-negligible advantage ϵ (according to the Theorem 6.2). As such, the probability of outputting $\omega' = \omega$ for the adversary $\mathcal{A}_2$ is $\frac{1}{2} + \epsilon$, where probability ϵ is for guessing that the received ciphertext is valid and probability $\frac{1}{2}$ is for guessing whether the valid encrypted message C_1 is related to w_0 or w_1.

Therefore, the overall advantage $Adv_{\mathcal{A}_2}^{IND-CKA}$ of the simulator $\mathcal{B}$ is $\frac{1}{2}(\frac{1}{2} + \epsilon + \frac{1}{2}) - \frac{1}{2} = \frac{\epsilon}{2}$. □

6.2. Performance Analysis

This section evaluates functionality, computation, storage and communication overhead of our proposed scheme. The computational overhead is shown in terms of num-

Table III: Functionality Comparison

	Authorized Keyword Search	Authentication	Replay Attack	Conjunctive Keyword Search	Revocation	Decryption	Technique
[Sun et al. 2016]	✓	✗	✗	✓	✓	✗	ABE
[Hu et al. 2017]	✓	✗	✗	✗	✗	✗	ABE
[Miao et al. 2017]	✓	✗	✗	✓	✓	✓	ABE
[Chaudhari and Das 2019]	✓	✗	✗	✗	✗	✗	ABE
Proposed scheme	✓	✓	✓	✓	✓	✓	RBE

Table IV: Evaluation of the Computation Overhead

Operations		Computation Complexity
Data Encryption		$(4+\|\Gamma\|+\|\Gamma_\Phi\|)Exp_{\mathbb{G}_1}+Exp_{\mathbb{G}_T}$
Trapdoor Generation		$(3+2\|\mathbb{S}_{\mathtt{ID_u}}\|)Exp_{\mathbb{G}_1}$
Data Search	Authentication	$(2+\|\Gamma_\Phi\|)Exp_{\mathbb{G}_1}+3T_p$
	KeySearch	$< ((\|\Gamma\|+1)Exp_{\mathbb{G}_1}+(2+\|\Gamma\|)T_p$
	PartialDec	$< ((\|\Gamma\|+\|\Gamma_\Phi\|)Exp_{\mathbb{G}_1}+(1+\|\Gamma\|)T_p$
Decryption		$Exp_{\mathbb{G}_T}$
Revocation		$< (1+2n_c+2n_u)Exp_{\mathbb{G}_1}$

Table V: Evaluation of the Storage and Communication Overhead

Items	Overhead
Ciphertext	$(4+2\|\Gamma\|)\|\mathbb{G}_1\|+\|\mathbb{G}_T\|$
Secret key	$(1+n_s)\|\mathbb{Z}_q^*\|+2\|\mathbb{S}_{\mathtt{ID_u}}\|\|\mathbb{G}_1\|$
Trapdoor	$\|\mathbb{Z}_q^*\|+(3+2\|\Gamma\|)\|\mathbb{G}_1\|$

Table VI: Computation Time (in Milliseconds) of Elementary Cryptographic Operations

	Exponentiation		Pairing	Group multiplication		Hash
	$\mathbb{G}_1$	$\mathbb{G}_T$		$\mathbb{G}_1$	$\mathbb{G}_T$	
Commodity Laptop	2.062	0.126	1.292	0.008	0.002	0.003
Workstation	1.153	0.091	0.645	0.005	0.001	0.002

ber of pairing (T_p) and group exponentiation operations ($Exp_{\mathbb{G}_1}$ and $Exp_{\mathbb{G}_T}$). We do not consider the other cryptographic operations such as hash and group element multiplication operations, as these operations take much less computation time compared with the pairing and group exponentiation operations (details can be seen in the Table VI). The storage and communication overheads are shown in terms of group element size $|\mathbb{Z}_q^*|, |\mathbb{G}_1|$ and $|\mathbb{G}_T|$. We use PBC library [pbc] which runs over GMP library [gmp] for the implementation purpose. Type A elliptic curve of 160-bit group order embedding degree 2 is used for the implementation. The chosen curve provides an equivalent of

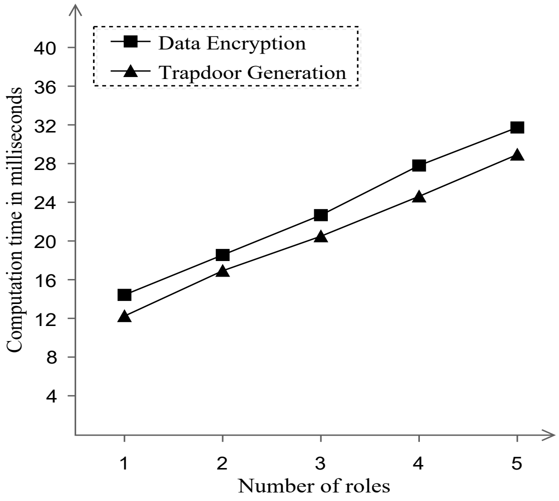


Fig. 4: Computation Time of *Data Encryption* and *Trapdoor Generation* Phases

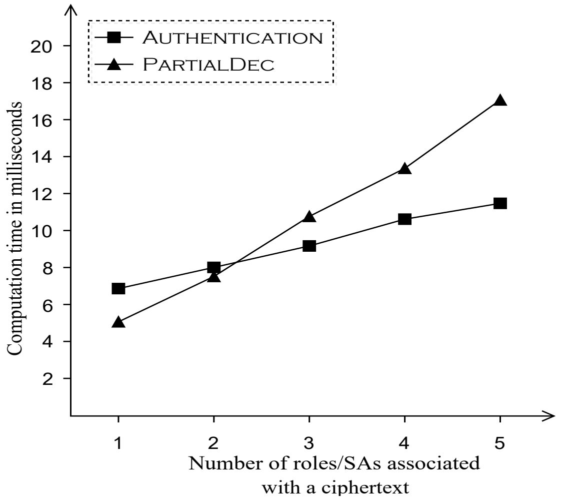


Fig. 5: Computation Time of AUTHENTICATION and PARTIALDEC Algorithms

1024-bit discrete log security. The elementary cryptographic operations that are performed by the owners and users are implemented using a commodity laptop Computer with Ubuntu 17.10 (64-bit) operating system and having 2.4GHz Core i3 processor with 4GB memory. The elementary cryptographic operations that are performed by public cloud is implemented using a workstation with Ubuntu 17.10 (64-bit) operating system and having 3.5 GHz Intel(R) Xeon(R) CPU E5-2637 v4 processor with 16 GB memory. Table VI shows the time required to perform each cryptographic operations. During the implementation, we consider that the number of SAs associated with a RBAC access policy is equal to the number of roles associated with a ciphertext, i.e., $|\Gamma_{\phi}| = |\Gamma|$. It is to be noted that all the implementation results are the mean of 50 trials. The notations used in the rest of this paper are shown in Table II.

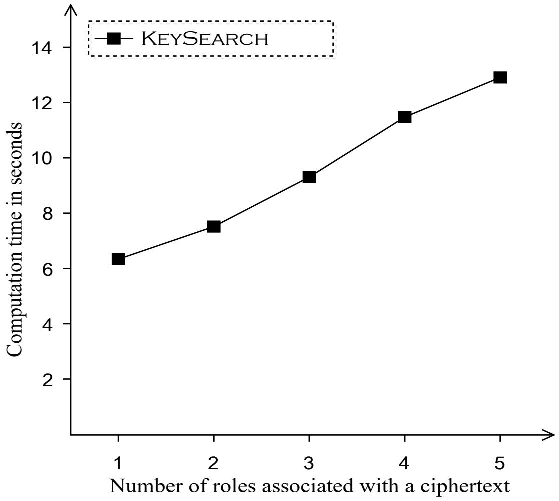


Fig. 6: Computation Time of KEYSEARCH Algorithm for 1000 Ciphertexts

Table III shows the functionality comparison of some notable ABE based keyword search schemes [Sun et al. 2016], [Hu et al. 2017], [Miao et al. 2017], [Chaudhari and Das 2019] with our proposed scheme. From the Table III, it can be observed that all the ABE based schemes [Sun et al. 2016], [Hu et al. 2017], [Miao et al. 2017], [Chaudhari and Das 2019] including our proposed scheme provide authorized keyword search functionality, as the owner can embed access policies of his/her choice on the encrypted data itself. However, unlike our proposed scheme, none of the schemes in [Sun et al. 2016], [Hu et al. 2017], [Miao et al. 2017], [Chaudhari and Das 2019] address the user authentication problem, which allows the public cloud to authenticate the user before performing computationally expensive keyword search operations. As such, [Sun et al. 2016], [Hu et al. 2017], [Miao et al. 2017], [Chaudhari and Das 2019] rely on some existing authentication mechanisms. Also, unlike [Sun et al. 2016], [Hu et al. 2017], [Miao et al. 2017], [Chaudhari and Das 2019], our proposed scheme can prevent the replay attacks even if the trapdoors are exposed to the adversaries. In [Sun et al. 2016], [Hu et al. 2017], [Miao et al. 2017], [Chaudhari and Das 2019], if an adversary gains access to a valid trapdoor, the adversary can re-use the trapdoor using a fresh random number. Further, our proposed scheme and [Sun et al. 2016], [Miao et al. 2017] support conjunctive keyword search and user revocation, while [Hu et al. 2017], [Chaudhari and Das 2019] do not support. Moreover, our proposed scheme and [Miao et al. 2017] support both the keyword search and decryption functionalities; while [Sun et al. 2016], [Hu et al. 2017], [Chaudhari and Das 2019] support only the keyword search functionality. Furthermore, [Sun et al. 2016], [Hu et al. 2017], [Miao et al. 2017], [Chaudhari and Das 2019] are designed using ABE technique; while our proposed scheme is designed using RBE technique, which enables it to support the role hierarchy property. Thus, it makes our proposed scheme more suitable for the real world organizations/enterprises. Therefore, it can be observed that our proposed scheme supports more functionalities compared with the other notable works [Sun et al. 2016], [Hu et al. 2017], [Miao et al. 2017], [Chaudhari and Das 2019].

Table IV shows the computation overhead of our proposed scheme[9]. The computation cost is shown in asymptotic upper bound in the worst cases. In Table IV, we consider the most frequently operated phases, e.g., *Data Encryption*, *Trapdoor Generation*, *Data Search*, *Decryption*, and *Revocation*.

Data Encryption. Owner encrypts the plaintext data and the associated keywords in the *Data Encryption* phase, which requires $(4 + 2|\Gamma|)$ group exponentiation operations on $\mathbb{G}_1$ and one exponentiation operation on $\mathbb{G}_T$. It can be observed that the encryption cost mainly depends on the number of roles associated with a ciphertext (i.e., associated with the chosen RBAC access policy). This can also be seen from the Figure 4. It can be observed that approximately 31 milliseconds are required to generate a ciphertext associated with 5 roles and 5 SAs. It is to be noted that, the encryption operation is performed by the owner only once for a particular data.

Trapdoor Generation. A user needs to perform $(3{+}2|\mathbb{S}_{\mathtt{ID_u}}|)$ group exponentiation operations on $\mathbb{G}_1$ to compute a trapdoor. It can be observed that the cost for the generation of a trapdoor depends on the number of roles associated with the user (i.e., associated with the trapdoor). Figure 4, shows the experimental results of the *Trapdoor Generation* phase, which demonstrates that our proposed scheme incurs less computation overhead on the user side. It takes approximately 29 milliseconds to generate a trapdoor having 5 roles.

Data Search. In the *Data Search* phase, the public cloud first authenticates the user which requires $2 + |\Gamma_\Phi|$ group exponentiation operations on $\mathbb{G}_1$ and three pairing operations. It can be observed that the cost of the user authentication operation (i.e., AUTHENTICATION algorithm) depends on the number of SAs associated with the RBAC access policy of a ciphertext. Figure 5 shows the computation time of AUTHENTICATION algorithm with respect to the number of SAs. It is to be noted that the AUTHENTICATION algorithm is performed only once per user request. After successful authentication of the user, the public cloud computes at most $|\Gamma| + 1$ group exponentiation operations on $\mathbb{G}_1$, and $2 + |\Gamma|$ pairing operations to complete the KEYSEARCH algorithm for the keyword search. It can be observed that the cost of the KEYSEARCH algorithm depends on the number of roles associated with the ciphertext, which can also be seen from the Figure 6. Finally, the public cloud computes at most $|\Gamma| + |\Gamma_\Phi|$ group exponentiation operations and $1 + |\Gamma|$ pairing operations to compute the PARTIALDEC algorithm. It can be observed that the cost to perform the PARTIALDEC algorithm depends on the number of roles and the number of SAs associated with the RBAC access policy. The computation time of PARTIALDEC algorithm is shown in the Figure 5. It is to be noted that the PARTIALDEC algorithm is performed for each ciphertext received from the KEYSEARCH algorithm.

Decryption. As most of the computationally expensive cryptographic operations are outsourced to the public cloud, a user requires only one group exponentiation operation on $\mathbb{G}_T$ to decrypt a ciphertext. It is to be noted that the time required to perform one group exponentiation operation on $\mathbb{G}_T$ is 0.126 milliseconds in a commodity laptop Computer. Hence, the decryption cost in our proposed scheme is considerably less. Thus, our proposed scheme is also suitable for an environment such as IoT, where the end-users have limited computing resources.

Revocation. The complete user revocation operation takes a minimal overhead in the system, as the SA can revoke the user simply by revoking (or removing) his/her

[9] We do not consider [Sun et al. 2016], [Hu et al. 2017], [Miao et al. 2017], [Chaudhari and Das 2019] for further comparison, as they are based on ABE; whereas our proposed scheme is based on RBE.

public key (from the public bulletin board). On the other hand, the SA requires at most $1 + 2n_c + 2n_u$ group exponentiation operations on $\mathbb{G}_1$ to revoke a role from a user. As the SA needs to re-encrypt all the ciphertexts and update role-keys of all the users related with the revoked roles, the cost of the role-level revocation depends mainly on the number of ciphertext and users associated with the revoked roles.

6.2.1. Storage and Communication Overhead Comparison. Table V shows the storage and communication overhead of our proposed scheme. For the evaluation purpose, the ciphertext size, size of the secret keys possessed by a user, and the trapdoor size are considered. From Table V, it can be observed that the ciphertext size mainly depends on the number of roles associated with the ciphertext. For each role r_i^k, the owner computes two ciphertext components $C_{r_i^k}$ and $C'_{r_i^k}$. Hence, the ciphertext size linearly increases with the roles associated with a ciphertext.

A user keeps a private key $\mathtt{Priv}^{\mathtt{k}}_{\mathtt{ID_u}}$ for each organization, and the user also keeps a common private key $\mathtt{Priv}_{\mathtt{ID_u}}$ for all the organizations. Moreover, the user keeps two role-keys for each role he/she possessed. Thus, the size of the secret key possessed by a user mainly depends on the number of SAs (i.e., number of organizations) and the number of roles associated with that user. Similarly, trapdoor size linearly increases with the roles associated with the trapdoor. The user computes two trapdoor components $tr^1_{r_x^k}$ and $tr^2_{r_x^k}$ for each role r_x^k associated with the trapdoor.

7. CONCLUSION

This paper has proposed a novel authorized keyword search mechanism with efficient decryption using the RBE technique for a cloud environment, where multiple organizations can outsource their sensitive data. The proposed scheme enables the owners to define and enforce RBAC access policies on the encrypted data, thereby avoiding reducing the dependency on the service provider. It also enables the public cloud to authenticate the users first before performing computationally expensive search operations, which reduces overhead on the system. In addition, the proposed scheme helps to prevent replay attacks. Conjunctive keyword search is supported without introducing any significant overhead into the system. Further, the complete and role-level user revocation mechanisms are supported for revoking access privileges of the users in both organization level and role level respectively. Moreover, an outsourced decryption mechanism is introduced in the proposed scheme to reduce decryption processing cost at the end-user side, which makes it suitable for resource constrained environment. Furthermore, we have formally proved that the proposed scheme provides provable security against Chosen Plaintext and Chosen Keyword Attacks. Our performance analysis shows that the proposed scheme is suitable for real-world applications in terms of computation, communication and storage overhead.

This paper has introduced a new direction in designing a searchable encryption mechanism using the RBE technique. Further works include improving the efficiency of role-level revocation of RBE based keyword search schemes as well as for dynamic addition (removal) of roles into (from) a role hierarchy.

ACKNOWLEDGMENTS

This paper is supported in part by European Union's Horizon 2020 research and innovation programme under the grant agreement No 830892, project SPARTA.

REFERENCES

PBC (Pairing-Based Cryptography) library. http://crypto.stanford.edu/pbc/ [Online accessed: 12-August.-2019].

GMP(GNU Multiple Precision) arithmetic library. http://gmplib.org/ [Online accessed: 12-August-2019].

S. G. Akl and P. D. Taylor. 1983. Cryptographic Solution to a Problem of Access Control in a Hierarchy. *ACM Transactions on Computer Systems* 1, 3 (Aug. 1983), 239–248.

F. Bao, R. H. Deng, X. Ding, and Y. Yang. 2008. Private Query on Encrypted Data in Multi-user Settings. In *Proceedings of the 4th International Conference on Information Security Practice and Experience (ISPEC'08)*. 71–85.

J. Bethencourt, A. Sahai, and B. Waters. 2007. Ciphertext-Policy Attribute-Based Encryption. In *2007 IEEE Symposium on Security and Privacy (SP '07)*. 321–334.

D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. 2004. Public Key Encryption with Keyword Search. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology (EUROCRYPT'04)*. 506–522.

D. Boneh and B. Waters. 2007. Conjunctive, Subset, and Range Queries on Encrypted Data. In *Proceedings of the 4th Conference on Theory of Cryptography (TCC'07)*. 535–554.

Christoph Bösch, Pieter Hartel, Willem Jonker, and Andreas Peter. 2014. A Survey of Provably Secure Searchable Encryption. *ACM Comput. Surv.* 47, 2 (Aug. 2014), 18:1–18:51.

M. Burmester and Y. Desmedt. 2005. A Secure and Scalable Group Key Exchange System. *Inform. Process. Lett.* 94, 3 (May 2005), 137–143.

P. Chaudhari and M. L. Das. 2019. Privacy Preserving Searchable Encryption with Fine-grained Access Control. *IEEE Transactions on Cloud Computing* (2019), 1–1. DOI:http://dx.doi.org/10.1109/TCC.2019.2892116

Y. R. Chen and W. G. Tzeng. 2017. Hierarchical Key Assignment with Dynamic Read-Write Privilege Enforcement and Extended KI-Security. In *Proceedings of the Applied Cryptography and Network Security*. 165–183.

R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. 2006. Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06)*. 79–88.

D. X. Song, D. Wagner, and A. Perrig. 2000. Practical techniques for searches on encrypted data. In *Proceeding 2000 IEEE Symposium on Security and Privacy. S P 2000*. 44–55.

J. D. Ferrer, O. Farrs, J. Ribes-Gonzlez, and D. Snchez. 2019. Privacy-preserving cloud computing on sensitive data: A survey of methods, products and challenges. *Computer Communications* 140-141 (2019), 38 – 60.

Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. 2006. Attribute-based Encryption for Fine-grained Access Control of Encrypted Data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06)*. 89–98.

F. Han, J. Qin, and J. Hu. 2016. "Secure searches in the cloud: A survey". *Future Generation Computer Systems* 62 (2016), 66 – 75.

T. Hoang, A. A. Yavuz, and J. Guajardo Merchan. 2019. A Secure Searchable Encryption Framework for Privacy-Critical Cloud Storage Services. *IEEE Transactions on Services Computing* (2019), 1–1. DOI:http://dx.doi.org/10.1109/TSC.2019.2897096

B. Hu, Q. Liu, X. Liu, T. Peng, G. Wang, and J. Wu. 2017. DABKS: Dynamic attribute-based keyword search in cloud computing. In *Proceedings of the 2017 IEEE International Conference on Communications (ICC)*. 1–6.

S. Kamara, C. Papamanthou, and T. Roeder. 2012. Dynamic Searchable Symmetric Encryption. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS '12)*. 965–976.

J. Li, Y. Huang, Y. Wei, S. Lv, Z. Liu, C. Dong, and W. Lou. 2019. Searchable Symmetric Encryption with Forward Search Privacy. *IEEE Transactions on Dependable and Secure Computing* (2019), 1–1. DOI:http://dx.doi.org/10.1109/TDSC.2019.2894411

M. Li, S. Yu, N. Cao, and W. Lou. 2011. Authorized Private Keyword Search over Encrypted Data in Cloud Computing. In *2011 31st International Conference on Distributed Computing Systems*. 383–392.

Y.L. Lin and C. L. Hsu. 2011. Secure key management scheme for dynamic hierarchical access control based on ECC. *Journal of Systems and Software* 84, 4 (2011), 679 – 685.

X. Liu, G. Yang, Y. Mu, and R. Deng. 2018. Multi-user Verifiable Searchable Symmetric Encryption for Cloud Storage. *IEEE Transactions on Dependable and Secure Computing* (2018), 1–1. DOI:http://dx.doi.org/10.1109/TDSC.2018.2876831

J. M. Marn Prez, G. M. Prez, and A. F. Skarmeta Gomez. 2017. SecRBAC: Secure data in the Clouds. *IEEE Transactions on Services Computing* 10, 5 (Sep. 2017), 726–740.

McAfee. White paper, 2018. Navigating a Cloudy Sky: Practical Guidance and the State of Cloud Security.

P. Mell and T. Grance. The NIST definition of cloud computing. Technical report, National Institute of Standards and Technology, 2009. http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf [Online accessed: 7-Feb.-2019].

Y. Miao, J. Ma, X. Liu, X. Li, Q. Jiang, and J. Zhang. 2017. Attribute-Based Keyword Search over Hierarchical Data in Cloud Computing. *IEEE Transactions on Services Computing* (2017), 1–1. DOI:http://dx.doi.org/10.1109/TSC.2017.2757467

G. Pareek and B. R. Purushothama. 2018. Efficient Strong Key Indistinguishable Access Control in Dynamic Hierarchies with Constant Decryption Cost. In *Proceedings of the 11th International Conference on Security of Information and Networks (SIN '18)*. 10:1–10:7.

R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. 1996. Role-based access control models. *Computer* 29, 2 (Feb 1996), 38–47.

W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li. 2016. Protecting Your Right: Verifiable Attribute-Based Keyword Search with Fine-Grained Owner-Enforced Search Authorization in the Cloud. *IEEE Transactions on Parallel and Distributed Systems* 27, 4 (April 2016), 1187–1198.

S. Tang, X. Li, X. Huang, Y. Xiang, and L. Xu. 2016. Achieving Simple, Secure and Efficient Hierarchical Access Control in Cloud Computing. *IEEE Trans. Comput.* 65, 7 (July 2016), 2325–2331.

L. Zhou, V. Varadharajan, and M. Hitchens. 2011. Enforcing Role-Based Access Control for Secure Data Storage in the Cloud. *Comput. J.* 54, 10 (Oct. 2011), 1675–1687. DOI:http://dx.doi.org/10.1093/comjnl/bxr080

L. Zhou, V. Varadharajan, and M. Hitchens. 2013. Achieving Secure Role-Based Access Control on Encrypted Data in Cloud Storage. *IEEE Transactions on Information Forensics and Security* 8, 12 (Dec 2013), 1947–1960. DOI:http://dx.doi.org/10.1109/TIFS.2013.2286456

Y. Zhu, G. Ahn, H. Hu, D. Ma, and S. Wang. 2013. Role-Based Cryptosystem: A New Cryptographic RBAC System Based on Role-Key Hierarchy. *IEEE Transactions on Information Forensics and Security* 8, 12 (Dec 2013), 2138–2153.