Securing Organization’s Data: A Role-Based Authorized Keyword Search Scheme With Efficient Decryption

Nazatul Haque Sultan©, Member, IEEE, Maryline Laurent©, Member, IEEE, and Vijay Varadharajan, Senior Member, IEEE

Abstract—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 article, we propose a novel authorized keyword search scheme using Role-Based Encryption (RBE) technique in a cloud environment. The contributions of this article are multi-fold. First, it presents a keyword search scheme which enables only authorized users, having properly 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 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.

Index Terms—Role-based encryption, role-based access control, searchable encryption, keyword search, outsourced decryption, provable security, cloud data privacy

1 INTRODUCTION

Searchable Encryption (SE) has gained a considerable amount of interest from the research community to address the issue of searching over encrypted data [1]. It has already been considered as one of the suitable cryptographic techniques that can prevent unauthorized entities including the cloud service provider from accessing and knowing any useful information about the outsourced sensitive data to the cloud [2]. SE enables the users to 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 [3]. 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”, it can get almost 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 [4], Predicate Encryption [5] and Attribute-Based Encryption [3], [6], where multiple users are able to perform
In an organization, typically employees are organized in a hierarchical structure where higher level authorities can inherit access rights of their subordinates. Thus, all these schemes [3], [4], [5], [6] are not able to reflect efficiently organization’s policies and structures [8].

Role-Based Encryption (RBE) [7], [9], [10] is an emerging cryptographic technique, which combines both properties of the traditional Role-Based Access Control (RBAC) [11] 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, 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 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 RBAC, in RBE, roles can inherit access permissions from other roles [7]. 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 [12], [13], as it can reflect closely a real-world organization’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 [7]. For better illustration, let us consider a sample role hierarchy, shown in Fig. 1, of an organization/enterprise based on the designation of the employees. The CEO of the organization supervises many employees such as directors, project leaders, engineers, etc. Similarly, the directors manage project leaders, engineers, and so on. In such scenarios, the higher-level authorities inherit the privileges of its lower-level authorities. As such, in an organization scenario, RBE is the more suitable cryptographic technique (due to its inheritance property) for designing a keyword search mechanism compared with other cryptographic techniques such as the ABE.

RBE has been used to provide access control in cloud environments over encrypted data [7], [8], [10]. 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, the owner of data might wish to have access policies defined over roles across multiple organizations and domains. For example, a data owner might want to share medical records only with the users who have the role of a doctor issued by a hospital and the role of a clinical researcher issued by a medical laboratory. The data owner can specify a RBAC access policy in such a way that only the users having the access privileges for the roles “Doctor” and “Clinical Researcher” can gain access to the content corresponding to the encrypted data. Similar instances can also happen in enterprises. For example, it may happen that a financial budget document in an organization can only be accessed by a finance officer in the enterprise and a tax officer from an auditing organization. The data owner can choose an access policy that can allow any user having privileges for the roles “Finance Officer” and “Tax Officer” and “Auditing Officer” to access to the financial document. Other examples include the formation of consortiums and collaborations among multiple organizations, where users can hold different roles in the participating organizations. Hence there is a need for designing effective keyword search schemes over encrypted data in the cloud with data coming from multiple organizations, as well as a controlling access to data, both of which are important and topical research problems which need to be addressed.

1.1 Contributions
Our main contribution in this paper is that we propose an authorized keyword search scheme over encrypted data for multi-organization settings using a novel Role-Based Encryption (RBE) technique. That is, our scheme enables keyword search over encrypted data, which is an authorized search in the sense that only users having authorized roles are allowed to perform keyword search on encrypted data stored in the cloud coming from data owners in different organizations. Furthermore, it allows the authorized users the ability to decrypt. In addition, we have presented an efficient user revocation and conjunctive keyword search mechanisms for the proposed scheme.

Our scheme fills the research gaps in the areas of searchable encryption and in RBE. The existing RBE schemes only support single organization environment, whereas our scheme enables the users to be part of more than one organization and to hold roles from these organizations simultaneously (such as in a consortium or collaborative projects).

The main technical results in this paper are as follows:

1) We develop a novel RBE scheme that enables the organizations to outsource their encrypted data in an untrusted public cloud. The RBE scheme is designed in such a way that any user possessing roles associated with the defined RBAC access policy or any higher-level roles in the hierarchy (due to the inheritance property) can delegate keyword search capabilities over encrypted data to the public cloud along with the ability to decrypt. Most importantly, our designed RBE scheme enables the data owners to define and enforce RBAC access policies in the
encrypted data itself, thereby data owners retain full control over their outsourced data.

2) Our scheme also addresses the data access control issue in a multi-organization cloud environment in RBE settings, and is the first doing so, to the best of our knowledge. Our scheme enables the users to hold roles from more than one organization. As such, a data owner can define RBAC access policy with the roles from more than one organization and can enforce it in the encrypted data itself so that any user holding the qualified roles can get access.

3) We introduce a conjunctive keyword search mechanism with minimal overhead in the system along with a privilege revocation technique.

4) We have provided a formal security analysis of our scheme to demonstrate its security against the Chosen Plaintext Attacks and the Chosen Keyword Attacks.

5) Moreover, our scheme presents a comprehensive performance analysis, which shows that it 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 WORK

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 [15]. Later, several searchable encryption schemes have been proposed, for various functionalities and security requirements such as conjunctive (or multi) keyword search [6], [16], Rank Keyword Search [17], verifiable keyword search [18], [19], traceable keyword search [20], forward and backward security [21], [22], keyword guessing attack [23], multi-authority [24], and so on. Broadly, all the keyword search schemes can be divided into two categories, namely, Symmetric Searchable Encryption (SSE) and Searchable Public Key Encryption (SPKE). The core difference between these two categories is that SSE is based on symmetric key cryptography, while SPKE is based on public key cryptography. Some of the notable SSE schemes are [21], [22], [25], [26], [27], [28], [29], [30], [31]; while [3], [6], [18], [24], [32], [33], [34], [35], [36] are SPKE.

In [25], Curtmola et al. proposed a scheme for multi-user settings, which can perform single keyword search. In [26], Kamara et al. proposed a dynamic version of the scheme [25] that can add and delete files at any time efficiently. However, the scheme [26] leaks significant information while performing update operation [29]. To reduce the information leakage, in [21], Stefanov et al. first introduced the forward secrecy and in [27], Bost et al. introduced backward secrecy in SSE. Later on, several works such as [22], [28], [31], have been proposed in the direction of forward and backward secrecy in SSE for achieving better security and efficiency. In [30], Liu et al. proposed a keyword search scheme which enables the users to verify the search results against the dishonest servers. Although the SSE schemes provide better efficiency in terms of computation cost, SPKE schemes provide more flexible and expressive search queries [6].

Recently, several SPKE schemes have been proposed using Attribute-Based Encryption (ABE) technique [3], [6], [24], [34], [35], [36], 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 [6], Sun et al. proposed keyword search schemes for both single and conjunctive keyword search, which does not introduce any additional overhead in the system. In [3], Hu et al. proposed another ABE based keyword 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 [34], Miao et al. proposed an ABE based keyword search scheme for hierarchical data, which also supports conjunctive keyword search. In [35], 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. In [36], Zeng et al. addressed the forward security issue for ABE settings using the concept of 0-Encoding and 1-Encoding. Recently, in [24] Miao et al. proposed a multi-authority ABE based keyword search scheme. Unlike the previous ABE based keyword search schemes such as [3], [6], [34], [35], [36], in [24] a user can hold attributes from more than one Attribute Authorities (AA).

All the aforementioned schemes do not support role hierarchy and inheritance properties.

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, 4. In conjunctive keyword search, a user can search for multiple keywords in a single request [14].

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

6. The motivation for studying forward secrecy came from file injection attacks on SSE [37].
possessing the required role(s) satisfying the associated RBAC access policy is allowed to decrypt the data. Some notable works in this area are [7], [8], [10], [38], [39], [40], [41], [42], where [38], [39], [40], [41], [42], [43], [44] are based on Hierarchical Key Assignment (HKA) method and [7], [8], [10] are based on RBE method.

Access control using HKA method has been studied in the early 1980s. In [38], 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., [39], [40], [41], [42], [43], [44]. However, the main drawback of the HKA schemes is the high complexity in setting up the encryption keys for a large set of users [7]. 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 be updated per user revocation, which may incur a high overhead on the system.

In [7], Zhou et al. proposed the first RBE scheme for data sharing in an untrusted hybrid cloud environment. In [7], the ciphertexts and secret keys of the users are constant in size. This scheme also offers user revocation capability. In [10], Zhu et al. proposed another RBE scheme. In this scheme, the ciphertext size linearly increases with the number of roles. Recently, in [8], 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.

Moreover, none of [7], [8], [10] address keyword search functionality. Also these schemes do not support multi-organization cloud storage systems, where a user can possess roles from more than one organization.

3 PROPOSED MODEL

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

3.1 System Model

Fig. 2 shows the proposed system model, where the dotted 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 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:

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 not 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) **Backward and Forward Secrecy [45]:** Backward Secrecy and Forward Secrecy are two important requirements in revocable access control systems. Backward secrecy and forward secrecy mean that a user should not be able to decrypt the previous and future ciphertexts, respectively, if his/her roles required in the decryption are revoked.

4) **Resistance against Replay Attacks:** If one or more valid trapdoors are exposed to an adversary, the adversary should not be able to launch replay attacks. Many recent keyword search schemes, e.g., [3], [6] 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.

### 3.3 Framework

Broadly the proposed scheme is divided into nine main phases, namely, **System Setup, Management of Roles, Public Cloud Key Generation, New User Enrollment, 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 System Setup 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 Key Generation** phase using the **PUBCLOUDKEYGEN** algorithm. In the **New User Enrollment** 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 on the encrypted data using the received trapdoor. In the **PARTIALDEC,** the public cloud 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 1.

| Notation | Description |
|----------|-------------|
| $q$      | a large prime number |
| $G_1, G_T$ | two cyclic multiplicative groups of order $q$ |
| $H_1(\cdot), H_2(\cdot)$ | hash functions $H_1: \{0,1\}^{*} \rightarrow \mathbb{Z}_q^*$ and $H_2: G_1 \rightarrow \mathbb{Z}_q^*$ |
| $\Phi$ | set of system authorities in the system |
| $m_\Phi$ | total number of system authorities in the system |
| $\Psi_k$ | set of roles associated with a role hierarchy of the $k^{th}$ system authority |
| $\Gamma$ | set of all roles associated with a ciphertext |
| $\Gamma^\Phi$ | system authorities associated with a ciphertext |
| $S_{ID_{a}}$ | set of roles associated with the user $ID_{a}$ |
| $R_{k,i}$ | $i^{th}$ role managed by $k^{th}$ system authority |
| $R_{k,i}$ | the set of ancestor roles of $R_{k,i}$ |
| $ID_{a}$ | unique identity of the $u^{th}$ user |
| $ID_{c}$ | unique identity of the public cloud |
| $RK_{u,k,i}$ | role manager which manages role $r_{k,i}$ |
| $t_{s}$ | current timestamp |

| Notation | Description |
|----------|-------------|
| $q$      | a large prime number |
| $G_1, G_T$ | two cyclic multiplicative groups of order $q$ |
| $H_1(\cdot), H_2(\cdot)$ | hash functions $H_1: \{0,1\}^{*} \rightarrow \mathbb{Z}_q^*$ and $H_2: G_1 \rightarrow \mathbb{Z}_q^*$ |
| $\Phi$ | set of system authorities in the system |
| $m_\Phi$ | total number of system authorities in the system |
| $\Psi_k$ | set of roles associated with a role hierarchy of the $k^{th}$ system authority |
| $\Gamma$ | set of all roles associated with a ciphertext |
| $\Gamma^\Phi$ | system authorities associated with a ciphertext |
| $S_{ID_{a}}$ | set of roles associated with the user $ID_{a}$ |
| $R_{k,i}$ | $i^{th}$ role managed by $k^{th}$ system authority |
| $R_{k,i}$ | the set of ancestor roles of $R_{k,i}$ |
| $ID_{a}$ | unique identity of the $u^{th}$ user |
| $ID_{c}$ | unique identity of the public cloud |
| $RK_{u,k,i}$ | role manager which manages role $r_{k,i}$ |
| $t_{s}$ | current timestamp |

3.4 Adversary and Security Model

3.4.1 Adversary 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.

3.4.2 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.

3.4.2.1 Semantic Security Against Chosen Plaintext Attack.

The semantic security of the proposed scheme defined on Chosen Plaintext Attack (CPA) security under Selective-ID Model. The CPA security can be illustrated using the

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).
following security game IND-CPA between a challenger $C$ and an adversary $A_1$.

**INIT** Adversary $A_1$ sends a challenged role set $\Gamma^*$, a keyword $w$, and two identities $ID^*_u, ID^*_v$ to the challenger $C$.

**Setup** Challenger runs the SystemSetup algorithm to generate public parameters and master secrets. Challenger $C$ 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 $C$ sends the public parameter, role public keys, proxy re-encryption keys, public and private keys to the adversary $A_1$. It keeps the master secret and role secrets in a secure place.

**Phase 1** Adversary $A_1$ submits a role set $S^*$ to the challenger $C$ for role-keys so that there exists at least one role $r'_k \in S^*$ such that $r'_k \notin R, k$, where $r'_k \in \Gamma^*$. Challenger $C$ runs the UserRoleKeyGen algorithm to generate role-keys for the adversary $A_1$. Adversary $A_1$ can send queries for the role-keys to the challenger $C$ by polynomially many times.

**Challenge** When adversary $A_1$ decides that **Phase 1** is over, it submits two equal length messages $k_0$ and $k_1$, which were not challenged before, to the challenger $C$. Challenger $C$ flips a binary coin $\omega$ and encrypts message $k_w$ using the Enc algorithm for the challenged role set $\Gamma^*$. Challenger $C$ sends the encrypted ciphertext of $w_0$ to the adversary $A_2$.

**Phase 2** Same as **Phase 1**.

**Guess** Adversary $A_2$ outputs a guess $\omega'$ of $\omega$. The advantage of winning this game for adversary $A_2$ is $Adv^{IND-CPA}_{A_2} = |Pr[\omega' = \omega] - \frac{1}{2}|$.

**Definition 3.1.** The proposed scheme is secure against chosen plaintext attack if $Adv^{IND-CPA}_{A_1}$ is negligible for any polynomial time adversary $A_1$.

3.4.2.2 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 3.4.2.1. The CKA security can be demonstrated using the following security game IND-CKA between a challenger $C$ and an adversary $A_2$.

**INIT** Adversary $A_2$ sends a set of challenged roles $\Gamma^*$ and two identities $ID^*_u, ID^*_v$ to the challenger $C$.

**Setup** Challenger runs the SystemSetup algorithm to generate public parameters and master secrets. Challenger $C$ 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 $C$ sends the public parameter, role public keys, proxy re-encryption keys, public and private keys to the adversary $A_2$. It keeps the master secret and role secrets in a secure place.

**Phase 1** Adversary $A_2$ submits a set of roles $S^*$ and a keyword $w$ to the challenger $C$ so that there exists at least one role $r'_k \in S^*$ such that $r'_k \notin R, k$, where $r'_k \in \Gamma^*$. Challenger initiates the TrapGen algorithm to generate a trapdoor for the adversary $A_2$. Finally, challenger $C$ sends the generated trapdoor to the adversary $A_2$. Afterwards, adversary $A_2$ can send queries to the trapdoor of the challenger $C$ by polynomially many times.

**Challenge** When adversary $A_2$ 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 $C$. Challenger $C$ flips a binary coin $\omega$ and encrypts keyword $w_\omega$ using the Enc algorithm for the challenged role set $\Gamma^*$. Challenger $C$ sends the encrypted ciphertext of $w_\omega$ to the adversary $A_2$.

**Phase 2** Same as **Phase 1**.

**Guess** Adversary $A_2$ outputs a guess $\omega'$ of $\omega$. The advantage of winning this game for adversary $A_2$ is $Adv^{IND-CKA}_{A_2} = |Pr[\omega' = \omega] - \frac{1}{2}|$.

**Definition 3.2.** The proposed scheme is secure against the chosen keyword attack if $Adv^{IND-CKA}_{A_2}$ is negligible for any polynomial time adversary $A_2$.

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. Fig. 3 shows two sample role hierarchies, namely Role Hierarchy 1 (Fig. 3a) and Role Hierarchy 2 (Fig. 3b). We consider Role Hierarchy 1 (Fig. 3a) as an example to define the following notations of a role hierarchy.

- $r'_k$: root role of a role hierarchy. We assume that in any role hierarchy there can be only one root role.
- $\Psi_r$: set of all roles in the role hierarchy. For example, $\Psi_r = \{r'_1, r'_2, r'_3, r'_4, r'_5, r'_6, r'_7\}$.
- $\Psi_a$: ancestor set of the role $r'_k$. For example, $\Psi_a = \{r'_1, r'_2, r'_3, r'_4, r'_5\}$ and $\Psi_{a'} = \{r'_6, r'_7, r'_1, r'_2, r'_3\}$.

4.2 Bilinear Pairing

Let $G_1$ and $G_T$ be two cyclic multiplicative groups of order $g$. Let $g$ be a generator of $G_1$. The bilinear map $\hat{e} : G_1 \times G_1 \rightarrow G_T$ has the following properties:

- *Bilinear*: $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}, \forall g \in G_1$ and $\forall (a, b) \in \mathbb{Z}_q^*$
In our scheme, the root role, i.e., Non-degenerate $k_2 G = a$ selects a random $g^x$ and $g^y$ in the group computes a common key $CK = (x_{i-1})^{x_i} \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdot \ldots \cdot X_{i-2} = g_1 a_2 a_3 + \cdots + a_{n-1} a_1$ without knowing others’ secrets and without disclosing the common key to any other unintended entities. More details can be found in [46].

4.3 Group Key Distribution

In [46], Burmester et al. proposed a two round group key distribution scheme using the concept of Diffie-Hellman assumption. Their scheme works as follows:

Let $U = \{I_{D_1}, I_{D_2}, \ldots, I_{D_n}\}$ be the group of $n$ users. Suppose the users are arranged into a cycle. To compute a group key among the users, each user $I_{D_i} \in U$ 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 $G_1$. Afterwards, it publishes $X_i = (y_i g^{a_2})^{\frac{a_i}{a_2}}$. Finally, each user $I_{D_i}$ in the group computes a common key $CK = (x_{i-1})^{x_i} \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdot \ldots \cdot X_{i-2} = g_1 a_2 a_3 + \cdots + a_{n-1} a_1$ without knowing others’ secrets and without disclosing the common key to any other unintended entities. More details can be found in [46].

4.4 Decisional Bilinear Diffie-Hellman (DBDH)

Let $G_1$ and $G_2$ be two cyclic multiplicative groups of order $q$. Let $g$ be a generator of $G_1$ and $\hat{\epsilon} : G_1 \times G_1 \rightarrow G_2$ 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 $< g, g^x, g^y, g^z, Z = \hat{\epsilon}(g, g)^{xy} >$ and $< g, g^a, g^b, g^c, Z = \hat{\epsilon}(g, g)^{abc} >$ 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 Fig. 4, a sample RKH is shown. Each node in a RKH represents a role, and each role (except the root role), say $r_k^i$, is associated with a role public key, say $PK_{r_k^i}$. 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. The role-keys are generated in such a way that the user can use them to compute trapdoors 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 $r_k^i$ to have access to data. The owner encrypts the data and the associated keywords using the role public key $PK_{r_k^i}$. Any user who possesses any one of the roles in $R_h = \{r_{k_1}^1, r_{k_2}^1, r_{k_3}^1, r_{k_4}^1\}$ can search and decrypt the encrypted data using their respective role-keys. That is, the user possesses a qualified role for accessing the ciphertext.

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 organization. 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.

1) \textbf{SystemSetup} \( (\{PP, \{MS_k\}_{k \in \Phi} \leftarrow 1^\lambda) \): It chooses two cyclic multiplicative bilinear groups $G_1$ and $G_2$ of order $q$, where $q$ is a large prime number. It also
chooses a generator \( g \in G_1 \), random numbers \( \{\eta_k, \mu_k, x_k\}_{k \in \Phi} \in Z_q^* \) and two hash functions \( H_1 : \{0,1\}^* \rightarrow Z_q^* \), \( H_2 : G_1 \rightarrow Z_q^* \). 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 \( y = y_1 \cdot y_2 \cdot y_3 \cdot \ldots \cdot y_m \cdot y_1 \) and \( m \) is the total number of system authorities. Afterward, it computes \( Y = \hat{c}(g, g)^y \) and \( h_1^k = g^{a_1} \), and then publishes the system public parameter \( PP = < G_1, G_2, g, \hat{c}, H_1, H_2, Y, \{h_1^k\}_{k \in \Phi}> \). Each system authority, say \( k^{th} \) system authority \( SA_k \), keeps master secret \( MS_k = < g^y, \eta_k, \mu_k, x_k > \) in a secure place.

**Remark 1.** All the system authorities can check validity of \( Y \) by comparing \( \hat{c}(g, g) = Y \). 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 \( SA_k \) wants to initialize a role hierarchy \( H \). The system authority \( SA_k \) generates role secrets \( RS_{r_k} \) and role public keys \( PK_{r_k} \) for each role \( r_k \) associated with \( H \). It also computes proxy re-encryption keys \( \{\text{PKey}_{r_k}^{P_k}\}_{r_k \in \mathcal{R}_k} \) for each role \( r_k \) (except the root role) associated with the role hierarchy \( H \). 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 \( r_k \), i.e., \( RS_{r_k} \) is shared with the role-manager which manages \( r_k \), i.e., \( RM_{r_k} \). 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.

1) **ManageRole** \( ((\text{RP}_k, \{\text{PK}_k\}_{r_k}, \{\text{RS}_k\}_{r_k}, \{\text{PKey}_k^{P_k}\}_{r_k} \in \mathcal{R}_k}, \{\text{RS}_{r_k}\}_{r_k} \in \mathcal{R}_k}, \{\text{PK}_{r_k}\}_{r_k} \in \mathcal{R}_k}, \{\text{PKey}_{r_k}^{P_k}\}_{r_k} \in \mathcal{R}_k}) \rightarrow (H, PP) \): It selects random numbers \( \{t_k\}_{r_k} \in \mathcal{R}_k \) and computes role secrets \( RS_{r_k} \), role public key \( PK_{r_k} = < \text{PK}_{r_k}, t_k; \mathcal{R}_k > \) and proxy re-encryption key \( \text{PKey}_{r_k}^{P_k} \) for each role \( r_k \in (\mathcal{R}_k \setminus \{r_k\}) \), where

\[
\text{RS}_{r_k} = \prod_{r_j \in \mathcal{R}_k \setminus \{r_k\}} t_j k_{r_j}
\]

\[
\text{PK}_{r_k}^{P_k} = g^{t_k k_{r_k}}
\]

\[
\text{PKey}_{r_k}^{P_k} = \prod_{r_j \in \mathcal{R}_k \setminus \{r_k\}} t_j k_{r_j}
\]

\( \mathcal{R}_k \) is the set of ancestor roles of \( r_k \) and role secret parameter \( \text{RP}_k = < \{t_k\}_{r_k} > \). 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 \( t_k \) and role secret \( RS_{r_k} \) are shared with the role-manager \( RM_{r_k} \). 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 \( SA_k \) wants to issue keys for the public cloud. It computes a private key \( \text{Priv}^k_x \) and two public keys \( (\text{Pub}^k_x, \text{Pub}^k_{y}) \) and sends the private key \( \text{Priv}^k_x \) to the public cloud using a secure-channel. It stores both the public keys \( (\text{Pub}^k_x, \text{Pub}^k_{y}) \) in its public bulletin board. This phase consists of the **PubCloudKeyGen** algorithm which is defined next.

1) **PubCloudKeyGen** \( ((\text{Priv}^k_x, \text{Pub}^k_x, \text{Pub}^k_{y}) \rightarrow (PP, MS_{k}, ID_{c})) \): It computes a private key \( \text{Priv}^k_x \) and two public keys \( (\text{Pub}^k_x, \text{Pub}^k_{y}) \) for the public cloud as follows:

\[
\text{Priv}^k_x = H_2(g^{H(ID_{c})/x_k}) = H_2(g^{H(ID_{c})/x_k})
\]

\[
\text{Pub}^k_x = g^{\text{Priv}_x^k} \text{Pub}^k_x
\]

\[
\text{Pub}^k_{y} = g^{\text{Priv}_y^k} \text{Pub}^k_{y}
\]

### 5.2.4 New User Enrollment

A system authority initiates this phase when a new legitimate user, say \( ID_{u} \), wants to join an organization, say \( k^{th} \) organization. The system authority \( SA_k \) generates a secret key \( SK^k_{ID_{c}} \) and public key \( \text{Pub}^k_{ID_{c}} \) for the user \( ID_{c} \). It also generates a user secret \( US_{ID_{c}} \) which is shared with all the role-managers under its control. \( SA_k \) sends the secret key \( SK^k_{ID_{c}} \) to the user \( ID_{c} \) using a secure-channel and keeps the public key \( \text{Pub}^k_{ID_{c}} \) in its public bulletin board. This phase comprises the **UserPrivKeyGen** algorithm which is defined next.

1) **UserPrivKeyGen** \( ((SK^k_{ID_{c}}, \text{Pub}^k_{ID_{c}}, US_{ID_{c}}) \rightarrow (MS_{k}, PP, ID_{c})) \): It issues a pair of secret key \( SK^k_{ID_{c}} \), public key \( \text{Pub}^k_{ID_{c}} \), and a user secret \( US_{ID_{c}} \) as follows:

\[
\text{Priv}_{ID_{c}} = H_2(g^{H(ID_{c})/x_k}) = H_2(g^{H(ID_{c})/x_k})
\]

\[
\text{Priv}^k_{ID_{c}} = (g^{\text{Priv}_{ID_{c}}^k})^{-x_k} \text{Priv}_{ID_{c}} = (g^{\text{Priv}_{ID_{c}}^k})^{-x_k}
\]

\[
\text{Pub}^k_{ID_{c}} = g^{\text{Priv}_{ID_{c}}^k} \text{Pub}^k_{ID_{c}}
\]

\[
US_{ID_{c}} = (g^y)^{\text{Priv}_{ID_{c}}} \cdot g^{h_k} = (g^y)^{\text{Priv}_{ID_{c}}} \cdot g^{h_k}
\]

Note that all the system authorities compute the same private key \( \text{Priv}_{ID_{c}} \) for the user \( ID_{c} \). Hence, the user \( ID_{c} \) 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 \( RM_{r} \) wants to assign a role \( r^k \) to the user \( ID_{c} \). To do so, \( RM_{r} \) computes two role-keys \( (\text{RK}_{1}^{r^k}, \text{RK}_{2}^{r^k}) \) for the user \( ID_{c} \) and sends the role-keys to the user \( ID_{c} \) using a secure-channel. This phase comprises the **UserRoleKeyGen** algorithm which is described next.
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 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:

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

The owner chooses random numbers $\{(d_1^i, d_2^i)_{\forall \gamma \in \Gamma}\} \in \mathbb{Z}_p^2$ where $d_1 = \sum_{\gamma \in \Gamma} d_1^i$, $d_2 = \sum_{\gamma \in \Gamma} d_2^i$ and $d = d_1 + d_2$. The owner also computes $\{d_k = \sum_{\gamma \in \Gamma} d_k^i \}_{\forall \gamma \in \Gamma}$. Finally, the owner generates a ciphertext $\text{CERT} = < \text{Enc}_k(m), C_1, C_2, C_3, \{C_{4k} \}_{\forall \gamma \in \Gamma}, \{C_{4k}^* \}_{\forall \gamma \in \Gamma}, \Gamma, \Gamma_{\Phi} >$ for the plaintext message $m$, where

\[ C_1 = k \cdot Y^d = k \cdot e(g, g)^{ed} \]
\[ C_2 = (k_1^d)_{\gamma \in \Gamma} = g^{d_1 \gamma d_2} \]
\[ C_3 = (\text{Pub}_{\text{c}}^2)^d = g^{d_{\gamma \gamma} \text{Priv}_{\text{c}} d_{\gamma \gamma}} \]
\[ C_{4k} = (\text{Pub}_{\text{c}})^d = g^{d_{\gamma \gamma} \text{Priv}_{\text{c}} d_{\gamma \gamma}} \]
\[ C_{4k}^* = (\text{Pub}_{\text{c}})^d \left( H_1(w) \right)^{d_{\gamma \gamma}} = g^{d_{\gamma \gamma} \text{Priv}_{\text{c}} d_{\gamma \gamma}} \]
\[ C_{4k}^* = (\text{Pub}_{\text{c}})^d \left( H_1(w) \right)^{d_{\gamma \gamma}} = g^{d_{\gamma \gamma} \text{Priv}_{\text{c}} d_{\gamma \gamma}} \]

Note that the data owner embeds the hashed value of the keyword, i.e., $H_1(w)$ for some roles in $\Gamma$ 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 $\text{Trap}$ 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 $\text{Trap}$ along with the associated roles to the public cloud using a secure-channel. This phase comprises the TrapGen algorithm which is described next.

1) $\text{TrapGen} \quad ((\text{Trap}, v) \leftarrow (\{\text{Priv}_{\text{c}}^k\}_{\forall \gamma \in \Gamma}, \text{Trap}, \text{Pub}_{\text{c}}^k, \text{ID}_u, ts'))$: 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 ⊥. 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 $U'$, $V_1'$, $V_2'$ and $V_3'$, based on its known $(\{Priv^\psi_k\}_{\psi \in \Gamma})$ keys, where

$$U' = \prod_{\psi \in \Gamma} \left( C_{\psi k} \right)^{Priv^\psi_k}$$

$$= \prod_{\psi \in \Gamma} \left( g^{\mu_k Priv^\psi_k d'_k} \right)^{Priv^\psi_k}$$

$$= g^{\sum_{\psi \in \Gamma} \mu_k d'_k}$$

$$V_1' = \hat{\epsilon} \left( \left( \text{Pub}_{1D_0}^{\psi_1}, U' \right) \right)$$

$$= \hat{\epsilon} \left( \frac{H_2(Priv_{1D_0}^{\psi_1})}{g^{\mu_k Priv_{1D_0}^{\psi_1} d'_k}}, \prod_{\psi \in \Gamma} g^{\sum_{\psi \in \Gamma} \mu_k d'_k} \right)$$

$$= \hat{\epsilon} \left( g^r, \prod_{\psi \in \Gamma} g^{\sum_{\psi \in \Gamma} \mu_k d'_k} \right)$$

$$= \hat{\epsilon} \left( g^y, \sum_{\psi \in \Gamma} \mu_k d'_k \right)$$

$$V_2' = \hat{\epsilon} \left( (tr_1)^{ts'}, U' \right)$$

$$= \hat{\epsilon} \left( g^{\psi_1}, \sum_{\psi \in \Gamma} g^{\psi d'_k} \right)$$

$$= \hat{\epsilon} \left( g^y, \sum_{\psi \in \Gamma} g^{\psi d'_k} \right)$$

$$V_3' = \hat{\epsilon} \left( (tr_1)^{ts'}, U' \right)$$

$$= \hat{\epsilon} \left( g^y, \sum_{\psi \in \Gamma} g^{\psi d'_k} \right)$$

Now, the public cloud checks whether $V_1' \neq V_2'$. If the equation holds, the public cloud performs the operations defined in the KeySearch algorithm. Otherwise, it aborts the connection.

Proof of Consistency.

$$V_1' = \hat{\epsilon} \left( g^y, \sum_{\psi \in \Gamma} \mu_k d'_k \right)$$

$$= \hat{\epsilon} \left( g^y, \sum_{\psi \in \Gamma} \mu_k d'_k \right)$$

$$= \hat{\epsilon} \left( g^y, \sum_{\psi \in \Gamma} \mu_k d'_k \right)$$

2) KeySearch ($(\text{CT}/\bot) \rightarrow (\text{CT}, \text{Trap}, V_3')$): Suppose the user $1D_0$ possesses a role set $S_{1D_0}$ and wants to access the $k$th organization’s data. Suppose $\text{CT} = < \text{Enc}_0(M), C_1, C_2, C_3, \text{C}_{\text{trap}}, \{C_{\psi k}\}_{\psi \in \Gamma} \} \in \Gamma, \Gamma \_ \Gamma \_ \Gamma >$ is the ciphertext of the $k$th organization on which the public cloud wants to perform the keyword search operation, where for all $r^k_i \in \Gamma$, there is at least one $r^k_i \in S_{1D_0}$ such that $r^k_i \in \mathbb{R}_k$.

The public cloud computes $V_4'$ and $V_2$. While computing $V_4'$, two cases are considered which are as follows:

Case 1: if $r^k_2 = r^k_3$, then

$$V_4' = \hat{\epsilon} \left( \left( tr_1^{r^k_2}, C_{\psi 2} \right) \right)$$

$$= \hat{\epsilon} \left( g^{\frac{y \cdot \text{Priv}_{1D_0}^{\psi_2} + \mu_k}{H_1(w)_r}, \prod_{\psi \in \Gamma} g^{\psi d'_k}, g^{H_1(w)_r} \prod_{\psi \in \Gamma} g^{\psi d'_k} \right)$$

$$= \hat{\epsilon} \left( g^y, \prod_{\psi \in \Gamma} g^{\psi d'_k} \right)$$

Otherwise, Case 2: if $r^k_2 \in \{\mathbb{R}_k \setminus \{r^k_3\}\}$ (let $\gamma = [g \cdot \text{Priv}_{1D_0}^{\psi_2} + \mu_k]$)

$$V_4' = \hat{\epsilon} \left( \left( tr_1^{r^k_2}, C_{\psi 2} \right) \right)$$

$$= \hat{\epsilon} \left( g^{\gamma \cdot \text{Priv}_{1D_0}^{\psi_2} + \mu_k}, \prod_{\psi \in \Gamma} g^{\psi d'_k}, g^{\gamma \cdot \text{Priv}_{1D_0}^{\psi_2} + \mu_k}, \prod_{\psi \in \Gamma} g^{\psi d'_k} \right)$$

$$= \hat{\epsilon} \left( g^y, \prod_{\psi \in \Gamma} g^{\psi d'_k} \right)$$

$$= \hat{\epsilon} \left( g^y, \prod_{\psi \in \Gamma} g^{\psi d'_k} \right)$$

Now, the public cloud computes $V_3'$, where

$$V_3' = \frac{V_2 \cdot V_4'}{V_1'}$$

$$= \hat{\epsilon} \left( g^y, \prod_{\psi \in \Gamma} g^{\psi d'_k} \right)$$

Note that $\hat{\epsilon} \left( g^y, \prod_{\psi \in \Gamma} g^{\psi d'_k} \right) = \hat{\epsilon} \left( g^y, \prod_{\psi \in \Gamma} g^{\psi d'_k} \right)$ (Please refer to Section 5.2.6).

Afterward, the public cloud computes $V_3$, $V_5$ and $V_6$, where
Finally, the public cloud compares $V_2$ and $V_8$. If both are equal then it performs the PartialDec algorithm, which is described next. Otherwise, it aborts all the operations and outputs $\bot$, which means that the ciphertext does not have the desired keyword.

3) \textbf{PartialDec} $(CT' \leftarrow (CT, \text{Trap}, \{\text{Priv}_k\}_{ID_k \in F_k}, \{s_{ID_k}\}))$: In this algorithm, the public cloud partially decrypts all the ciphertexts returned by the KeySearch algorithm. Suppose ciphertext $CT = < \text{Enc}_{k}(M), C_1, C_2, C_3, \{C_{dk}, C_{dk}'\}_{ID_k \in F_k}, \{C_{dk}, C_{dk}'\}_{ID_k \in F_k}, \Gamma, \Gamma_F >$ has a matching keyword with the trapdoor Trap. To partially decrypt the ciphertext CT, the public cloud first computes $V_{T_1}$ and $V_2$. Similar to $V_{T_1}$, the computation procedure considers the two following cases to compute $V_{T_2}$:

Case 1: if $r_2^k = r_2^{k'}$, then

$$V_{T_2} = \hat{e}(tr_2^{k}, C_2) = \hat{e}\left( g^{\sum_{j \in R_{k}}} Y_{j}^{l_{a}} \prod_{j \in R_{k}} l_{j}^{l_{a}}, \frac{g^{\sum_{j \in R_{k}}} Y_{j}^{l_{a}} \prod_{j \in R_{k}} l_{j}^{l_{a}}}}{g^{\sum_{j \in R_{k}}} Y_{j}^{l_{a}} \prod_{j \in R_{k}} l_{j}^{l_{a}}} \right)$$

$$= \hat{e}(g, g)^{\mu_{ID_k} + \mu_{ID_k'}}^{v_2}$$

Otherwise, Case 2: if $r_2^k \in (R_{k} \setminus \{r_2^{k'}\})$ (let $\gamma = [y \cdot \text{Priv}_{ID_k} + \mu_{ID_k}]$)

$$V_{T_2} = \hat{e}\left( tr_2^{k}, C_2 \right) = \hat{e}\left( g^{\sum_{j \in R_{k}} Y_{j}^{l_{a}} \prod_{j \in R_{k}} l_{j}^{l_{a}}}, g^{\sum_{j \in R_{k}} Y_{j}^{l_{a}} \prod_{j \in R_{k}} l_{j}^{l_{a}}} \right)$$

$$= \hat{e}(g, g)^{\mu_{ID_k} + \mu_{ID_k'}}^{v_2}$$

The public cloud knows its private key $\{\text{Priv}_k\}_{ID_k \in F_k}$ computes $U$ and $V_8$, as follows:

$$U = \prod_{ID_k \in F_k} (C_{dk})^{\text{priv}_{k}}$$

$$= \prod_{ID_k \in F_k} (g^{\mu_{ID_k} + \mu_{ID_k'}})^{\text{priv}_{k}} = g^{\sum_{ID_k \in F_k} \mu_{ID_k} v_{ID_k}}$$

$$V_8 = \hat{e}(tr_1, U)$$

$$= \hat{e}(g, g)^{\sum_{ID_k \in F_k} \mu_{ID_k} v_{ID_k}}$$

Now, the public cloud computes $V_9$, where

$$V_9 = \frac{V_8}{\hat{e}(g, g)^{\sum_{ID_k \in F_k} \mu_{ID_k} v_{ID_k}}}$$

$$= \hat{e}(g, g)^{\sum_{ID_k \in F_k} \mu_{ID_k} v_{ID_k}}$$

Note that $\hat{e}(g, g)^{\sum_{ID_k \in F_k} \mu_{ID_k} v_{ID_k}} = \hat{e}(g, g)^{\sum_{ID_k \in F_k} \mu_{ID_k} v_{ID_k}}$ (Please refer Section 5.2.6).

The public cloud computes $V_9$, where

$$V_10 = V_6 \cdot V_9$$

$$= \hat{e}(g, g)^{\mu_{ID_k} + \mu_{ID_k'}} \hat{e}(g, g)^{\sum_{ID_k \in F_k} \mu_{ID_k} v_{ID_k}}$$

Finally, the public cloud sends the partially decrypted ciphertext $CT' = < \text{Enc}_{k}(M), C_1, V_{10} >$ to the user $ID_k$.

5.2.9 Decryption

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

1) \textbf{FullDec}(M $\leftarrow (CT', \text{Priv}_{ID_k}, v)$): It computes $K$ from the ciphertext $CT'$ using his/her secret keys, $\text{Priv}_{ID_k}$ and $v$

$$K = \frac{C_1}{(V_{10})^{\text{priv}_{ID_k}}} = \frac{K \cdot \hat{e}(g, g)^{v}}{(\hat{e}(g, g)^{v})^{\text{priv}_{ID_k}}} = K \cdot \hat{e}(g, g)^{v}.$$ 

Finally, user $ID_k$ gets the actual plaintext data by decrypting $\text{Enc}_{k}(M)$ using $K$ 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 in the main construction defined in Section 5.2: the owner computes additional ciphertext components \( C^t_i \) for each keyword in \( \mathcal{W} \). A user computes trapdoor components \( t^{tr}_{i,k} = (R_i^k)^{t_{i,k}} \sum H_i(u_i) \) and \( t^{tr}_{i,j} = R_i^{t_{i,j}} \sum H_i(u_i) \). The public cloud computes \( V^1_{ir} = e(t^{tr}_{ir}, \prod C^t_i) \) (or \( V^1_{ir} = e((t^{tr}_{ir}) PKey^2, \prod C^t_i) \)). Similarly, the public cloud can also compute \( V^2_{ir} \). It can be observed that, our conjunctive keyword search mechanism does not introduce any significant 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 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 \( \text{Pub}^{ID_u} \) 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, the SA removes the public key \( \text{Pub}^{ID_u} \) of the revoked user \( ID_u \) 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 \( r_i \) from one or more users. To do that, the SA first chooses a fresh random number \( t_{i,k} \in \mathbb{Z}_q^* \) and updates all the parameters related with the revoked role \( r_i \). The SA computes updated public keys \( (PKey^2)^{t_{i,k}} \), role secrets \( \langle \text{RS}, t^*_j, \frac{r^*_j}{t^*_j} \rangle \) and proxy re-encryption keys \( \langle \text{PKey}, \frac{r^*_j}{t^*_j}, t^*_j \rangle \) related with the revoked role \( r_i \) (i.e., for all \( r_i \) such that \( r_i \in \mathbb{R}_{\mathcal{A}} \)), where \( t_{i,k} \) is the previously chosen random number associated with \( r_i \). The SA then sends the \( t_{i,k} \) to the public cloud for re-encryption of the stored ciphertexts associated with the revoked role \( r_i \). It also sends \( t_{i,k} \) to the corresponding role-managers for updating the role-keys associated with the revoked role \( r_i \).

The public cloud re-encrypts the ciphertext components \( (C^a_i)^{t_{i,k}} \) and \( (C^b_i)^{t_{i,k}} \) for all \( r_i \) such that \( r_i \in \mathbb{R}_{\mathcal{A}} \). 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) \( (R_i^k)^{t_{i,k}} \) for all the non-revoked users who possess \( r_i \) and ii) \( (R_i^k)^{t_{i,k}} \) for all the non-revoked users who possess \( r_i \), such that \( r_i \in \mathbb{R}_{\mathcal{A}} \).

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 2. If a probabilistic-polynomial time (PPT) adversary \( A_1 \) wins the CPA security game as defined in Section 3.4.2.1 with a non-negligible advantage \( \epsilon \), then a PPT simulator \( B \) can be constructed to break the DBDH assumption with non-negligible advantage \( \frac{\epsilon}{2} \).

Proof. In this proof, we show that a simulator \( B \) can be constructed to help an adversary \( A_1 \) to gain advantage \( \frac{\epsilon}{2} \) against our proposed scheme. More details are given in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCC.2021.3071304.

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 3. If a PPT adversary \( A_2 \) wins the CKA security game defined in Section 3.4.2.2 with a non-negligible advantage \( \epsilon \), then a PPT simulator \( B \) can be constructed to break DBDH assumption with non-negligible advantage \( \frac{\epsilon}{2} \).

Proof. In this proof, we show that a simulator \( B \) can be constructed to help an adversary \( A_2 \) to gain advantage \( \frac{\epsilon}{2} \) against our proposed scheme. More details are given in Appendix B, available in the online supplemental material.

6.1.3 Backward and Forward Secrecy
Theorem 4. Our proposed scheme guarantees forward and backward secrecy of the outsourced (encrypted) data against the revoked users.

Proof. This proof is presented in Appendix C, available in the online supplemental material.
Our proposed scheme is resistant against replay attacks.

**Theorem 5.** Our proposed scheme is resistant against replay attacks.

**Proof.** This proof is given in Appendix D, available in the online supplemental material.

### 6.2 Performance Analysis

This section evaluates functionality, computation, storage and communication overhead of our scheme. The computational overhead is shown in terms of number of pairing ($T_p$) and group exponentiation operations ($\text{Exp}_{G_1}$ and $\text{Exp}_{G_T}$).

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. The storage and communication overheads are shown in terms of group element size $|Z|^g$, $|G_1|$ and $|G_T|$. To evaluate the actual performance of our scheme, we have implemented our scheme in Charm$^a$ and used Enron Email Dataset$^{10}$ that contains about 0.5 million files from 150 users. We have used Python 3.5.2 and symmetric groups “SS512” for our implementations. We have performed the simulations in a virtual machine of Ubuntu 17.10 (64-bit) on a commodity laptop computer having 2.11 GHz Core i7 processor with 11GB memory. We set $|\Gamma_0| = |\Gamma|$ for our implementation. It is to be noted that all the implementation results are the mean of 20 trials. The notations used in the rest of this paper are shown in Table 2.

#### 6.2.1 Functionality Comparison

Table 3 shows the functionality comparison of some notable ABE based search schemes [3], [6], [24], [34], [35] and RBE based access control schemes such as [7], [8], [10] with our scheme. From Table 3, we can observe that [3], [6], [24], [34], [35] are based on ABE technique, while our scheme is based on RBE. As such, our scheme supports the role hierarchy property, which in turn makes it more suitable for the real world organizations/enterprises compared with [3], [6], [24], [34], [35]. On the other hand, unlike our scheme, the existing RBE based schemes such as [7], [8] and [10] do not support authorized keyword search functionality, as these schemes are designed to achieve access control over encrypted data. As such, [7], [8] and [10] also cannot address conjunctive keyword search. Unlike [3], [6], [34], [35], our scheme and Miao et al.’s scheme [24] support multi organization scenario. However, unlike our scheme, [24] requires a centralized authority (i.e., Certification Authority) to manage different authorities (commonly known as Attribute Authorities). Our scheme including [3], [6], [24], [34], [35] provide authorized keyword search functionality, as the owner can embed access policies of his/her choice on the encrypted data itself. However, our scheme and [6], [34] support conjunctive keyword search, while [3], [24], [35] do not support. Moreover, unlike our scheme, none of the keyword search schemes [3], [6], [24], [34], [35] verify (authenticate) the freshness of the trapdoors (search requests) before performing computationally expensive keyword search operations. Thanks to the authentication mechanism, unlike [3], [6], [34], [35], our scheme can prevent the replay attacks even if the trapdoors are exposed to the adversaries. In [3], [6], [34], [35], if an adversary gains access to a valid trapdoor, the adversary can re-use the trapdoor using a fresh random number. Our scheme supports both complete as well as role-level user revocations, while [6], [34], and [24] support attribute level revocation. Moreover, our scheme, [24], [34] support both the keyword search and decryption functionalities; while [3], [6], [35] support only the keyword search functionality. However, unlike our scheme and [24], [34] does not support outsourced decryption functionality.

#### 6.2.2 Computation Overhead Comparison

This section presents computation overhead comparison. A theoretical cost comparison of our scheme with [3], [6], [24], [35] and [34] is provided. [7], [8] and [10] are not considered as these schemes are not designed for keyword search. An experimental computation cost comparison based on simulations is also provided. Our scheme is compared with [6] and [34] in the experimental result analysis, as [6] and [34] are the only schemes that support both single and conjunctive keyword search operations, like ours. Table 4 shows the theoretical computation overhead comparison of our scheme with [3], [6], [24], [35] and [34]. The computation cost is shown in asymptotic upper bound in the worst cases. In Table 4, 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 $(4 + |\Gamma| + |\Gamma_0| + 2|\mathcal{W}|)$ group exponentiation operations on $G_1$ and one exponentiation operation on $G_T$. It can be observed that the encryption cost mainly depends on the number of roles and SAs associated with a ciphertext (i.e.,

---

**TABLE 2**

Notations

| Notation | Description |
|----------|-------------|
| $|\Gamma|$ | Total number of roles associated with a ciphertext |
| $|\Gamma_0|$ | Total number of attributes associated with a ciphertext |
| $|\Gamma_0|$ | Total number of SAs associated with $\Gamma$ (i.e., ciphertext) |
| $|S_{TD_0}|$ | Total number of roles associated with a trapdoor |
| $|S_{TD_0}'|$ | Number of attributes associated with a trapdoor |
| $|\mathcal{W}|$ | Number of keywords associated with a ciphertext |
| $|L|$ | Number of access levels or number of records [34] |
| $v$ | Values associated with each attributes [35] |
| $n_c$ | Total number of ciphertexts associated with a revoked role |
| $n_c'$ | Total number of ciphertexts associated with a revoked attribute |
| $n_u$ | Total number of users associated with a revoked role |
| $n_u'$ | Total number of users associated with a revoked attribute in |
| $n_s$ | Total number of SA associated with a user |
| $n_{u_s}$ | Total number of revoked attributes of a user |
| $|U_A|$ | Total number of attributes in the system |

---

9. www.charm-crypto.io/
10. www.cs.cmu.edu/enron/
associated with the chosen RBAC access policy). Similarly, the encryption cost of [3] depends on the attributes associated with an access policy. On the other hand, the encryption cost in [6], [24], [35] and [34] depends on the total number of attributes in the system. To achieve the same level of access control, we observe that our scheme is less costly for the encryption operation compared with [6], [24], [35] and [34], as the total number of attributes in the system in [6], [24], [35] and [34] can be large, i.e., larger than the number of roles associated with a ciphertext in our scheme (i.e., $|U_A| > |\Gamma|$).

![Fig. 5. Data encryption time in our scheme, Sun et al.'s Scheme [6] and Mia et al.'s Scheme [34].](image)

**Trapdoor Generation.** A user needs to perform $(3 + 2|S_{DB}|)$ group exponentiation operations on $G_1$ 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). Similarly, the trapdoor generation cost of [3], [24], and [34] depends on the number of attributes associated with a user, while in [6] and [35], it depends on the total number of attributes in the system. From Table 4, we can observe that our scheme is less costly for generating a trapdoor compared with [6] and [35], while it is comparable to [3], [24] and [34]. Fig. 6, shows the actual computation time for the trapdoor generation of our scheme and [34]. It can be observed that the trapdoor generation time of our scheme and [34] is comparable. Our scheme takes less time than [6] when $|U_A| > 1.5|S_{DB}|$.

**Data Search.** In the Data Search phase, the public cloud first verifies the user and freshness of the trapdoor. This requires $2 + |\Gamma|$ group exponentiation operations on $G_1$ and three pairing operations. It can be observed that the cost of the verification operation (i.e., AUTHENTICATION algorithm) depends on the number of SAs associated with the RBAC access policy of a ciphertext. Fig. 7 shows the

![Fig. 7. Table 3](image)
computation time of our scheme for the verification operation. We can observe that our scheme takes approximately 6ms and 50ms to verify a search request of a user (or access policy) associated with 1 and 50 organizations respectively. After successful authentication (or verification) of the user, the public cloud computes at most $j_{S_{ID}} + 1$ group exponentiation operations on $G_1$, and $2 + |S_{ID}|$ pairing operations to complete the $\text{KEY}\text{SEARCH}$ algorithm for the keyword search. It can be observed that the cost of the $\text{KEY}\text{SEARCH}$ algorithm of our scheme depends on the number of roles associated with the trapdoor. Similarly, the keyword search cost in [3] and [34] depends on the attributes associated with a trapdoor; while [6] and [35] depend on the total number of attributes in the system. Although [24] performs better than our scheme in terms of keyword search cost, [24] only supports single keyword search. Fig. 8 shows the computation time of our scheme and [6], [34] for keyword search operation. It can be observed that our scheme takes less time to perform keyword search operation compared with [34]. Our scheme also performs better than [6] when $|U_{\Delta}| \geq 3|S_{ID}|$. Finally, the public cloud computes at most $|S_{ID}| + |G_{\phi}|$ group exponentiation operations and $1 + |S_{ID}|$ pairing operations to complete the $\text{PARTIAL}\text{DEC}$ algorithm. It can be observed that the cost to perform the $\text{PARTIAL}\text{DEC}$ algorithm depends on the number of roles and the number of SAs associated with the RBAC access policy. Fig. 9 shows the computation time of the partial decryption operation of our scheme. Our scheme takes 61ms to perform the partial decryption operation over a ciphertext for 50 roles. Note that [24] also provides outsourced decryption facility, and its cost mainly depends on the number of attributes associated with a trapdoor.

Decryption. Unlike the other schemes, our scheme, [24] and [34] support the decryption operation. In our scheme and [24], as most of the computationally expensive cryptographic operations are outsourced to the public cloud, a user requires only one group exponentiation operation on $G_T$ to decrypt a ciphertext. On the other hand, the decryption cost in [34] depends mainly on the number of attributes associated with a user. As such, the decryption cost in our scheme is significantly less. Thus, our scheme is also suitable for an environment such as IoT, where the end-users have limited computing resources. Fig. 10 shows the computation time comparison for the decryption phase between our scheme and [34]. We can observe that the decryption operation in our scheme takes approximately (constant) 1ms per ciphertext.

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 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 $G_1$ 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 ciphertexts and users associated with the revoked roles. On the other hand, the revocation cost in [6] and [24] depend mainly on the number of ciphertexts and users associated with the revoked attributes, as it requires ciphertext re-encryption and key update operations. The revocation cost in [34] mainly depends on the number of users associated with the revoked attributes, as it only requires a key update operation. We can see that our revocation process is more costly than [24] and [34]. However, unlike our scheme, [24] and [34] do not support complete user revocation.

### 6.2.3 Storage and Communication Overhead Comparison

Similar to the computation overhead comparison in the previous section, in this section, we compare the storage and communication overhead of our scheme with [3], [6], [24], [35] and [34]. Table 5 shows the storage and communication overhead of our scheme with [3], [6], [24], [35] and [34]. For the evaluation purpose, the ciphertext size, size of the secret keys possessed by a user, and the trapdoor size are considered. From Table 5, it can be observed that the ciphertext size in our scheme increases with the number of roles associated with a ciphertext. For each role $r^k_i$, the owner computes two ciphertext components $C^i_{r^k}$ and $C^j_{r^k}$. We can see from Table 5 that ciphertext size in our scheme is less than [6], [24], [35], and [34], while it is comparable to [3].

A user keeps a private key $Priv_{ID_u}$ for each organization, and the user also keeps a common private key $Priv_{ID_o}$ for all the organizations. Moreover, the user keeps two role-keys for each role he/she possesses. 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 $tr^i_{r^k}$ and $tr^j_{r^k}$ for each role $r^k$ associated with the trapdoor. We can observe that the secret key size of [3] is less than our scheme, while secret key size of the other schemes is either more or comparable to ours. The trapdoor size of [24] is smaller than ours, while the trapdoor size in other schemes is either more or comparable to ours.

### 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 article was supported in part by European Union’s Horizon 2020 research and innovation programme under the Grant 830892 and Project SPARTA.

### REFERENCES

[1] F. Han, J. Qin, and J. Hu, “Secure searches in the cloud: A survey,” *Future Gener. Comput. Syst.*, vol. 62, pp. 66–75, 2016.
Nazatul Haque Sultan (Member, IEEE) received the BE degree in computer science and engineering from Jorhat Engineering College, Dibrugarh University, the MTech degree in information technology from Tezpur University, and the PhD degree from the Indian Institute of Information Technology Guwahati. He was a research and development engineer with Telecom SudParis, Institut Polytechnique de Paris. He is currently a research associate with the School of Electrical Engineering and Computer Science, University of Newcastle. Previously, his research interests include security and privacy related to cloud and IoT, searchable encryption, access control, blockchain, and applied cryptography.

Maryline Laurent (Member, IEEE) is currently a professor with Télécom SudParis, Institut Mines-Télécom, France. She leads the research team R3S (Network, Systems, Services, Security) with the CNRS SAMOVAR lab of Télécom SudParis. Her research interests include network security and privacy applied to clouds, Internet of Things, and identity management. She is co-founder of the chair of Institut Mines-Télécom on “Values and Policies of Personal Information”. She is an area editor of the Annals of Telecommunication journal, and the editor of several books including “the Digital Identity Management” (2015). She chaired the IFIP WISTP 2019 conference on Information Security Theory and Practice.

Vijay Varadharajan (Senior Member, IEEE) is currently the global innovation chair professor with Cyber Security, University of Newcastle. He is also the director with the Advanced Cyber Security Research Centre. He has authored or co-authored more than 400 papers in international journals and conferences. He is or was on the editorial board of several journals, including the ACM Transactions on Information and System Security, IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Information Forensics and Security, and IEEE Transactions on Cloud Computing.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdli.