
Article:
Sultan, N.H., Kaaniche, N. orcid.org/0000-0002-1045-6445, Laurent, M. et al. (1 more author) (2019) Authorized keyword search over outsourced encrypted data in cloud environment. IEEE Transactions on Cloud Computing, 10 (1). pp. 216-233. ISSN 2168-7161

https://doi.org/10.1109/tcc.2019.2931896

Authorized Keyword Search over Outsourced Encrypted Data in Cloud Environment

Nazatul Haque Sultan, *Student Member, IEEE,* Nesrine Kaaniche, *Member, IEEE,* Maryline Laurent, *Member, IEEE,* and Ferdous Ahmed Barbhuiya, *Member, IEEE*

Abstract—For better data availability and accessibility while ensuring data secrecy, end-users often tend to outsource their data to the cloud servers in an encrypted form. However, this brings a major challenge to perform the search for some keywords over encrypted content without disclosing any information to unintended entities. This paper proposes a novel expressive authorized keyword search scheme relying on the concept of ciphertext-policy attribute-based encryption. The originality of the proposed scheme is multifold. First, it supports the generic and convenient multi-owner and multi-user scenario, where the encrypted data are outsourced by several data owners and searchable by multiple users. Second, the formal security analysis proves that the proposed scheme is semantically secure against chosen keyword and outsiders keyword guessing attacks. Third, an interactive protocol is introduced which avoids the need of any secure-channels between users and service provider. Fourth, due to the concept of bilinear-map accumulator, the system can efficiently revoke users and/or their attributes, and authenticate them prior to launching any expensive search operations. Fifth, conjunctive keyword search is provided thus enabling to search for multiple keywords simultaneously, with minimal cost. Sixth, the performance analysis shows that the proposed scheme outperforms closely-related works.

Index Terms—keyword search, conjunctive keyword search, searchable encryption, attribute-based encryption, user revocation, keyword guessing attack.

1 Introduction

With ever increasing popularity of cloud computing, cloud users are outsourcing their data to cloud servers. This enables the cloud users, i.e. individuals/ organizations, to take the advantages offered by cloud computing such as *on demand service, low investment cost*, *ubiquitous* and *flexible access* [1]. Usually, the data are outsourced to the remote cloud servers, also known as cloud storage servers. These servers are hosted by a third-party service-provider popularity named as Cloud Service Provider (CSP). As the outsourced data are getting out of the perimeter of data owners (who own the data) while going under full control of the CSP, the data owners (Owners) have to rely on the CSP for safekeeping of those data. However, the outsourced data may contain sensitive information, such as medical health records, financial documents, personal photographs, etc., which the CSP may illegally access for various motivations [2]. Therefore, trusting the CSP blindly for security of the outsourced data may not be a good idea.

Encryption-before-outsourcing is the fundamental approach to preserve data privacy and to prevent unauthorized access [2], [3]. In this approach, the data are encrypted before outsourcing to the cloud storage servers. However, the data encryption makes searching over the encrypted data a difficult task [1]. The trivial solution is to download the whole database locally, decrypt it using the corresponding key and then perform search (i.e. *keyword search*). Clearly, decryption of the whole database involves high computation on the local machine and is a cumbersome task. Downloading the whole database for each and every keyword search incurs heavy communication overhead on the system. Thus, this solution is not practical. The other possible solution is to let CSP decrypt the outsourced data and perform keyword search over the decrypted data. This approach violates data privacy [4], as the CSP will get to know about the plaintext data.

Searchable Encryption (SE) is a preferred approach to address these issues, where the users (who use the outsourced data) delegate keyword search capabilities over encrypted data to CSP without disclosing any useful information. Notable works in this area mainly focus on *single-owner and multi-user* scenario, where a single owner manages his/her encrypted data and search capabilities over them [5], [6], [7], [8], [9]. Some of the *single-owner and multi-user* based schemes need to grant search access rights to the users via a shared secret key [5], [6], [9]; while the others need to issue search access rights in the form of *trapdoors* [7], [8]. The shared secret key based SE schemes [5], [6], [9] need to perform frequent key re-distribution among the authorized users to maintain a valid secret key, which is a costly and cumbersome task [10]. While the *trapdoor* based SE schemes [7], [8] need the Owners to be online all the time for issuing *trapdoor* to the authorized users. Moreover, these schemes are not suitable for *multi-owner and multi-user* based data outsourcing platforms. The main benefit of this kind of platforms is that many Owners outsource their data and many users access those data [11]. For example, in the platforms like [12], [13], [14], many Owners share

- *N. H. Sultan and F. A. Barbhuiya are with the Department of Computer Science and Engineering, Indian Institute of Information Technology Guwahati, Assam, India, 781015. E-mail: [nazatul,ferdous]@iiitg.ac.in*
- *N. Kaaniche is with the Department of Computer Science, University of Sheffield, UK. E-mail: n.kaaniche@sheffield.ac.uk*
- *M. Laurent is with SAMOVAR, CNRS, Télécom SudParis, Evry, France. E-mail: maryline.laurent@telecom-sudparis.eu*

their data with multiple users while outsourcing. The shared secret key based SE schemes cannot perform well in these platforms due to the high complexity of secret key management among a large number of users. Further, the *trapdoor* based SE schemes require Owners availability for granting authorizations to users in real time. Thus, it is a challenging task to design a SE scheme which performs well in a *multi-owner and multi-user* based cloud environment along with fine-grained search authorization[1].

Attribute-Based Encryption (ABE) has already emerged as a popular cryptographic technique for achieving fine-grained access control over encrypted data in untrusted cloud environment [15], [16], [17], [18], [19], [20]. In ABE, encrypted data and secret keys are closely linked to some attributes and any user is able to decrypt an encrypted data if there is a match between the attributes associated with the encrypted data and the attributes associated with the secret keys of the users [21]. ABE has two variants: Key-Policy Attribute-Based Encryption (KP-ABE) [22] and Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [23]. In KP-ABE, data are encrypted using some attributes and the secret keys are associated with an access policy defined over the attributes. On the contrary, in CP-ABE, access policy is associated with the encrypted data and the attributes are associated with the secret key of the users. Thus, CP-ABE can be a suitable mechanism to design *multi-owner and multi-user* based SE system, as it provides the Owners to choose access policies of their choice and any user having a qualified set of attributes can search those encrypted data. Using the advantages of CP-ABE, Sun *et al.* [11], [24] and Hu *et al.* [25] design *multi-owner and multi-user* based SE schemes for achieving various functionalities. However, Sun *et al.*'s schemes [11], [24] do not support expressive access policy[2] and practical usability of the schemes are also questionable due to the use of excessive costly cryptographic operations [26]. Moreover, the existing schemes [11], [24], [25] require setting up secure-channels to communicate between the users and CSP. Otherwise, any outsider can launch a *keyword guessing attack*[3], which may compromise data secrecy. It is to be noted that, the construction of a secure-channel is costly in case high-level security is required, as a cumbersome public key infrastructure with heavy trust relationship management is usually needed [27], [28]. There are some additional challenges such as conjunctive keyword search[4], granting search access right to the newly joined users and revoking search access right from a user in the keyword search area that need to be addressed.

In this paper, we further investigate the issues of keyword search mechanisms and propose a secure and efficient scheme for authorized keyword search over encrypted data for *multi-owner and multi-user* cloud environment. The salient features of this paper are enumerated as follows:

1) an expressive fine-grained authorized keyword search scheme is designed for *multi-owner and multi-user* scenarios using the concept of CP-ABE.
2) an *interactive protocol* is constructed between the user and CSP to avoid overhead involved in establishing secure-channel.
3) essential functionalities, like conjunctive keyword search and user revocation, are added without any significant overhead.
4) the proposed scheme is proved semantically secure against *chosen keyword attack* and *outsider's keyword guessing attack* under the standard assumptions.
5) the implementation of the proposed scheme show that it takes substantially less computation time to perform keyword search which makes it suitable for real life cloud environment compared with the closely related works [24] and [25].

The rest of this paper is organized as follows: Section 2 presents a brief overview on some notable works. Problem Statement of the proposed scheme is presented in Section 3, followed by Preliminaries in Section 4. The proposed scheme is described in Section 5 followed by its Security Analysis in Section 6 and Performance Analysis in Section 7. Finally, the paper is concluded in Section 8.

2 Related Works

After the introduction of the first public key encryption keyword search (PKEKS) scheme by Boneh *et al.* [29], several mechanism has been proposed for enhancing security, robustness, and efficiency, e.g. resistance against keyword guessing attacks [30], [31], [32], multi-key keyword search [33], [34], [35], conjunctive keyword search [36], [37], [38], verifiable keyword search [39], [40], [41], keyword search across multiple independent CSPs [42], keyword search in multi-authority[5] settings [43] and so on. Below, we provide a brief overview of some existing works related to the proposed scheme which we divide into two main categories: *Secure-Channel Free Public Key Encryption Keyword Search* and *Public Key Encryption Authorized Keyword Search*.

2.1 Secure-Channel Free Public Key Encryption Keyword Search (SCF-PKEKS)

In [28], Baek *et al.* highlight that Boneh *et al.*'s scheme needs secure-channel between a user and the CSP to transmit trapdoors, which may incur high overhead in terms of communication and processing costs. In addition, if (somehow) an adversary captures a trapdoor of a user, he/she can use the captured trapdoor for launching a replay attack. So, Baek *et al.* proposed an SCF-PKEKS scheme [28] to address the issue of Boneh *et al.*'s scheme [29]. Later, Rhee *et al.* proposed [44] to enhance the Baek et al.'s security model [28] for SCF-PKEKS, where they allow the attacker to obtain the relation between non-challenge ciphertexts and a trapdoor. Fang *et al.* proposed two SCF-PKEKS schemes [30] and [45], where the former is designed without using random oracle model and the latter, is designed to prevent outsider keyword

[1]Fine-grained search authorization means that different users may possess different search access rights over different encrypted data.

[2]Expressive access policy consists of both boolean *"AND"* and *"OR"* gates. The access policy in [11], [24] consists of only boolean *"AND"* gates.

[3]In keyword guessing attack, an attacker successfully guesses the keywords [27].

[4]In conjunctive keyword search, the user should be able to search multiple keywords with one search request [24].

[5]In multi-authority settings, a user may possess search access rights from multiple independent authorities.

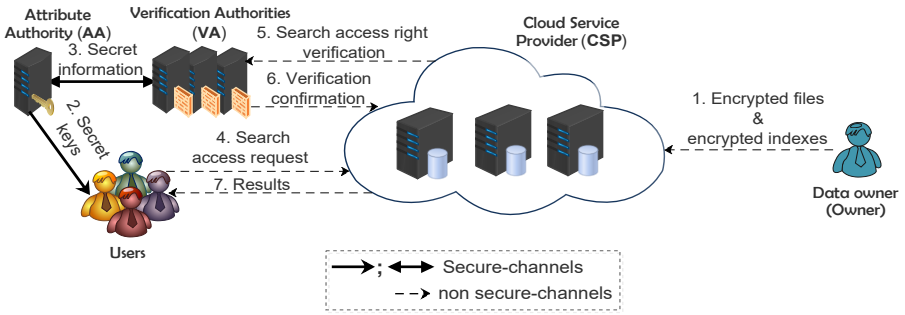


Fig. 1: Proposed Cloud Architecture

guessing attack without random oracle. Recently, Jiang *et al.* proposed an SCF-PKEKS scheme [27] where the users can perform keyword search over the authorized encrypted data only. However, none of the SCF-PKEKS schemes provide fine-grained search authorization over encrypted data.

2.2 Public Key Encryption Authorized Keyword Search (PKAKS)

Li *et al.* proposed an PKAKS scheme [46] using the concept of Hierarchical Predicate Encryption [47] for *multi-owner and multi-user* setting, where only authorized users can perform keyword search over the encrypted indexes. However, it incurs high communication overhead, as a user needs to contact the attribute authority to generate trapdoors for each search request. Also, the scheme is suitable to perform keyword search over structured database which contains limited keywords [24]. Sun *et al.* addressed the issue of Li *et al.*'s scheme [46] and proposed an PKAKS scheme [11] using the concept of CP-ABE to provide fine-grained search authorization functionality. A few notable PKAKS schemes based on ABE techniques have appeared to provide fine-grained search authorization [24], [25], [26]. However, Sun *et al.*'s schemes [11], [24] are not expressive, as the access policies are constructed using boolean "*AND*" gates only. Also, practical usability of the two schemes is questionable, due to the heavy computational overhead on the system [26]. In addition, encrypted index size linearly increases with the total number of attributes in the system. Moreover, the schemes [11], [24] need secure-channel between a user and CSP to transmit the trapdoors which is essential to prevent replay attacks[6] and keyword guessing attacks. Cui *et al.* proposed a KP-ABE based keyword search scheme [26], which uses both "*AND*" and "*OR*" in the access policies. However, in [26], a trusted authority is responsible for choosing access policy related to the keywords to be searched, i.e., the trusted third-party needs to be contacted by the users each time a fresh keyword search is requested for embedding the chosen access policy in the trapdoor. As the trusted third-party is required to remain online, this is a major limitation of [26]. Hu *et al.*'s scheme [25] is expressive, and also it can update access policies at any time with the help of the CSP using proxy re-encryption and secret sharing techniques. However, like [11], [24], Hu *et al.*'s scheme [25] needs secure-channels between the users and CSP to transmit trapdoors. Moreover, the scheme does not provide conjunctive keyword search and user revocation.

[6]In [11], [24], if an attacker captures a trapdoor, the attacker can launch replay attacks using the received trapdoor.

3 Problem Statement

This section first presents the *System Model* of the proposed scheme followed by its *Design Goals*, *Security Requirements*, *Framework*, and *Adversary* and *Security Model*.

3.1 System Model

The proposed cloud architecture is shown in Figure 1. It consists of five entities, namely, *Attribute Authority* (AA), *Verification Authority* (VA), *Owner*, *Cloud Service Provider* (CSP), and *Users*. In Figure 1, all the dotted lines represent non secure-channels, while all the solid lines represent secure-channels. AA is an independent trusted third-party entity which maintains public parameter and master secret in the system. It also issues access rights in the form of secret keys to the users based on the attributes they possess. VA is also another third-party entity which keeps secret information received from the AA. It is to be noted that VA can also be a set of designated servers under the control of the Cloud Service Provider. VA mainly verifies search access rights of the users upon Cloud Service Provider's request relying on the shared secret information with AA. *Owner* outsources encrypted data to the cloud storage servers. CSP is a third-party entity which maintains the cloud storage servers and performs keyword search over the encrypted data based on a valid search access right of a user and *Users* are requesting entities that are authorized to search and access encrypted data based on their granted privileges.

To enable CSP for performing keyword search, the Owner generates an encrypted index of some keywords for a file and stores the encrypted index along with the encrypted file in remote cloud storage servers. Upon finding matching encrypted indexes for the search access right, the CSP returns the corresponding encrypted files to the requesting user. It is to be noted that the file can be encrypted using any secure encryption mechanism, like *Advanced Encryption Standard (AES)*, which is outside the scope of this paper.

3.2 Design Goals

The design of the proposed scheme is motivated by providing security guarantees and efficiency, while ensuring:

i) *Expressive authorized keyword search*: the proposed scheme should first support general access policies that can be presented in any boolean formula with "*AND*" and "*OR*" gates. Second, it should allow only the authorized users to perform keyword search.

ii) *Multi-owner and multi-user keyword search*: the scheme should accommodate many Owners and users. Each user should be able to perform search over the encrypted data belonging to multiple Owners.

iii) *Conjunctive keyword search*: the scheme should be able to search multiple keywords with one search request.

iv) *Efficient revocation*: the scheme should be able to revoke users and their attributes efficiently.

v) *Secure-channel free keyword search*: users should be able to send their trapdoors with no use of secure-channels.

3.3 Security Requirements

The proposed scheme has to ensure the following security requirements:

a) *Keyword secrecy*: without qualified trapdoors any outsider and the CSP should not be able to learn any useful information about the plaintext keywords from the encrypted indexes. Similarly, an outsider should not be able to learn any useful information about the plaintext keywords from the trapdoors. These two security notions can be captured by *Keyword Semantic Security*, which implies that encrypted indexes do not reveal any useful information about the plaintext keywords. This security notion is referred to as *selective indistinguishability against chosen keyword attack under selective ciphertext policy model* (IND-sCP-CKA).

b) *Keyword guessing attack*: in practical scenarios, the keyword space is considered to be small (limited) [45]. Therefore, any malicious entity may try to guess a plaintext keyword from the encrypted index as well as from the trapdoor by exhaustively guessing the keywords offline, which is referred to as *keyword guessing attack*. The keyword guessing attack can be launched by an insider (i.e. cloud servers) or by an outsider (neither the cloud servers nor the receiver). It is desirable that a scheme should prevent these two attacks. But, according to the earlier works [45], [48], it is impossible to design a public key keyword search scheme which can protect insider's keyword guessing attack. Thereafter, in this paper, we only consider the outsider keyword guessing attack.

3.4 Framework

In a high-level system overview, the proposed scheme is organized into five phases: *Initialization, Owner Registration, Encryption, Key Generation*, and *Keyword Search*. In *Initialization* phase, AA generates system parameters by initiating ASETUP algorithm. Before enciphering search indexes, Owners need to register himself/herself to the AA. This registration process is done in the *Owner Registration* phase, where AA starts the OREG algorithm to register the Owners. Before outsourcing data to the remote cloud servers, Owner encrypts selected keywords of a file and forms an index.

TABLE 1: ACRONYMS

Notation	Description
q	A large prime number
$\mathbb{F}_q$	A field of integer modulo q
$\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$	Multiplicative cyclic subgroups of $\mathbb{F}_q$
$\mathbb{Z}_q^*$	Multiplicative group of $\mathbb{Z}_q$, the integer modulo q
$\mathbb{CT}$	Encrypted index
ID_i, ID_o	Unique identity of i^{th} user and an Owner respectively
$\mathbb{W}$	A set of keywords
$\mathbb{U}_\mathbb{A}$	Attribute universe
$\mathbb{S}_{ID_i}$	Attribute set possessed by the i^{th} user
Γ	Access policy
$\mathtt{Pub}_{\mathtt{ID_i}}$	Public key of the i^{th} user
$\mathtt{SK}_{\mathtt{ID_i}}$	Secret key of the i^{th} user
$\mathtt{D_{pj}}, \mathtt{D'_{pj}}$	Components of the secret key $\mathtt{SK}_{\mathtt{ID_i}}$ related to the j^{th} attribute
$\mathtt{Priv}^1_{\mathtt{ID_i}}, \mathtt{Priv}^2_{\mathtt{ID_i}}$	Private keys of the i^{th} user
$\mathtt{AW}_{\mathbb{S}_{\mathtt{ID_i}}}$	Attribute witness of the i^{th} user

Afterwards, Owner sends the encrypted index and the corresponding encrypted file to the CSP. Owner encrypts the keywords by initiating ENCRYPT algorithm. When User wants to retrieve files from the CSP, he/she initiates *Keyword Search* phase. The *Keyword Search* phase is further divided into two sub-phases: *Trapdoor Generation* and *Search*. In the *Trapdoor Generation* phase, the user delegates search capability to the CSP by sending his/her search access rights in a trapdoor form. User derives trapdoor by initiating an *interactive protocol* with the CSP. When the interactive protocol is over, the user initiates the TRAPGEN algorithm to derive trapdoor. Upon receiving the trapdoor from the user, CSP first checks the validity of the user's search access right by taking assistance from the VA which runs the ATTVERI algorithm. Afterwards, CSP runs the KEYSEARCH algorithm to perform keyword search over the encrypted indexes using the received trapdoor. A brief overview of the different algorithms of these phases is given below. The acronyms use in this paper are shown in Table 1.

1) ASETUP $((\mathtt{MSK}, \mathtt{PP}) \leftarrow 1^\lambda)$: this algorithm takes a security parameter 1^λ as input and outputs master secret key MSK and public parameter PP.

2) OREG $(\mathtt{EKey}_{\mathtt{ID_o}} \leftarrow (\mathtt{MSK}, ID_o))$: this algorithm takes the master secret key MSK and identity ID_o as input and outputs an encryption key $\mathtt{EKey}_{\mathtt{ID_o}}$ for the Owner ID_o.

3) ENCRYPT $(\mathbb{CT} \leftarrow (\mathbb{W}, \Gamma, \mathtt{EKey}_{\mathtt{ID_o}}, \mathtt{PP}))$: it takes a keyword set $\mathbb{W}$, an access policy Γ, Owner's encryption key $\mathtt{EKey}_{\mathtt{ID_o}}$, and public parameter PP as input and outputs an encrypted index $\mathbb{CT}$ for the keyword set $\mathbb{W}$.

4) KEYGEN $((\mathtt{SK}_{\mathtt{ID_u}}, \mathtt{Pub}_{\mathtt{ID_u}}) \leftarrow (\mathtt{MSK}, ID_u, \mathbb{S}_{ID_u}))$: this algorithm takes the master secret key MSK, unique identity ID_u of a user, and attribute set $\mathbb{S}_{ID_u}$ of the user ID_u as input and outputs a secret key $\mathtt{SK}_{\mathtt{ID_u}} = \langle \{\mathtt{D_{pi}}, \mathtt{D'_{pi}}\}_{i \in \mathbb{S}_{ID_u}}, \mathtt{Priv}^1_{\mathtt{ID_u}}, \mathtt{Priv}^2_{\mathtt{ID_u}}, \mathtt{AW}_{\mathbb{S}_{\mathtt{ID_u}}} \rangle$ and a public key $\mathtt{Pub}_{\mathtt{ID_u}}$ for the user ID_u.

5) TRAPGEN $(\mathtt{Trap} \leftarrow (w, \mathtt{SK}_{\mathtt{ID_u}}, \mathtt{Pub}_{\mathtt{ID_u}}))$: it is executed in the *Trapdoor Generation* phase by the user. It takes a plaintext keyword w, secret key $\mathtt{SK}_{\mathtt{ID_u}}$ and public key $\mathtt{Pub}_{\mathtt{ID_u}}$ of a user ID_u as input and outputs a trapdoor $\mathtt{Trap} =$

$\langle \{tr_{1i}, tr_{2i}\}_{\forall i \in \mathbb{S}_{ID_u}}, tr_3, tr_4, tr_5, tr_6 \rangle$.

6) ATTVERI $((\{(g_1^d)^{\varphi \cdot a_i}\}_{\forall i \in \mathbb{S}_{ID_u}} / \perp) \leftarrow (\mathtt{MK}, tr_6, g_1^d, \mathbb{S}_{ID_u}, ID_u))$: this algorithm is initiated by the VA in the *Search* phase. It takes secret MK, trapdoor component tr_6, g_1^d, and attribute set $\mathbb{S}_{ID_u}$ of the user ID_u as input, where g_1^d is sent by the user ID_u to the CSP during the *interactive protocol* which is described in Section 5.2.5 and MK is a secret tuple sent by AA to the VA which is defined in Section 5.2.1. It outputs either $\{(g_1^d)^{\varphi \cdot a_i}\}_{\forall i \in \mathbb{S}_{ID_u}}$ or $\perp$, where a_i represents the secret key associated with the i^{th} attribute and $(\varphi, \{a_i\}_{\forall i \in \mathbb{U}_\mathbb{A}}) \in \mathtt{MK}$. It is to be noted that, $\perp$ represents unsuccessful authentication (i.e., the user ID_u does not have qualified attributes). It is also to be noted that VA computes $\{(g_1^d)^{\varphi \cdot a_i}\}_{\forall i \in \mathbb{S}_{ID_u}}$ after successful authentication of the user ID_u (i.e., user ID_u possesses qualified attributes).

7) KEYSEARCH $(Files \leftarrow (\{(g_1^d)^{\varphi \cdot a_i}\}_{\forall i \in \mathbb{S}_{ID_u}} / \perp, \mathtt{Trap}, \mathbb{S}_{ID_u}, \mathbb{CT}))$: it is run by the CSP in the *Search* phase after the ATTVERI algorithm. If the algorithm takes $\perp$ as input, then it aborts. If it takes $\{(g_1^d)^{\varphi \cdot a_i}\}_{\forall i \in \mathbb{S}_{ID_u}}$, trapdoor Trap, attribute set $\mathbb{S}_{ID_u}$ of the user ID_u, and encrypted index $\mathbb{CT}$ as input, then it outputs all the (encrypted) $Files$ related with the encrypted indexes which match with the Trap.

3.5 Adversary and Security Model

This section first presents the adversary model followed by the security model.

3.5.1 Adversary Model

In the proposed scheme, we consider two kinds of adversaries: honest-but-curious entity and malicious entity. The CSP and the VA are considered to be the honest-but-curious entities. They honestly perform all the tasks assigned to them, but they may try to gain extra knowledge based on the data available to them. Any malicious outsider (neither the CSP nor the legitimate user) may try to learn about the plaintext keywords from the trapdoors which are transmitted through public channels. It is assumed that the CSP does not collude with the malicious users, like in [11], [24].

3.5.2 Security Model

The security model of the proposed scheme is defined by the following two games, namely *Semantic Security against Chosen Keyword Attacks* (IND-sCP-CKA) and *Indistinguishability against Outsider's Keyword Guessing Attacks* (IND-O-KGA) which are presented in Section 3.5.2.1 and Section 3.5.2.2 respectively.

3.5.2.1 **Semantic Security against Chosen Keyword Attacks**: The semantic security game IND-sCP-CKA is played between an adversary $\mathcal{A}_1$ and a challenger defining the following steps:

- INIT The adversary $\mathcal{A}_1$ submits a challenge access policy Γ^* to the challenger.

- SETUP The challenger runs the ASETUP algorithm to obtain the public parameters PP and master secret MSK. The challenger sends the public parameters PP, and extra secret information φ and $\{a_i\}_{\mathbb{U}_\mathbb{A}}$ to the adversary $\mathcal{A}_1$. It keeps MSK secret.

- PHASE 1 Adversary $\mathcal{A}_1$ makes the following queries:

- **Interactive protocol queries** Adversary $\mathcal{A}_1$ submits a challenged identity to the challenger and the challenger returns interactive protocol parameters to the adversary.
- **Trapdoor queries** Adversary $\mathcal{A}_1$ requests for trapdoors by submitting a keyword $w \in \mathbb{W}$ to the challenger. It is to be noted that the adversary $\mathcal{A}_1$ can query for trapdoors by polynomially many times to the challenger. The challenger generates the corresponding trapdoor using TRAPGEN algorithm and sends the trapdoor to the adversary $\mathcal{A}_1$.

- CHALLENGE The adversary $\mathcal{A}_1$ submits two equal length keywords w_0 and $w_1 \in \mathbb{W}$, which were not challenged before, to the challenger. The challenger flips a random coin $b \in \{0, 1\}$ and generates an encrypted index for the keyword using ENCRYPT algorithm, where Γ^* is the challenge access policy. The challenger returns the encrypted index $\mathbb{CT}_b$ to the adversary $\mathcal{A}_1$.

- PHASE 2 Same as PHASE 1.

- GUESS The adversary $\mathcal{A}_1$ submits a guess $b' \in \{0, 1\}$ and wins the game if $b = b'$.

It is to be noted that in the **Phase 1** the challenger sends φ and $\{a_i\}_{\mathbb{U}_\mathbb{A}}$ to the adversary $\mathcal{A}_1$. This enables the adversary $\mathcal{A}_1$ to answer all the queries of the attribute verification oracle. Therefore, we are not considering attribute verification oracle in our security model. Also, the trapdoor query oracle in PHASE 1 implicitly includes the key generation query oracle, which may send the secret keys (please refer to Section 5.2.4) back to the adversary $\mathcal{A}_1$. Moreover, in our selective model, the game allows the adversary $\mathcal{A}_1$ to query any keywords of its choice at PHASE 1 and PHASE 2 as long as the attribute sets related with the queried trapdoors do not satisfy the challenge access policy Γ^*.

Definition 3.1. The proposed scheme is IND-sCP-CKA secure if the advantage $Adv_{\mathcal{A}_1}^{\text{IND-sCP-CKA}}$ in winning the security game is negligible for any polynomial time adversary $\mathcal{A}_1$.

$$Adv_{\mathcal{A}_1}^{\text{IND-sCP-CKA}}(1^\lambda) = \left| Pr[b' = b] - \frac{1}{2} \right|$$

3.5.2.2 **Indistinguishability against Outsider's Keyword Guessing Attacks**: The IND-O-KGA ensures that outsiders cannot guess the keyword from a given trapdoor using keyword guessing attacks. The IND-O-KGA game [45] is played between a challenger and an adversary as follows:

- SETUP The challenger runs the ASETUP algorithm to obtain the public parameters PP and master secret MSK. The challenger sends the public parameters PP, φ, and $\{a_i\}_{\mathbb{U}_\mathbb{A}}$ to the adversary $\mathcal{A}_2$. It keeps MSK secret.

- PHASE 1 Adversary $\mathcal{A}_1$ makes the following queries:

- **Interactive protocol queries** Adversary $\mathcal{A}_1$ submits a challenged identity to the challenger and the challenger returns interactive protocol parameters.
- **Trapdoor queries** Adversary $\mathcal{A}_2$ requests for trapdoors by submitting a keyword $w \in \mathbb{W}$ and attribute set of his/her choice to the challenger. It is to be

noted that the adversary $\mathcal{A}_2$ can query for trapdoors by polynomially many times to the challenger. The challenger generates the corresponding trapdoor using TRAPGEN algorithm and sends the trapdoor to the adversary $\mathcal{A}_2$.

- CHALLENGE The adversary $\mathcal{A}_2$ submits two equal length keywords w_0 and $w_1 \in \mathbb{W}$, which were not challenged before, to the challenger. The challenger flips a random coin $b \in \{0,1\}$ and generates a trapdoor $\mathtt{Trap}_b$ for the keyword using TRAPGEN algorithm. The challenger returns the generated trapdoor $\mathtt{Trap}_b$ to the adversary $\mathcal{A}_2$.

- PHASE 2 Same as PHASE 1.

- GUESS The adversary $\mathcal{A}_2$ submits a guess $b' \in \{0,1\}$ and wins the game if $b = b'$.

Similar with the semantic security game IND-sCP-CKA, the adversary $\mathcal{A}_2$ is able to answer all attribute verification queries by itself. So, we are not considering attribute verification oracle in the security model.

Definition 3.2. The proposed scheme is IND-O-KGA secure if the advantage $Adv_{\mathcal{A}_2}^{\text{IND-O-KGA}}$ in winning the security game is negligible for any polynomial time adversary $\mathcal{A}_2$.

$$Adv_{\mathcal{A}_2}^{\text{IND-O-KGA}}(1^\lambda) = \left| Pr[b' = b] - \frac{1}{2} \right|$$

4 PRELIMINARIES

This section presents a brief description on some basic concepts, e.g. *Sibling Intractable Function Family, Bilinear Map, Bilinear-map Accumulator*, and some mathematical assumptions, which shall be used in the following sections of this paper.

4.1 The Sibling Intractable Function Family (SIFF)

SIFF is a generalization of the concept of the universal one-way hash function family [49]. Zheng *et al.* presented the concept of SIFF in [49]. In SIFF, given a set of initial strings colliding with one another, it is hard to compute another string that would collide with the initial strings.

SIFF can be applied to solve many existing problems [49]. One of the applications of SIFF is the shared mailbox problem, i.e., *how to manage 1 million passwords for shared devices?* Let us assume that a system has n users, and it wants to share a message m among the n users after encrypting using a secret key, say $\mathtt{sk}$. Typically, each n users will have a different key, say i^{th} user possesses the key $\mathtt{k_i}$ where $1 \leq i \leq n$. The system will choose a polynomial $P(x)$, where

$$P(x) = x^n + a_1 x^{n-1} + ... + a_j x^{n-j} + ... + a_n$$

Now, the system computes an equation $P(x) = \mathtt{sk}$, where each key of the n users will be a root of the equation. Afterward, the system can compute the coefficients of the equation and publishes them. As a result, a user among the n users who holds a key, say $\mathtt{k_i}$, can get the secret key $\mathtt{sk}$ by computing $P(\mathtt{k_i})$. As such, the user can access the message m by decrypting the encrypted message.

4.2 Bilinear Map

This section introduces the concept of asymmetric bilinear pairings, used in this paper, which is modified Weil/Tate pairing defined on Elliptic Curve [50].

a) Bilinear pairing: Let g_1 and g_2 be two random generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. The Bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ has the following properties:

- *Bilinear*: $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$, $\forall g_1, g_2$ and $\forall (a,b) \in \mathbb{Z}_q^{*2}$; where $\mathbb{Z}_q^*$ denotes the multiplicative group of $\mathbb{Z}_q$, the integer modulo q.
- *Non-degenerate*: if g_1 generates $\mathbb{G}_1$ and g_2 generates $\mathbb{G}_2$, then $\hat{e}(g_1, g_2)$ generates $\mathbb{G}_T$.
- *Computable*: there exists an efficient algorithm to compute $\hat{e}(g_1, g_2)$, for all $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$.

Note that, if $\mathbb{G}_1 = \mathbb{G}_2$ then the bilinear pairing is called as *symmetric pairing*.

4.3 Bilinear-map Accumulator

Bilinear-map accumulator [51], [52] is an efficient mechanism which provides a constant-size value, called an *Accumulator*, of a large set of inputs. It also provides another constant-size value, called as *Witness*, to facilitate authentication and revocation of the inputs. For authentication and revocation, the *witness* for any element in the set is used to verify the (non-)membership of the element in this set.

Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_\mathbb{T}$ be multiplicative groups of a large prime number q. Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_\mathbb{T}$ be a bilinear map having all the properties as described in Section 4.2. Let g_1 and g_2 be two generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. An *accumulator* $Acc_\mathbb{L}$ can be constructed as follows: $Acc_\mathbb{L} = g_1^{\prod_{a_i \in \mathbb{L}} (a_i + s)}$, where $\mathbb{L} = \{a_1, a_2, ..., a_n\}$ in $\mathbb{Z}_q^{*n}$ for some positive integer n and $s \in \mathbb{Z}_q^*$. The $\prod_{a_i \in \mathbb{Z}_q^*} (a_i + s)$ is called characteristic polynomial for the set $\mathbb{L}$. A *witness* $Wit_{(\mathbb{L} \setminus \mathbb{L}')}$ for a set $\mathbb{L}' \subseteq \mathbb{L}$ can be constructed as follows: $Wit_{(\mathbb{L} \setminus \mathbb{L}')} = g_1^{\prod_{a_i \in (\mathbb{L} \setminus \mathbb{L}')} (a_i + s)}$. The set $\mathbb{L}'$ verification can be carried as follows:

$$\hat{e}(Wit_{(\mathbb{L} \setminus \mathbb{L}')}, g_2^{\prod_{a_i \in \mathbb{L}'} (a_i + s)}) \stackrel{?}{=} \hat{e}(Acc_\mathbb{L}, g_2)$$

The security of bilinear-map accumulator is based on the q-strong bilinear Diffie-Hellman (q-SBDH) assumption [53].

4.4 Mathematical Assumptions

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative groups of a large prime order q and let g_1 and g_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a non-degenerate bilinear map.

i) *External Diffie-Hellman (XDH) Assumption* [7] [55]: No probabilistic
polynomial-time algorithm is able to distinguish whether $Z = g_1^{ab}$ or $Z = g_1^r$ from the tuple $< g_1, g_1^a, g_1^b, Z >$ with non-negligible advantage, where $(a, b, r) \in \mathbb{Z}_q^{*3}$ (i.e., the DDH assumption holds within $\mathbb{G}_1$).

[7] It is believed that the XDH assumption may hold in certain subgroups of MNT elliptic curves [54].

ii) *Mixed External Diffie-Hellman (MXDH) Assumption*[8] [56]: Given $< g_1, g_1^a, g_1^b, Z, g_2^a, g_2^b >$, no probabilistic polynomial-time algorithm is able to distinguish whether $Z = g_1^{ab}$ or $Z = g_1^r$ from the tuple $< g_1, g_1^a, g_1^b, Z >$ with non-negligible advantage, where $(a, b, r) \in \mathbb{Z}_q^{*3}$ (i.e., the DDH assumption holds within $\mathbb{G}_1$).

5 The Proposed Scheme

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

5.1 Overview

The main goal of the proposed scheme is to enable the Owners to enforce search access policies within the data encrypted index itself to achieve fine-grained search authorization. By leveraging and modifying the concept of Zhao *et al.*'s work [57], the Owners embed expressive access policies in the form of propositional formula, which contains "AND" and "OR" gates, into the encrypted indexes. Any user, who holds a qualified set of attributes that satisfy the access policies of the encrypted indexes, can delegate keyword search capabilities to the CSP without disclosing any useful information about the actual contents of the ciphertexts (i.e., encrypted indexes and encrypted data).

Moreover, for the VA to verify attributes of a user and also to revoke the attributes without performing costly ciphertext re-encryption operations, the proposed scheme uses the concept of bilinear-map accumulators. This search access right verification relying on ID-based techniques efficiently improves the authorization phase, as the expensive *Search* phase is launched only if the requesting user has correct search access rights.

5.2 Construction

Let $\mathbb{U}_\mathbb{A} = \{Att_1, Att_2, ..., Att_{|\mathbb{U}_\mathbb{A}|}\}$ be the attribute universe in the system for a certain positive number $|\mathbb{U}_\mathbb{A}|$. The proposed scheme consists of five phases and these phases are described below in details.

5.2.1 *Initialization*

In this phase, AA initializes the system and generates system parameters like master secret and public parameters. It consists of ASetup algorithm, which is described below.

– **ASetup** $((\mathtt{MSK}, \mathtt{PP}) \leftarrow 1^\lambda)$: AA chooses bilinear groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ of a large prime order q and bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. AA also selects two generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, hash functions $H : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ and $H_1 : \{0,1\}^l \rightarrow \mathbb{Z}_q^*$, and random numbers $(\mu, \rho, \varphi, \varrho, \theta, \upsilon, \{t_i\}_{\forall i \in \mathbb{U}_\mathbb{A}}, \{a_i\}_{\forall i \in \mathbb{U}_\mathbb{A}}) \in \mathbb{Z}_q^{*(6+|\mathbb{U}_\mathbb{A}|)}$. AA publishes the public parameters PP, where

$$\mathtt{PP} = \langle q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, \hat{e}, H, H_1, g_1^\rho, g_1^\mu, g_2^\mu, g_1^\varrho, \{T_i = g_1^{t_i}, T_i' = g_2^{t_i}\}_{(\forall Att_i \in \mathbb{U}_\mathbb{A})} \rangle$$

[8]MXDH is a slightly stronger variant of the XDH assumption [56].

It keeps the master secret MSK in a secure place, where

$$\mathtt{MSK} = \langle g_2, \mu, \rho, \varphi, \varrho, \theta, \upsilon, \{t_i, a_i\}_{\forall i \in \mathbb{U}_\mathbb{A}} \rangle$$

Also, the AA sends the tuple MK, where $\mathtt{MK} = \langle \varphi, \theta, \{a_i\}_{\forall i \in \mathbb{U}_\mathbb{A}} \rangle$ and ϱ to the VA and the CSP respectively using secure-channel[9]. It is to be noted that VA is going to use MK for verification of users' attributes in the *Keyword Search* phase as detailed in Section 5.2.5 and ϱ is going to be used by the CSP to randomize the trapdoors in the *Trapdoor Generation* phase as described in Section 5.2.5.

5.2.2 *Owner Registration*

In this phase, AA initiates registrations of the Owners and issues a secret key for the registered Owner. Any newly joined Owner needs to register himself/herself with the AA. This phase relies on the OReg algorithm defined as follows.

– **OReg** $(\mathtt{EKey}_{\mathtt{ID}_o} \leftarrow (\mathtt{MSK}, ID_o))$: let ID_o be the unique identity of the newly registered Owner. AA chooses a random number $r_o \in \mathbb{Z}_q^*$ and computes secret key $\mathtt{EKey}_{\mathtt{ID}_o}$ for the Owner as follows:

$$\mathtt{EKey}_{\mathtt{ID}_o} = \langle g_1^{\upsilon \cdot r_o}, g_2^{r_o} \rangle$$

Afterwards, AA sends $\mathtt{EKey}_{\mathtt{ID}_o}$ to the newly registered Owner ID_o using a secure-channel.

5.2.3 *Encryption*

In this phase, Owner encrypts a set of keywords of a file. This phase consists of an Encrypt algorithm.

– **Encrypt** $(\mathbb{CT} \leftarrow (\mathbb{W}, \Gamma, \mathtt{EKey}_{\mathtt{ID}_o}, \mathtt{PP}))$: it takes a keyword set $\mathbb{W}$, an access policy Γ, the secret encryption key of the Owner $\mathtt{EKey}_{\mathtt{ID}_o}$ and the public parameters PP as input. It outputs an encrypted index of the keyword set $\mathbb{W}$. The Owner chooses an access policy in propositional formula. Let Γ be the access policy defined as:

$$\Gamma = \{ap_1' \ OR \ ap_2' \ AND \ ap_3' \ AND \ ... \ OR \ ap_{m'}'\}$$

where $ap_1', ap_2', ..., ap_m'$ are sub-propositional formula of Γ. Owner can easily transform the access policy Γ into *Sum of Product* (SOP) form. In the sequel, the new access policy is defined as:

$$\Gamma = \{ap_1 \ OR \ ap_2 \ OR \ ... \ OR \ ap_n\}$$

where $ap_i \forall i \in [1, n]$ are sub-propositional formula of Γ and each ap_i contains only boolean *AND* of the attributes in $\mathbb{U}_\mathbb{A}$. That is,

$$ap_{p_i} = \{Att_{p1} \ AND \ Att_{p2} \ AND \ ... \ AND \ Att_{pi'}\}$$

where $Att_{pj} \in \mathbb{U}_\mathbb{A}$ and $1 \leq j \leq i'$.

Then, the Owner computes Q_i', Q_i and ω_i for all $i \in [1, n]$ as:

$$Q_i = g_1^{i' \cdot \mu} \cdot \prod_{j=1}^{i'} T_{ij} = g_1^{(i' \cdot \mu + \sum_{j=1}^{i'} t_{ij})}$$

$$Q_i' = \prod_{j=1}^{i'} T_{ij}' = g_2^{\sum_{j=1}^{i'} t_{ij}}$$

$$\omega_i' = \hat{e}(Q_i, Q_i') = \hat{e}(g_1, g_2)^{\left(i' \mu + \sum_{j=1}^{i'} t_{ij}\right) \cdot \left(\sum_{j=1}^{i'} t_{ij}\right)}$$

$$\omega_i = H(\omega_i')$$

[9]The secure-channel can be established using Secure Socket Layer (SSL).

The Owner then chooses a random number $y \in \mathbb{Z}_q^*$ and computes a polynomial $P(x)$ using SIFF, described in Section 4.

$$P(x) = x^n + a_1 x^{(n-1)} + ... + a_i x^{(n-i)} + ... + a_n$$

where ω_i for all $i \in [1, n]$ are the n solutions of the equation $P(x) - y = 0$. Finally, the Owner chooses a keyword set $\mathbb{W}$ of a file and encrypts each keyword $w_k \in \mathbb{W}$ as follows:

i) chooses a random number $r \in \mathbb{Z}_q^*$ and computes C_k, C_1, and C_2 as follows:

$$C_k = \left((g_1^{\varrho})^{H_1(w_k)} \cdot g_1^{\upsilon \cdot r_o}\right)^{r \cdot y} = g_1^{\rho \cdot H_1(w_k) \cdot r \cdot y} \cdot g_1^{\upsilon \cdot r_o \cdot r \cdot y}$$
$$C_1 = (g_2^{\mu})^r = g_2^{\mu \cdot r}$$
$$C_2 = (g_2^{r_o})^r = g_2^{r_o \cdot r}$$

ii) computes all the coefficients of $P(x)$, $\{Co[j] = a_j\}_{(\forall j \in [1,n])}$.

The final encrypted index is $\mathbb{CT} = \left\langle \{C_k\}_{\{\forall w_k \in \mathbb{W}\}}, C_1, C_2, \{Co[j]\}_{(\forall j \in [1,n])}, \Gamma \right\rangle$. Finally, the Owner sends the encrypted index $\mathbb{CT}$ along with the encrypted file to the CSP.

5.2.4 *Key Generation*

This phase is run by the AA to generate secret keys and public keys for the users. It consists of KEYGEN algorithm.

– **KEYGEN** $((\mathtt{SK}_{\mathtt{ID_u}}, \mathtt{Pub}_{\mathtt{ID_u}}) \leftarrow (\mathtt{MSK}, ID_u, \mathbb{S}_{ID_u}))$: in this algorithm, the master secret MSK, user's unique identity ID_u and the attribute set $\mathbb{S}_{ID_u}$ of the user are taken as input. It outputs a secret key $\mathtt{SK}_{\mathtt{ID_u}}$ and a public key $\mathtt{Pub}_{\mathtt{ID_u}}$ for the user ID_u.

Let, $\mathbb{S}_{ID_u} = \{Att_{p1}, Att_{p2}, ..., Att_{ps'}\}$ be the set attributes possess by the user ID_u. AA computes the secret key $\mathtt{SK}_{\mathtt{ID_u}} = \left\langle \{\mathtt{D_{pi}}, \mathtt{D'_{pi}}\}_{\forall i \in [1,s']}, \mathtt{Priv}^1_{\mathtt{ID_u}}, \mathtt{Priv}^2_{\mathtt{ID_u}}, \mathtt{AW}_{\mathbb{S}_{\mathtt{ID_u}}} \right\rangle$ and public key $\mathtt{Pub}_{\mathtt{ID_u}}$ for the user ID_u as follows: let $(b, u) \in \mathbb{Z}_q^{*2}$ be two random numbers and suppose $\mathtt{Q}_{\mathtt{ID_u}} = [H_1(ID_u) + u]$

i) $\mathtt{D_{pi}} = g_1^{(\mu + t_{pi})\mu \cdot b} \cdot g_1^{\varphi \cdot a_i}$
ii) $\mathtt{D'_{pi}} = g_2^{\frac{t_{pi}}{\mu \cdot b}}$
iii) $\mathtt{Priv}^1_{\mathtt{ID_u}} = g_1^{\frac{\rho \cdot \mathtt{Q}_{\mathtt{ID_u}}}{\mu}}$
iii) $\mathtt{Priv}^2_{\mathtt{ID_u}} = g_1^{\upsilon \cdot \mathtt{Q}_{\mathtt{ID_u}}}$
v) $\mathtt{Pub}_{\mathtt{ID_u}} = g_2^{\mathtt{Q}_{\mathtt{ID_u}}}$
vi) $\mathtt{AW}_{\mathbb{S}_{\mathtt{ID_u}}} = g_2^{\prod\limits_{i \in (\mathbb{U}_{\mathbb{A}} \setminus \mathbb{S}_{ID_u})} [a_i + \theta \cdot H_1(ID_u)]}$

Afterwards, AA sends the secret keys $\mathtt{SK}_{\mathtt{ID_u}}$ to the user ID_u using a secure-channel and publishes the public key $\mathtt{Pub}_{\mathtt{ID_u}}$ in its public bulletin board.

5.2.5 *Keyword Search*

This phase begins when a user wants to retrieve relevant encrypted files of a keyword from the CSP. It includes two sub-phases, namely, *Trapdoor Generation* and *Search*, detailed below.

5.2.5.1 ***Trapdoor Generation***: in this phase, a user generates trapdoors for the keywords of his/her choice to delegate search capabilities to the CSP. This phase consists of a TRAPGEN algorithm.

– **TRAPGEN** $(\mathtt{Trap} \leftarrow (w_k, \mathtt{SK}_{\mathtt{ID_u}}, \mathtt{Pub}_{\mathtt{ID_u}}))$: this algorithm takes the public key $\mathtt{Pub}_{\mathtt{ID_u}}$, secret key $\mathtt{SK}_{\mathtt{ID_u}} = \left\langle \{\mathtt{D_{pi}}, \mathtt{D'_{pi}}\}_{(\forall i \in [1,s'])}, \mathtt{Priv}^1_{\mathtt{ID_u}}, \mathtt{Priv}^2_{\mathtt{ID_u}}, \mathtt{AW}_{\mathbb{S}_{\mathtt{ID_u}}} \right\rangle$ of the user ID_u and a keyword w_k as input, and outputs a trapdoor Trap to perform keyword search. Before generating the trapdoor Trap, the user and the CSP participate in an *interactive protocol*. In this protocol, the user needs to interact with the CSP for sending the trapdoor to the CSP with no use of secure-channels. The procedure for the interactive protocol is given below.

i) the user ID_u chooses a random number $d \in \mathbb{Z}_q^*$ for the current session. The user ID_u computes g_1^d and sends g_1^d and identity ID_u to the CSP. The user ID_u temporarily keeps d in his/her database.
ii) after receiving the request from the user ID_u, CSP chooses a random number n_c for current session. It computes $(\mathtt{Pub}_{\mathtt{ID_u}})^{\frac{1}{n_c}}$ and $g_1^{\varrho \cdot n_c}$, where $\mathtt{Pub}_{\mathtt{ID_u}}$ is the public key of the user ID_u which is available from the AA's public bulletin board and ϱ is the secret key of the CSP. The CSP temporarily keeps g_1^d and n_c in its database, and sends $(\mathtt{Pub}_{\mathtt{ID_u}})^{\frac{1}{n_c}}$ and $g_1^{\varrho \cdot n_c}$ to the user ID_u.

Upon receiving the tuple $\left\langle (\mathtt{Pub}_{\mathtt{ID_u}})^{\frac{1}{n_c}}, g_1^{\varrho \cdot n_c} \right\rangle$ from the CSP, the user ID_u checks validity of the tuple by comparing $\hat{e}(g_1^{\varrho}, \mathtt{Pub}_{\mathtt{ID_u}}) \stackrel{?}{=} \hat{e}(g_1^{\varrho \cdot n_c}, (\mathtt{Pub}_{\mathtt{ID_u}})^{\frac{1}{n_c}})$, where g_1^{ϱ} is a public parameter. If the tuple is valid, the user chooses a random number $d' \in \mathbb{Z}_q^*$ and then generates trapdoor $\mathtt{Trap} = \left\langle \{tr_{1i}, tr_{2i}\}_{\forall i \in [1,s']}, tr_3, tr_4, tr_5, tr_6 \right\rangle$ as follows:

i) $tr_{1i} = (\mathtt{D_{pi}} \cdot g_1^{\varrho \cdot n_c})^d = g_1^{(\mu + t_{pi})\mu \cdot b \cdot d} \cdot g_1^{\varphi \cdot d \cdot a_i} \cdot g_1^{\varrho \cdot n_c \cdot d}$
ii) $tr_{2i} = \mathtt{D'_{pi}}^{1/d} = g_2^{\frac{ts_i}{\mu \cdot b \cdot d}}$
iii) $tr_3 = (\mathtt{Priv}^1_{\mathtt{ID_u}})^{H_1(w_k) \cdot d \cdot d'} = g_1^{\frac{\rho \cdot \mathtt{Q}_{\mathtt{ID_u}} \cdot H_1(w_k) \cdot d \cdot d'}{\mu}}$
iv) $tr_4 = \left(\mathtt{Priv}^2_{\mathtt{ID_u}}\right)^{d \cdot d'} = g_1^{\upsilon \cdot \mathtt{Q}_{\mathtt{ID_u}} \cdot d \cdot d'}$
v) $tr_5 = \left((\mathtt{Pub}_{\mathtt{ID_u}})^{\frac{1}{n_c}}\right)^{d \cdot d'} = g_2^{\frac{\mathtt{Q}_{\mathtt{ID_u}} \cdot d \cdot d'}{n_c}}$
vi) $tr_6 = (\mathtt{AW}_{\mathbb{S}_{\mathtt{ID_u}}})^{1/d} = g_2^{\prod\limits_{i \in (\mathbb{U}_{\mathbb{A}} \setminus \mathbb{S}_{ID_u})} (a_i + \theta \cdot H_1(ID_u))/d}$

The user ID_u sends the trapdoor $\mathtt{Trap} = \left\langle \{tr_{1i}, tr_{2i}\}_{\forall i \in [1,s']}, tr_3, tr_4, tr_5, tr_6 \right\rangle$, $\mathbb{S}_{ID_u}$, and ID_u to the CSP.

5.2.5.2 ***Search***: in this phase, CSP performs keyword search over the encrypted indexes using the trapdoor Trap received from the user and finally returns the (encrypted) files associated with the keywords that has match with the trapdoor Trap. This phase is composed of ATTVERI and KEYSEARCH algorithms. The ATTVERI algorithm is run by the VA to check validity the users' attributes which they possess; while the KEYSEARCH algorithm is run by the CSP to perform keyword search if and only if the VA successfully validates the requesting user's attributes in the ATTVERI algorithm. It implies that the CSP needs to contact the VA for validity checking of user's attributes. Details of the two algorithms are presented below.

– **ATTVERI**$((\{(g_1^d)^{\varphi \cdot a_i}\}_{\forall i \in \mathbb{S}_{ID_u}} / \perp) \leftarrow (\mathtt{MK}, tr_6, g_1^d, \mathbb{S}_{ID_u}, ID_u))$: in this algorithm, VA takes tr_6, g_1^d, $\mathbb{S}_{ID_u}$, and ID_u as inputs, where $tr_6, g_1^d, \mathbb{S}_{ID_u}$, and ID_u are sent by the CSP and MK is sent by the AA. It outputs

$\{(g_1^d)^{\varphi \cdot a_i}\}_{\forall i \in \mathbb{S}_{ID_u}}$ or $\perp$ based on the following comparison: the VA compares

$$\hat{e}\left(\left(g_1^d\right)^{\prod\limits_{i\in\mathbb{S}_{ID_u}}(a_i+\theta\cdot H_1(ID_u))}, tr_6\right) \stackrel{?}{=} Acc_{\mathbb{U}_\mathbb{A}},$$

$$\text{where } Acc_{ID_u} = \hat{e}\left(g_1^{\prod\limits_{i\in\mathbb{U}_\mathbb{A}}(a_i+\theta\cdot H_1(ID_u))}, g_2\right)$$

If both are equal then the VA returns $\{(g_1^d)^{\varphi \cdot a_i}\}_{\forall i \in \mathbb{S}_{ID_u}}$ to the CSP which implies that the attribute set $\mathbb{S}_{ID_u}$ is valid. Note that Acc_{ID_u} can be precomputed by the VA.

Proof of consistency:

$$\begin{aligned}
&\hat{e}\left(\left(g_1^d\right)^{\prod\limits_{i\in\mathbb{S}_{ID_u}}(a_i+\theta\cdot H_1(ID_u))}, tr_6\right)\\
&= \hat{e}\left(g_1^{d[\prod\limits_{i\in\mathbb{S}_{ID_u}}(a_i+\theta\cdot H_1(ID_u))]}, g_2^{\prod\limits_{i\in(\mathbb{U}_\mathbb{A}\setminus\mathbb{S}_{ID_u})}(a_i\,+\,\theta\,\cdot\,H_1(ID_u))/d}\right)\\
&= \hat{e}\left(g_1, g_2\right)^{\prod\limits_{i\in\mathbb{U}_\mathbb{A}}(a_i+\theta\cdot H_1(ID_u))}\\
&= \hat{e}(g_1^{\prod\limits_{i\in\mathbb{U}_\mathbb{A}}(a_i+\theta\cdot H_1(ID_u))}, g_2)\\
&= Acc_{ID_u}
\end{aligned}$$

Otherwise VA returns $\perp$ to the CSP which implies that the attribute set $\mathbb{S}_{ID_u}$ is not valid (i.e. the attribute set $\mathbb{S}_{ID_u}$ contains one or more revoked or invalid attributes).

– **KeySearch** $(Files \leftarrow (\{(g_1^d)^{\varphi \cdot a_i}\}_{\forall i \in \mathbb{S}_{ID_u}} / \perp, \mathtt{Trap}, \mathbb{S}_{ID_u}, \mathbb{CT}))$: in this algorithm, if the CSP receives $\perp$ from the VA, the CSP aborts the connection for the user ID_u. Otherwise, if the CSP receives $\{(g_1^d)^{\varphi \cdot a_i}\}_{\forall i \in \mathbb{S}_{ID_u}}$ from the VA for the user ID_u, it performs the keyword search over the encrypted index $\mathbb{CT}$ which access policy is satisfied by the user's attribute set $\mathbb{S}_{ID_u}$. For the keyword search, the CSP uses the trapdoor $\mathtt{Trap} = \langle\{tr_{1i}, tr_{2i}\}_{\forall i\in[1,s']}, tr_3, tr_4, tr_5, tr_6\rangle$ and if the trapdoor matches with some keywords then it returns the files associated with the keywords to the user. Let $\mathbb{CT} = \langle\{C_k\}_{\{\forall w_k \in \mathbb{W}\}}, C_1, C_2, \{Co[j]\}_{(\forall j\in[1,n])}, \Gamma\rangle$ be the index of the keyword set $\mathbb{W}$. CSP performs the keyword search as follows.

Let $ap_{p_{s'}} = \{Att_{p1}\ AND\ Att_{p2}\ AND\ ...\ AND\ Att_{ps'}\}$ be the sub-access policy, where $ap_{p_{s'}} \subseteq \Gamma$, which is satisfied by the user's attribute set $\mathbb{S}_{ID_u}$. The CSP computes $(g_1^d)^{\varrho \cdot n_c} = g_1^{\varrho \cdot n_c \cdot d}$ and then computes $X, Y, w'_s, value, y$, V_1, V_2 and V_3 as follows:

i) $X = g_1^{(s'\mu + \sum\limits_{i=1}^{s'} t_{pi})\mu \cdot b \cdot d}$, where

$$\begin{aligned}
X &= \frac{\prod\limits_{i\in ap_{p_{s'}}} tr_{1i}}{\prod\limits_{i\in ap_{p_{s'}}} (g_1^d)^{\varphi\cdot a_i}\cdot g_1^{\varrho\cdot n_c\cdot d}}\\
&= \frac{g_1^{(s'\mu+\sum\limits_{i=1}^{s'} t_{pi})\mu\cdot b\cdot d}\cdot g_1^{\varphi\cdot d\sum\limits_{i=1}^{s'} a_i}\cdot g_1^{\varrho\cdot n_c\cdot d}}{g_1^{\varphi\cdot d\sum\limits_{i=1}^{s'} a_i}\cdot g_1^{\varrho\cdot n_c\cdot d}}\\
&= g_1^{(s'\mu+\sum\limits_{i=1}^{s'} t_{pi})\mu\cdot b\cdot d}
\end{aligned}$$

ii) $Y = \prod\limits_{i\in ap_{p_{s'}}} tr_{2i} = g_2^{\frac{\sum\limits_{i=1}^{s'} t_{pi}}{\mu\cdot b\cdot d}}$

iii) $\omega'_s = \hat{e}(X, Y) = \hat{e}\,(g_1, g_2)^{(s'\mu + \sum\limits_{j=1}^{s'} t_{pj})(\sum\limits_{j=1}^{s'} t_{pj})}$

iv) $value = H(\omega'_s)$

v) $y = P(value)$. It is to be noted that, if $\exists ap_j \in \mathbb{AP}$ such that $ap_j = \{Att_{p1}\ AND\ Att_{p2}\ AND\ ...\ AND\ Att_{ps'}\}$ then $value$ is one of the solutions of $P(x) - y = 0$. Hence, $y = P(value)$.

vi) computes $z = \frac{y}{n_c}$. Note that n_c is known to the CSP.

vii) computes V_1, where

$$\begin{aligned}
V_1 = \hat{e}\,((tr_3)^z, C_1) &= \hat{e}\left(\left(g_1^{\frac{\rho\cdot \mathtt{Q}_{\mathtt{ID_u}}\cdot H_1(w_k)\cdot d\cdot d'}{\mu}}\right)^{\frac{y}{n_c}}, g_2^{\mu\cdot r}\right)\\
&= \hat{e}(g_1, g_2)^{\frac{\rho\cdot \mathtt{Q}_{\mathtt{ID_u}}\cdot H_1(w_k)\cdot d\cdot d'\cdot y\cdot r}{n_c}}
\end{aligned}$$

viii) computes V_2, where

$$\begin{aligned}
V_2 = \hat{e}((tr_4)^z, C_2) &= \hat{e}\left(\left(g_1^{\upsilon\cdot \mathtt{Q}_{\mathtt{ID_u}}\cdot d\cdot d'}\right)^{\frac{y}{n_c}}, g_2^{r_o\cdot r}\right)\\
&= \hat{e}\,(g_1, g_2)^{\frac{\upsilon\cdot \mathtt{Q}_{\mathtt{ID_u}}\cdot d\cdot d'\cdot r_o\cdot r\cdot y}{n_c}}
\end{aligned}$$

ix) computes V_3, where

$$\begin{aligned}
V_3 &= \hat{e}(C_k, tr_5)\\
&= \hat{e}\left(g_1^{\rho\cdot H_1(w_k)\cdot r\cdot y}\cdot g_1^{\upsilon\cdot r_o\cdot r\cdot y}, g_2^{\frac{\mathtt{Q}_{\mathtt{ID_u}}\cdot d\cdot d'}{n_c}}\right)\\
&= \hat{e}\,(g_1, g_2)^{[\rho\cdot H_1(w_k)\cdot r\cdot y + \upsilon\cdot r_o\cdot r\cdot y]\cdot\frac{\mathtt{Q}_{\mathtt{ID_u}}\cdot d\cdot d'}{n_c}}\\
&= \hat{e}\,(g_1, g_2)^{\frac{\rho\cdot H_1(w_k)\cdot r\cdot y\cdot \mathtt{Q}_{\mathtt{ID_u}}\cdot d\cdot d'}{n_c} + \frac{\upsilon\cdot r_o\cdot r\cdot y\cdot \mathtt{Q}_{\mathtt{ID_u}}\cdot d\cdot d'}{n_c}}\\
&= \hat{e}\,(g_1, g_2)^{\frac{\rho\cdot \mathtt{Q}_{\mathtt{ID_u}}\cdot H_1(w_k)\cdot d\cdot d'\cdot y\cdot r}{n_c}}\cdot \hat{e}(g_1, g_2)^{\frac{\upsilon\cdot \mathtt{Q}_{\mathtt{ID_u}}\cdot d\cdot d'\cdot r_o\cdot r\cdot y}{n_c}}
\end{aligned}$$

Now, the CSP checks $V_3 \stackrel{?}{=} V_1 \cdot V_2$, if equals then CSP returns the file associated with the keyword w_k.

Proof of consistency:

$$\begin{aligned}
V_1 \cdot V_2 &= \hat{e}\,(g_1, g_2)^{\frac{\rho\cdot \mathtt{Q}_{\mathtt{ID_u}}\cdot H_1(w_k)\cdot d\cdot d'\cdot y\cdot r}{n_c}}\cdot \hat{e}(g_1, g_2)^{\frac{\upsilon\cdot \mathtt{Q}_{\mathtt{ID_u}}\cdot d\cdot d'\cdot r_o\cdot r\cdot y}{n_c}}\\
&= V_3
\end{aligned}$$

5.3 Conjunctive Keyword Search

In practical use cases, a user may want to access files which contain several intended keywords using one search request, which is known as *Conjunctive Keyword Search*. The proposed scheme can easily perform conjunctive keyword search. For this purpose, the user needs to construct tr_3 component of the trapdoor $trap$ as follows: $tr_3 = \left(\mathtt{Priv}^1_{\mathtt{ID_u}}\right)^{\sum H_1(w_k)\cdot d\cdot d'}$ in TrapGen algorithm. Accordingly, the CSP needs to compute $V_3 = \hat{e}\left(\prod C_k, tr_5\right)$ and $V_2 = \hat{e}\left(tr_4^{(z\cdot|\sum H_1(w_k)|)}, C_2\right)$ in the KeySearch algorithm, where $|\sum H_1(w_k)|$ is the number of keywords in the conjunctive keyword search. Thus, the proposed scheme achieves conjunctive keyword search without incurring additional overhead on the system, which is of high interest.

5.4 Revocation

Sometimes, it is essential to revoke a user (or attributes of a user) from the system so that he/she can no longer perform keyword search (using his/her revoked attributes). The proposed scheme can revoke a user in two ways: first, AA can revoke a user by removing the revoked user's (say the revoked user be ID'_u) public key $\texttt{Pub}_{\texttt{ID}'_\texttt{u}}$ from its public bulletin board and second, updating the attribute witness of the non-revoked users as well as the revoked user by the AA. It can be observed that the former revocation mechanism can be easily done by the AA. The latter revocation mechanism, where attributes of a user are revoked, is explained below:

Let the revoked attribute be Att_i. AA chooses a fresh random number $a'_i \in \mathbb{Z}_q^*$ for the revoked attribute Att_i and computes a fresh attribute witness $\texttt{AW}'_{\mathbb{S}_{\texttt{ID}_\texttt{u}}}$ for each non-revoked user ID_u who holds the revoked attribute Att_i, where $\texttt{AW}'_{\mathbb{S}_{\texttt{ID}_\texttt{u}}} = \left(g_2^{\prod_{j \in (\mathbb{U}_\mathbb{A} \setminus \mathbb{S}_{ID_u})} (a_j + \theta \cdot H_1(ID_u))} \right)^{\left(\frac{a'_i + \theta \cdot H_1(ID_u)}{a_i + \theta \cdot H_1(ID_u)}\right)}$.

It is to be noted that a_i is the old random number associated with the attribute Att_i. Also, AA shares the fresh a'_i with VA and VA replaces the old a_i related with the attribute Att_i with the fresh a'_i. It is worth to mention that our revocation mechanism avoids re-encryption of all the encrypted indexes related with the revoked attributes. Moreover, the revoked user ID'_u needs to update attribute witness from the AA to perform further keyword search using his/her non-revoked attributes. The updated attribute witness of the revoked user ID'_u is $\texttt{AW}'_{\mathbb{S}_{\texttt{ID}'_\texttt{u}}}$, where

$$\texttt{AW}'_{\mathbb{S}_{\texttt{ID}'_\texttt{u}}} = \left(g_2^{\prod_{j \in (\mathbb{U}_\mathbb{A} \setminus \mathbb{S}_{ID'_u})} (a_j + \theta \cdot H_1(ID'_u))} \right)^{\left(\frac{1}{a_i + \theta \cdot H_1(ID'_u)}\right)}$$

6 Security Analysis

This section presents the security analysis of the proposed scheme, with respect to security requirements detailed in Section 3.3.

It has been observed that the users are authorized based on the *attribute witness* they possess. The *attribute witness* is based on the concept of bilinear-map accumulator. The VA uses the trapdoor component tr_6, associated to the *attribute witness*, to perform user authorization. As explained in Section 4.3, the security of the accumulator is based on the *q-SBDH* assumption. The security proof of accumulators can be found in [51], [52].

6.1 Keyword Semantic Security

Theorem 1 and its proof demonstrate that the proposed scheme is IND-sCP-CKA secure under the standard model.

Theorem 1. *If a probabilistic polynomial-time adversary $\mathcal{A}_1$ can win the IND-sCP-CKA game with non-negligible advantage ϵ, then a PPT simulator $\mathcal{B}$ can be constructed to break the MXDH assumption with non-negligible advantage $\frac{\epsilon}{2}$.*

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

The MXDH challenger $\mathcal{C}$ chooses random numbers $(\mathbb{x}, \mathbb{y}, \mathbb{r}) \in \mathbb{Z}_q^{*3}$ and flips a random coin $l \in \{0,1\}$. It sets $Z = g_1^{\mathbb{x} \cdot \mathbb{y}}$ if $l = 0$ and $Z = g_1^{\mathbb{r}}$ otherwise. Afterwards the challenger $\mathcal{C}$ sends the tuple $\{g_1, A = g_1^{\mathbb{x}}, B = g_1^{\mathbb{y}}, C = g_2^{\mathbb{x}}, D = g_2^{\mathbb{y}}, Z\}$ to the simulator $\mathcal{B}$ and asks the simulator $\mathcal{B}$ to output l.

In the IND-sCP-CKA game, the simulator $\mathcal{B}$ interacts with the adversary $\mathcal{A}_1$ which is defined as follows (simulator $\mathcal{B}$ acts as a challenger in the rest of the security game).

- Init The adversary $\mathcal{A}_1$ sends an access policy Γ^* in a form of propositional formula to the simulator $\mathcal{B}$. The simulator $\mathcal{B}$ easily converts the propositional formula Γ^* into the SOP form, where each sub-formula (or sub-access policy) is represented as ∂.

– Setup The simulator $\mathcal{B}$ chooses random numbers $(\psi, \sigma, \xi, \zeta, \kappa, \{\phi_i, \eta_i\}_{\forall i \in \mathbb{U}_\mathbb{A}}) \in \mathbb{Z}_q^{*(5+2|\mathbb{U}_\mathbb{A}|)}$. It also chooses two collision-resistant hash functions H and H_1. Simulator $\mathcal{B}$ computes $A^\sigma = g_1^{\sigma'}$ and $C^\sigma = g_2^{\sigma'}$, where $\sigma' = \mathbb{x} \cdot \sigma$. It also computes $\{B^{\phi_i} = g_1^{\phi'_i}, D^{\phi_i} = g_2^{\phi'_i}\}_{\forall i \in \mathbb{U}_\mathbb{A}}$, where $\{\phi'_i = \mathbb{y} \cdot \phi_i\}_{\forall i \in \mathbb{U}_\mathbb{A}}$. The simulator $\mathcal{B}$ outputs the public parameters $\texttt{PP} = \langle q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbb{Z}_q^*, H, H_1, \hat{e}, g_1, A, g_1^{\sigma'}, g_2^{\sigma'}, g_1^{\zeta}, \{g_1^{\phi'_i}, g_2^{\phi'_i}\}_{\forall i \in \mathbb{U}_\mathbb{A}} \rangle$. It sends PP along with ξ, ζ and $\{\eta_i\}_{\mathbb{U}_\mathbb{A}}$ to the adversary $\mathcal{A}_1$ and keeps master secret $\texttt{MSK} = \langle \psi, \sigma, \kappa, \{\phi_i\}_{\forall i \in \mathbb{U}_\mathbb{A}} \rangle$ in a secure place.

– Phase 1 Adversary $\mathcal{A}_1$ sends the following queries to the simulator $\mathcal{B}$:

Interactive protocol queries: Adversary $\mathcal{A}_1$ sends a request along with the challenged identity ID_j for the interactive protocol parameters to the simulator $\mathcal{B}$. Simulator $\mathcal{B}$ chooses random numbers $(u, r_j, r''_j) \in \mathbb{Z}_q^{*3}$ and then sets $h_{id_j} = H_1(ID_j)$. Afterwards, the simulator $\mathcal{B}$ computes the interactive protocol parameters $C^{h_{id_j} + u/r_j} = g_2^{\mathbb{x}(h_{id_j} + u)/r_j}$ and $g_1^{\zeta \cdot r_j \cdot r''_j}$. It then sends the interactive protocol parameters $g_2^{\mathbb{x}(h_{id_j} + u)/r_j}, g_1^{\zeta \cdot r_j \cdot r''_j}$ and r_j to the adversary $\mathcal{A}_1$.

Trapdoor queries: Without loss of generality, adversary $\mathcal{A}_1$ submits a keyword $w_j \in \mathbb{W}$ and an attribute set $\mathbb{S}_j$.

Simulator $\mathcal{B}$ computes $h_{w_j} = H_1(w_j)$. It chooses random numbers $(r'_j, r'''_j) \in \mathbb{Z}_q^{*2}$. The simulator $\mathcal{B}$ computes trapdoor $\texttt{Trap}_j = \langle \{tr_{1i_j}, tr_{2i_j}\}_{\forall i \in \mathbb{S}_j}, tr_{3_j}, tr_{4_j}, tr_{5_j}, tr_{6_j} \rangle$ as follows:

$$tr_{1i_j} = \left((A^\sigma \cdot B^{\phi_i})^{\sigma \cdot r'_j} \cdot (g_1^{\eta_i})^{\xi}\right)^{r''_j} \cdot g_1^{\zeta \cdot r_j \cdot r''_j}$$
$$= g_1^{(\sigma' + \phi'_i)\frac{\sigma'}{\mathbb{x}} \cdot r'_j \cdot r''_j + \xi \cdot \eta_i \cdot r''_j + \zeta \cdot r_j \cdot r''_j}$$

$$tr_{2i_j} = \left((D^{\phi_i})^{\frac{1}{\sigma \cdot r'_j}}\right)^{\frac{1}{r''_j}} = g_2^{\frac{\phi'_i \cdot \mathbb{x}}{\sigma' \cdot r'_j \cdot r''_j}}$$

$$tr_{3_j} = A^{\frac{(h_{id_j} + u) h_{w_j} \cdot r''_j \cdot r'''_j}{\sigma}} = g_1^{\frac{\mathbb{x} \cdot \mathbb{x}(h_{id_j} + u) h_{w_j} \cdot r''_j \cdot r'''_j}{\sigma'}}$$

$$tr_{4_j} = A^{(h_{id_j} + u)\psi \cdot r''_j \cdot r'''_j} = g_1^{\psi \cdot \mathbb{x}(h_{id_j} + u) r''_j \cdot r'''_j}$$

$$tr_{5_j} = \left(C^{(h_{id_j} + u)/r_j}\right)^{r''_j \cdot r'''_j} = g_2^{\mathbb{x}(h_{id_j} + u) r''_j \cdot r'''_j / r_j}$$

$$tr_{6_j} = C^{\prod_{i \in (\mathbb{U}_\mathbb{A} \setminus \mathbb{S}_j)} (\eta_i + \kappa \cdot h_{id_j})/r''_j} = g_2^{\mathbb{x} \prod_{i \in (\mathbb{U}_\mathbb{A} \setminus \mathbb{S}_j)} (\eta_i + \kappa \cdot h_{id_j})/r''_j}$$

Finally, simulator $\mathcal{B}$ sends the $trap_j$ to the adversary $\mathcal{A}_1$.

- Challenge Upon receiving the challenged keywords w_0 and w_1 from the adversary $\mathcal{A}_1$, the simulator $\mathcal{B}$ flips a random coin $b \in \{0,1\}$ and encrypts w_b with the challenged

access policy Γ^*. The simulator $\mathcal{B}$ computes $h_{w_b} = H_1(w_b)$ and chooses a random number $r_k \in \mathbb{Z}_q^*$.

For each sub-access policy (or sub-access formula) $\partial \in \Gamma^*$, the simulator $\mathcal{B}$ does the following computations: let $|\partial|$ be the number of attributes associated with the sub-access policy ∂.

$$\Phi_1 = (A^{\sigma})^{|\partial|} \prod_{i=1}^{|\partial|} B^{\phi_i} = g_1^{\sigma' \cdot |\partial| + \sum_{i=1}^{|\partial|} \phi_i'}; \Phi_2 = \prod_{i=1}^{|\partial|} D^{\phi_i} = g_2^{\sum_{i=1}^{|\partial|} \phi_i'}$$
$$\vartheta_{\partial} = H(\hat{e}(\Phi_1, \Phi_2))$$

The simulator $\mathcal{B}$ chooses random numbers $(\nu, a_1, a_2, ..., a_n) \in \mathbb{Z}_q^{*(n+1)}$, where n is the number of sub-access policies ∂ in the access policy Γ^*. The simulator $\mathcal{B}$ computes a polynomial $\delta_k(x)$ using all the values of ϑ_{∂} for each sub-access policy such that $\delta_k(x) - \nu = 0$.

Finally, the simulator $\mathcal{B}$ chooses a random number $r_c \in \mathbb{Z}_q^*$ and sends the encrypted index $\mathbb{CT}_{w_b} = \langle C_{w_b}, C_1, C_2, Co_{w_b}[k]_{\forall k \in \{1,n\}} \rangle$ of the keyword w_b to the adversary $\mathcal{A}_1$, where $C_{w_b} = Z^{h_{w_b} \cdot \nu} \cdot B^{\psi \cdot \cdot r_c \cdot \nu}$, $C_1 = D^{\sigma} = D^{\frac{\sigma'}{\mathbb{x}}}$, $C_2 = D^{r_c}$ and $Co_{w_b}[k]_{\forall k \in [1,n]}$ are the coefficients of the equation $\delta_k(x) - \nu = 0$.

- PHASE 2 Same as **PHASE 1**.

- GUESS The adversary $\mathcal{A}_1$ guesses a bit b' and sends to the simulator $\mathcal{B}$. If $b' = b$ then the adversary $\mathcal{A}_1$ wins IND-sCP-CKA game; otherwise it fails. If $b' = b$, simulator $\mathcal{B}$ answers "MXDH" in the game (i.e. outputs $l = 0$); otherwise $\mathcal{B}$ answers "random" (i.e. outputs $l = 1$).

If $Z = g_1^{\mathbb{r}}$; then C_{w_b} is completely random from the view of the adversary $\mathcal{A}_1$. So, the received encrypted index $\mathbb{CT}_{w_b}$ is not compliant to the game (i.e. invalid encrypted index). Therefore, the adversary $\mathcal{A}_1$ chooses b' randomly. Hence, probability of the adversary $\mathcal{A}_1$ for outputting $b' = b$ is $\frac{1}{2}$.

If $Z = g_1^{\mathbb{x} \cdot \mathbb{y}}$, then adversary $\mathcal{A}_1$ receives a valid encrypted index. The adversary $\mathcal{A}_1$ knowing more than expected in the IND-sCP-CKA game (random parameters being set by the challenger to some predefined values), wins the IND-sCP-CKA game with non-negligible advantage ϵ (according to the Theorem 1). So, the probability of outputting $b' = b$ for the adversary $\mathcal{A}_1$ is $\frac{1}{2} + \epsilon$, where probability ϵ is for guessing that the received encrypted index is valid and probability $\frac{1}{2}$ is for guessing whether the valid encrypted index $\mathbb{CT}_{w_b}$ is related to w_0 or w_1.

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

6.2 Keyword Guessing Attack

In this section, we prove Theorem 2, thus proving resistance of our scheme to the keyword guessing attack.

Theorem 2. *If a probabilistic polynomial-time adversary $\mathcal{A}_2$ (in our case any outsider adversary) can win the IND-O-KGA game with non-negligible advantage ϵ, then a PPT simulator $\mathcal{B}$ can be constructed to break the MXDH assumption with non-negligible advantage $\frac{\epsilon}{2}$.*

Proof. Assume that an adversary $\mathcal{A}_2$ with an advantage ϵ breaks the proposed scheme. Then a simulator $\mathcal{B}$ can be constructed to solve the MXDH problem with an advantage $\frac{\epsilon}{2}$.

The MXDH challenger $\mathcal{C}$ chooses random numbers $(\mathbb{a}, \mathbb{b}, \mathbb{r}) \in \mathbb{Z}_q^{*3}$ and flips a random coin $l \in \{0, 1\}$. It sets $Z = g_1^{\mathbb{a}\mathbb{b}}$ if $l = 0$ and $Z = g_1^{r}$ otherwise. Afterwards, the challenger $\mathcal{C}$ sends the tuple $\{g_1, A = g_1^{\mathbb{a}}, B = g_1^{\mathbb{b}}, C = g_2^{\mathbb{a}}, D = g_2^{\mathbb{b}}, Z\}$ to the simulator $\mathcal{B}$ and asks to output l. Now the simulator $\mathcal{B}$ acts as a challenger in the rest of the security game and interacts with the adversary $\mathcal{A}_2$.

– SETUP The simulator $\mathcal{B}$ chooses random numbers $(\psi, \sigma, \xi, \zeta, \kappa, \{\phi_i, \eta_i\}_{\forall i \in \mathbb{U}_{\mathbb{A}}}) \in \mathbb{Z}_q^{*(5+2|\mathbb{U}_{\mathbb{A}}|)}$. It also chooses two collision-resistant hash functions H and H_1. Simulator $\mathcal{B}$ computes $A^{\sigma} = g_1^{\sigma'}$ and $C^{\sigma} = g_2^{\sigma'}$, where $\sigma' = \mathbb{a} \cdot \sigma$. It also computes $\{B^{\phi_i} = g_1^{\phi_i'}, D^{\phi_i} = g_2^{\phi_i'}\}_{\forall i \in \mathbb{U}_{\mathbb{A}}}$, where $\{\phi_i' = \mathbb{b} \cdot \phi_i\}_{\forall i \in \mathbb{U}_{\mathbb{A}}}$. The simulator $\mathcal{B}$ outputs the public parameters $\mathtt{PP} = \langle q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbb{Z}_q^*, H, H_1, \hat{e}, g_1, A, g_1^{\sigma'}, g_2^{\sigma'}, g_1^{\zeta}, \{g_1^{\phi_i'}, g_2^{\phi_i'}\}_{\forall i \in \mathbb{U}_{\mathbb{A}}} \rangle$. It sends PP along with ξ and $\{\eta_i\}_{\mathbb{U}_{\mathbb{A}}}$ to the adversary $\mathcal{A}_1$ and keeps master secret $\mathtt{MSK} = \langle \psi, \zeta, \sigma, \kappa, \{\phi_i\}_{\forall i \in \mathbb{U}_{\mathbb{A}}} \rangle$ in a secure place.

– PHASE 1 The following queries are issued by the adversary $\mathcal{A}_2$:

Interactive protocol queries: Adversary $\mathcal{A}_1$ sends a request along with the challenged identity ID_j for the interactive protocol parameters to the simulator $\mathcal{B}$. Simulator $\mathcal{B}$ chooses random numbers $(u, r_j, r_j'') \in \mathbb{Z}_q^{*3}$ and then sets $h_{id_j} = H_1(ID_j)$. Afterwards, the simulator $\mathcal{B}$ computes the interactive protocol parameters $C^{(h_{id_j} + u)/r_j} = g_2^{\mathbb{a}(h_{id_j} + u)/r_j}$ and $g_1^{\zeta \cdot r_j \cdot r_j''}$. It then sends $g_2^{\mathbb{a}(h_{id_j} + u)/r_j}$ and $g_1^{\zeta \cdot r_j \cdot r_j''}$ to the adversary $\mathcal{A}_2$.

Trapdoor queries: Adversary $\mathcal{A}_2$ submits a keyword $w_j \in \mathbb{W}$. Simulator $\mathcal{B}$ generates a trapdoor using the attribute set $\mathbb{S}_j$.

Simulator $\mathcal{B}$ computes $h_{w_j} = H_1(w_j)$. It chooses random numbers $(x_j', x_j''') \in \mathbb{Z}_q^{*2}$. The simulator $\mathcal{B}$ computes trapdoor $\mathtt{Trap}_j = \langle \{tr_{1i_j}, tr_{2i_j}\}_{\forall i \in \mathbb{S}_j}, tr_{3_j}, tr_{4_j}, tr_{5_j}, tr_{6_j} \rangle$ as follows:

$$tr_{1i_j} = \left((A^{\sigma} \cdot B^{\phi_i})^{\sigma \cdot x_j'} \cdot (g_1^{\eta_i})^{\xi}\right)^{x_j''} \cdot g_1^{\zeta \cdot x_j \cdot x_j''}$$
$$= g_1^{(\sigma' + \phi_i')\frac{\sigma'}{\mathbb{a}} \cdot x_j' \cdot x_j'' + \xi \cdot \eta_i \cdot x_j'' + \zeta \cdot x_j \cdot x_j''}$$
$$tr_{2i_j} = \left((D^{\phi_i})^{\frac{1}{\sigma \cdot x_j'}}\right)^{\frac{1}{x_j''}} = g_2^{\frac{\phi_i' \cdot \mathbb{a}}{\sigma' \cdot x_j' \cdot x_j''}}$$
$$tr_{3_j} = A^{\frac{(h_{id_j} + u) h_{w_j} \cdot x_j'' \cdot x_j'''}{\sigma}} = g_1^{\frac{\mathbb{a} \cdot \mathbb{a}(h_{id_j} + u)(h_{w_j} \cdot x_j'' \cdot x_j''')}{\sigma'}}$$
$$tr_{4_j} = A^{\psi(h_{id_j} + u) x_j'' \cdot x_j'''} = g_1^{\psi \cdot \mathbb{a}(h_{id_j} + u) x_j'' \cdot x_j'''}$$
$$tr_{5_j} = C^{(h_{id_j} + u) x_j'' \cdot x_j'''/x_j} = g_2^{\mathbb{a}(h_{id_j} + u) x_j'' \cdot x_j'''/x_j}$$
$$tr_{6_j} = C^{\prod_{i \in (\mathbb{U}_{\mathbb{A}} \setminus \mathbb{S}_j)} (\eta_i + \kappa \cdot h_{id_j})/x_j''} = g_2^{\mathbb{a} \prod_{i \in (\mathbb{U}_{\mathbb{A}} \setminus \mathbb{S}_j)} (\eta_i + \kappa \cdot h_{id_j})/x_j''}$$

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

– CHALLENGE Adversary $\mathcal{A}_2$ sends two keywords w_0 and w_1, that he/she wishes to be challenged on. After receiving the challenged keywords, the simulator $\mathcal{B}$ flips a random coin $b \in \{0, 1\}$. It computes $h_{w_b} = H_1(w_b)$ and $h_{id_c} = H_1(ID_c)$. It then generates a trapdoor $\mathtt{Trap}_{w_b}^c = \langle \{tr_{1i}^c, tr_{2i}^c\}_{\forall i \in \mathbb{S}_j}, tr_3^c, tr_4^c, tr_5^c, tr_6^c \rangle$ as follows: simulator chooses random numbers $x_c, x_c', x_c'' \in \mathbb{Z}_q^*$.

TABLE 2: Computation Time (in Milliseconds) of Elementary Cryptographic Operations in MNT Curve

	Exponentiation			Pairing	Group multiplication			Hash
	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_T$		$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_T$	
Commodity Laptop PC	0.650	4.89	0.543	3.82	0.005	0.002	0.003	0.004
Workstation	0.437	3.376	0.389	2.665	0.003	0.001	0.0012	0.002

TABLE 3: NOTATIONS

Notation	Description
$\lvert\mathbb{G}_1\rvert, \lvert\mathbb{G}_2\rvert, \lvert\mathbb{G}_T\rvert$	size of an element in $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ respectively
$\lvert\mathbb{Z}_q^*\rvert$	size of an element in $\mathbb{Z}_q^*$
$T_{exp_{\mathbb{G}_1}}, T_{exp_{\mathbb{G}_2}}$, $T_{exp_{\mathbb{G}_T}}$	computation cost of one exponentiation operation on $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_\mathbb{T}$ elements respectively
T_p	computation cost of a pairing operation
$\lvert\mathbb{U}_\mathbb{A}\rvert$	total number of attribute in the system
n_{or}	number of "*OR*" gates in an access policy
n_t, n_{ei}	number of attributes associated with a trapdoor and encrypted index respectively
n_r	number of revoked attributes
n_k	total number of elements in all the keyword subsets as defined in [26]
n_a	total number of encrypted indexes associated with a revoked attribute
n_u	total number of users associated with a revoked attribute
$\mathcal{K}_c$	total number of keywords associated with an encrypted index
$\mathcal{K}_a$	total number of keywords associated with an access structure as defined in [26]

$$
\begin{aligned}
tr_{1_{ij}} &= \left((A^{\sigma}\cdot B^{\phi_i})^{\sigma\cdot x_c'}\cdot (g_1^{\eta_i})^{\xi}\right)^{x_c''}\cdot g_1^{\zeta\cdot x_c\cdot x_c''}\\
&= g_1^{(\sigma'+\phi_i')\frac{\sigma'}{\mathfrak{a}}\cdot x_c'\cdot x_c''+\xi\cdot\eta_i\cdot x_c''+\zeta\cdot x_c\cdot x_c''}\\
tr_{2_{ij}} &= \left((D^{\phi_i})^{\frac{1}{\sigma\cdot x_c'}}\right)^{\frac{1}{x_c''}} = g_2^{\frac{\phi_i'\cdot\mathfrak{a}}{\sigma'\cdot x_c'\cdot x_c''}}\\
tr_{3_j} &= Z^{\frac{(h_{id_j}+u)h_{w_b}\cdot x_c''}{\sigma}} = Z^{\frac{\mathfrak{a}(h_{id_j}+u)(h_{w_b}\cdot x_c'')}{\sigma'}}\\
tr_{4_j} &= Z^{\psi(h_{id_j}+u)x_c''}\\
tr_{5_j} &= Z^{(h_{id_j}+u)x_c''/x_c}\\
tr_{6_j} &= C^{\prod_{\forall i\in(\mathbb{U}_\mathbb{A}\setminus\mathbb{S}_j)}(\eta_i+\kappa\cdot h_{id_j})/x_c''}\\
&= g_2^{\mathfrak{a}\prod_{\forall i\in(\mathbb{U}_\mathbb{A}\setminus\mathbb{S}_j)}(\eta_i+\kappa\cdot h_{id_j})/x_c''}
\end{aligned}
$$

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

– PHASE 2 Same as **PHASE 1**

– GUESS The adversary $\mathcal{A}_2$ guesses a bit b' and sends to the simulator $\mathcal{B}$. If $b'=b$ then the adversary $\mathcal{A}_2$ wins IND-O-KGA game; otherwise it fails. If $b'=b$, simulator $\mathcal{B}$ answers "MXDH" in the game (i.e. outputs $l=0$); otherwise $\mathcal{B}$ answers "random" (i.e. outputs $l=1$).

If $Z=g_1^{\mathfrak{r}}$; then $\mathtt{Trap}_{w_b}^{c}$ is completely random from the view of the adversary $\mathcal{A}_2$. Therefore, the adversary $\mathcal{A}_2$ chooses b' randomly. Hence, probability of the adversary $\mathcal{A}_2$ for outputting $b'=b$ is $\frac{1}{2}$.

If $Z=g_1^{\mathfrak{ab}}$, then adversary $\mathcal{A}_2$ receives a valid encrypted index. The adversary $\mathcal{A}_2$ knowing more than expected in the IND-sCP-CKA game (random parameters being set by the challenger to some predefined values), wins the IND-sCP-CKA game with non-negligible advantage ϵ (according to the Theorem 2). So, the probability of outputting $b'=b$ for the adversary $\mathcal{A}_2$ is $\frac{1}{2}+\epsilon$, where probability ϵ is for guessing that the received trapdoor is valid and probability $\frac{1}{2}$ is for guessing whether the valid trapdoor $\mathtt{Trap}_{w_b}$ is related to w_0 or w_1.

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

7 PERFORMANCE ANALYSIS

This section presents a comprehensive performance analysis of the proposed scheme, based on a detailed comparison with most closely-related works [24], [25], [26] in terms of functionality, storage and computation overhead. For this purpose, we consider the same security level for the computation of cryptographic algorithms of all studied schemes. The proposed scheme as well as two closely-related constructions Sun *et al.*'s scheme [24] and Hu *et al.*'s scheme [25] are implemented using PBC library [58]. The elementary cryptographic operations that are performed by the data owners and users are implemented using a commodity Laptop Computer having 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 VA and CSP are implemented using a workstation having 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. MNT curve with embedding degree 6 of 160-bit group order is used to implement the proposed scheme which provides an equivalent 1024-bit discrete log security. Table 2 shows computation time of the elementary mathematical functions like exponentiation operations on $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, pairing operation, group element multiplication operations on $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, and hash operation in the chosen MNT curve. From the implementation results of elementary mathematical functions, it has been observed that the computation time of cryptographic operations like group element multiplications and hash operation are negligible compared with the other cryptographic operations.

The notations used in the following sections are introduced in Table 3.

7.1 Functionality Comparison

Table 4 shows the functionality comparison of the proposed scheme with most closely-related works [24], [26], and [25]. From Table 4, it can be observed that the proposed scheme, [24] and [25] are based on CP-ABE, while [26] is based on KP-ABE. It is to be noted that CP-ABE encryption mechanism is considered as more suitable than KP-ABE [10], [24] for data sharing in a large distributed environment like cloud environment, as it enables the Owners to choose the access policies of their choice. In [26], an online trusted *authority* issues trapdoors for the users. This is not a desirable property as the user may have to contact the authority for each search request. In addition, the authority chooses the

TABLE 4: Functionality Comparison

	ABE type	Expressiveness (gates)	SCF	OKGA resistant	Replay attack resistant	CKS	Revocation	Pairing type
[24]	CP-ABE	AND	No	No	No	Yes	Yes	Symmetric
[26]	KP-ABE	AND, OR	Yes	Yes	Yes	Yes	No	Symmetric
[25]	CP-ABE	AND, OR	No	No	No	No	No	Symmetric
Proposed scheme	CP-ABE	AND, OR	Yes	Yes	Yes	Yes	Yes	Asymmetric

SCF: Secure-Channel Free; OKGA: Outsider's Keyword Guessing Attacks; CKS: Conjunctive Keyword Search.

TABLE 5: Storage and Communication Overhead Comparison

	$\mathbb{CT}$ size	Trapdoor size	Master secret size
[24]	$(1+\lvert\mathbb{U}_\mathbb{A}\rvert+j\mathcal{K}_c)^{\#}\lvert\mathbb{G}_1\rvert+\lvert\mathbb{G}_T\rvert$	$\lvert\mathbb{Z}_q^*\rvert+(1+2\lvert\mathbb{U}_\mathbb{A}\rvert)\lvert\mathbb{G}_1\rvert$	$(3\lvert\mathbb{U}_\mathbb{A}\rvert+1)\lvert\mathbb{Z}_q^*\rvert$
[26]	$(1+5\mathcal{K}_c)\lvert\mathbb{G}_1\rvert+\lvert\mathbb{G}_T\rvert$	$(2+6\mathcal{K}_a)\lvert\mathbb{G}_1\rvert$	$5\lvert\mathbb{Z}_q^*\rvert$
[25]	$(2n_{ei}+\mathcal{K}_c+2)\lvert\mathbb{G}_1\rvert$	$(2n_t+3)\lvert\mathbb{G}_1\rvert$	$3\lvert\mathbb{Z}_q^*\rvert$
Proposed scheme	$\mathcal{K}_c\lvert\mathbb{G}_1\rvert+2\lvert\mathbb{G}_2\rvert+(n_{or}+1)\lvert\mathbb{Z}_q^*\rvert$	$(n_t+2)(\lvert\mathbb{G}_1\rvert+\lvert\mathbb{G}_2\rvert)$	$(2\lvert\mathbb{U}_\mathbb{A}\rvert+6)\lvert\mathbb{Z}_q^*\rvert+\lvert\mathbb{G}_2\rvert$

$^{\#}1\leq j\leq\lvert\mathbb{U}_\mathbb{A}\rvert$;

TABLE 6: Computation Complexity Comparison

	Encryption	Trapdoor Generation	Search	Revocation
[24]	$(\mathcal{K}_c+\lvert\mathbb{U}_\mathbb{A}\rvert)T_{exp_{\mathbb{G}_1}}+T_{exp_{\mathbb{G}_T}}$	$(1+2\lvert\mathbb{U}_\mathbb{A}\rvert)T_{exp_{\mathbb{G}_1}}$	$T_{exp_{\mathbb{G}_T}}+(\lvert\mathbb{U}_\mathbb{A}\rvert+1)T_p$	$n_r(n_u+n_a)T_{exp_{\mathbb{G}_1}}$
[26]	$(7\mathcal{K}_c+1)T_{exp_{\mathbb{G}_1}}+T_{exp_{\mathbb{G}_T}}$	$(2+10\mathcal{K}_a)T_{exp_{\mathbb{G}_1}}+T_{exp_{\mathbb{G}_T}}+T_p$	$\leq(n_1+1)T_{exp_{\mathbb{G}_1}}+(6n_1+1)T_p$	Not applicable
[25]	$(2n_{ei}+\mathcal{K}_c+3)T_{exp_{\mathbb{G}_1}}$	$(2n_t+4)T_{exp_{\mathbb{G}_1}}$	$(2n_t+3)T_p+n_tT_{exp_{\mathbb{G}_1}}$	Not applicable
Proposed scheme	$(n_{or}+2\mathcal{K}_c+1)T_{exp_{\mathbb{G}_1}}+2T_{exp_{\mathbb{G}_2}}+(n_{or}+1)T_p$	User: $(n_t+3)T_{exp_{\mathbb{G}_1}}+(n_t+2)T_{exp_{\mathbb{G}_2}}+2T_p$; CSP: $T_{exp_{\mathbb{G}_1}}+T_{exp_{\mathbb{G}_2}}$	VA: $[(n_t+2)T_{exp_{\mathbb{G}_1}}+2T_p]^{\#}$; CSP: $3T_{exp_{\mathbb{G}_1}}+4T_p$	$n_r\cdot n_u\cdot T_{exp_{\mathbb{G}_2}}$

Computed only once per user request.

access policy that has to be integrated into the trapdoor. Thus, only the authority has full control over the access policies.

Access policy expressiveness is an important requirement. In the proposed scheme, Owner can choose a propositional formula which contains both boolean "AND" and "OR" gates. Similarly, [26] and [25] also provide expressiveness in the access policies. However, access policies, in [24], contain only "AND" gates.

Unlike [24] and [25], the proposed scheme does not need any secure-channels between a user and CSP to transmit trapdoors, which reduces overhead in the system. The proposed scheme also prevents replay attacks and keyword guessing attacks if an outsider adversary captures a trapdoor; while [24] and [25] are susceptible to both replay attacks and keyword guessing attacks if an adversary captures a trapdoor.

The proposed scheme, like [24], [26], provides conjunctive keyword search without incurring additional overhead in the system, which is desirable; while [25] does not provide conjunctive keyword search.

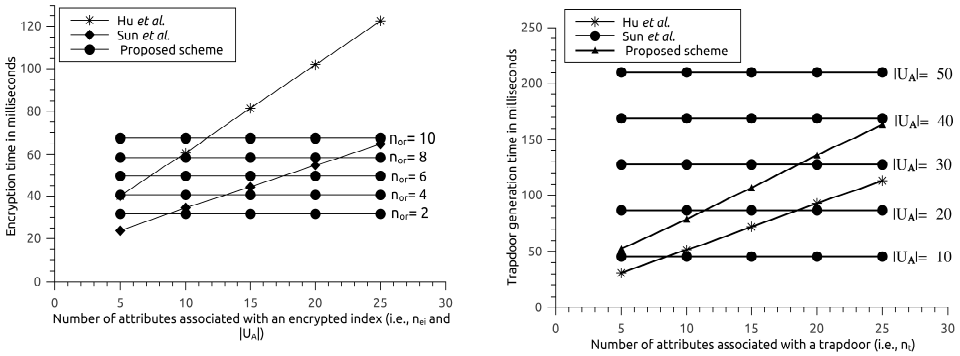
In the proposed scheme users are revoked in two ways: by revoking a user completely from the system and by revoking some attributes of the user. The former approach is performed by removing the public key of the user from the public bulletin board and the latter approach is performed by updating witness of the users who possess the revoked attributes. While, in [24], users are revoked per file by removing his/her identity from a *user list* associated with the file and also, it revokes attributes of a user by performing re-encryption of the encrypted indexes associated with the revoked attributes and updating secret keys of the non-revoked users who possess the revoked attributes. It can be observed that the re-encryption is a costly operation in [24], as the revoked attributes can be associated with a large number of encrypted indexes.

Finally, it is worth noting that the proposed scheme is constructed using an asymmetric pairing function, while most closely-related schemes rely on symmetric pairings. It is believed that asymmetric pairing is a better choice for designing a cryptographic scheme due to its better security than symmetric pairings [59], [60]. Thus, it can be concluded that the proposed scheme provides better security than [24], [25] and [26].

7.2 Storage and Communication Overhead Comparison

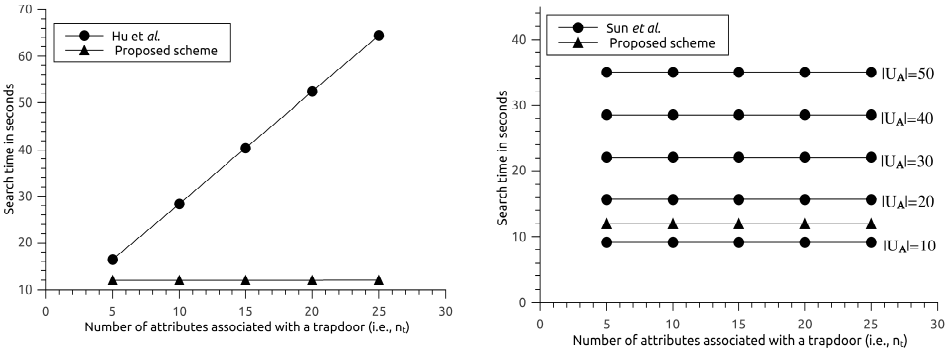
In this section, we compare the proposed scheme with [24], [25] and [26] schemes. The storage and communication overhead are measured in terms of group element size, i.e. $\lvert\mathbb{G}_1\rvert, \lvert\mathbb{G}_2\rvert, \lvert\mathbb{G}_T\rvert$ and $\lvert\mathbb{Z}_q^*\rvert$.

Table 5 presents storage and communication overhead incur due to an encrypted index (i.e. $\mathbb{CT}$), a trapdoor, and master secret key. From Table 5, it can be observed that the size of the encrypted index of the proposed scheme depends on *OR* gates in the access policy and the set of keywords associated with the encrypted index. It can be concluded that the encrypted index size in the proposed scheme is less than [24] and [25], as the number of *OR* gates associated with an encrypted index is smaller than the total number of attributes in the system as in [24] and the number of attributes associated with the encrypted index as in [25]. On the other hand, the encrypted index size in [26] depends on the number of keywords associated with the encrypted index.



(a) Encryption ($\mathcal{K}_c = 5$) (b) Trapdoor Generation

Fig. 2: Comparison of computation time of the proposed scheme with Sun *et al.*'s scheme [24] and Hu *et al.*'s scheme [25]. (a) Encryption and (b) Trapdoor Generation



(a) Comparison between the proposed scheme and Hu *et al.*'s scheme [25] (b) Comparison between the proposed scheme and Sun *et al.*'s scheme [24]

Fig. 3: Comparison of computation time for the Search phase considering 1000 encrypted indexes

The trapdoor size of the proposed scheme depends on the attributes associated with the trapdoor, like [25]; while in [24], the trapdoor size depends on the total number of attributes in the system which can be a large number. Thus, we can conclude that the trapdoor generation mechanism of the proposed scheme is better than [24]. On the other hand, trapdoor size in [26] depends on the keywords associated with the trapdoor, thus meaning that separate trapdoor per keyword search has to be issued by AA, and AA has to always remain online.

From Table 5, it can be seen that the size of the master secret key of the proposed scheme is greater than [25] and [26], while it is less than [24].

7.3 Computation Cost Comparison

In this section, the computation overhead of the proposed scheme is compared with the closely related works [24], [25] and [26]. First, we provide a comparison of the proposed scheme with the related works [24], [25] and [26] in terms of computation complexities. It is followed by the implementation results comparison of the proposed scheme with [24] and [25][10].

[10]For the implementation, Cui *et al.*'s scheme [26] is not considered, as it is designed using KP-ABE technique; while the proposed scheme and [24], [25] are designed using CP-ABE technique.

7.3.1 Comparison of Computation Complexities

Table 6 presents a comparison of the computation complexities of the proposed scheme with [24], [25] and [26] in terms of exponentiation operations (i.e., $T_{exp_{\mathbb{G}_1}}$, $T_{exp_{\mathbb{G}_2}}$, $T_{exp_{\mathbb{G}_T}}$) and pairing operation (i.e., T_p). It is to be noted that the computation cost of group element multiplications and hash operations are negligible compared with the pairing and exponentiation operations according to the Table 2. Therefore, we ignore them in the rest of the comparisons. The computation cost, as depicted in Table 6, represents the upper bound in the worst cases and incurs in *Encryption*, *Trapdoor Generation*, *Search* and *Revocation* phases as detailed hereafter.

The computational overhead during the *Encryption* phase is incurred by the Owner who has to perform cryptographic operations to generate an encrypted index. In the *Encryption* phase of the proposed scheme, Owner encrypts a set of keywords by computing $(n_{or} + 2\mathcal{K}_c + 3)$ exponentiation operations and $(n_{or} + 1)$ pairing operations. Thus, it shows that the encryption cost of the proposed scheme mainly depends on the number of both "OR" gates in the access policy and keywords associated with an encrypted index. The encryption cost of [24], [25] and [26] depends on the number of keywords associated with an encrypted index, and also [24], [25] depend on the cardinal of attributes' universe in the system as well as the number of attributes

associated with an encrypted index respectively. It can be observed that the number of "OR" gates in an encrypted index (or access policy) is smaller than the number of attributes associated with an access policy and the total number of attributes in the system. Therefore, the *Encryption* phase of the proposed scheme is less expensive than [24] and [25].

The computational overhead during the *Trapdoor Generation* phase is mainly incurred by the User who has to perform cryptographic operations to generate a trapdoor using his/her secrets and the desired keywords. In the *Trapdoor Generation* phase, the proposed scheme uses an interactive protocol between a user and the CSP to generate a trapdoor. Due to the interactive protocol, in the proposed scheme, the CSP computes two exponentiation operations, while the user computes one exponentiation and two pairing operations in addition to the $2n_t + 4$ exponentiation operations for computing the actual trapdoor. From the table, it can be observed that the trapdoor generation cost of the proposed scheme is lower than [24] as $n_t < |\mathbb{U}_{\mathbb{A}}|$. On the other hand, the trapdoor generation cost in [26] depends on the number of keywords associated with the access structure. However, in [26], AA chooses the access structure based on the keywords to be searched, which is not desirable as mentioned earlier.

The computational overhead during the *Search* phase is mainly incurred by the CSP and VA who has to perform cryptographic operations during keyword search operation over the encrypted indexes. In the *Search* phase of the proposed scheme, VA first performs attribute witness verification procedure which takes $(n_t + 2)$ exponentiation and 2 pairing operations. Note that these operations are performed by the VA only once per keyword search request from a user. If the user successfully passes the attribute verification phase, CSP performs keyword search by executing 3 exponentiation and 4 pairing operations per encrypted index, which substantially reduces search time compared with [24], [25] and [26].

For the attributes revocation of a user, the proposed scheme requires to compute one exponentiation operation per revoked attribute. This exponentiation operation is required to update the attribute witness associated with each user who possesses the revoked attribute. While revoking attributes of a user, in [24], one exponentiation operation per revoked attribute is required to update secret keys of each non-revoked user who possesses the revoked attribute and another one exponentiation operation per revoked attribute is required to re-encrypt the encrypted indexes associated with the revoked attribute. Thus, the attribute revocation operation in [24] requires more exponentiation operations compared with the proposed scheme, as the number of re-encrypted indexes can be large.

7.3.2 Comparison of the Implementation Results

Figure 2a shows the encryption time comparison of the proposed scheme with [24], and [25], where we consider fixed 5-keywords associated with an encrypted index (i.e. $\mathcal{K}_c = 5$). The computation cost of the encryption process in the proposed scheme mainly depends on the number of "OR" gates associated with access policy; while [24] and [25] depend on the total number of attributes in the system and the number of attributes associated with the access policy respectively. As the number of "OR" gates (i.e., n_{or}) can be a smaller number than the total number of attributes in the system (i.e., $|\mathbb{U}_{\mathbb{A}}|$) and the number of attributes associated with the access policy (i.e., n_{ei}), the proposed scheme takes less computation time compared with [24] and [25] which can be observed from the Figure 2a. In [24], cost of the trapdoor generation process depends on the total number of attributes in the system; while the cost of the trapdoor generation process in the proposed scheme depends on the number of attributes associated with the trapdoor, i.e. n_t. As the number of attributes associated with the trapdoor can be smaller than the total number of attributes in the system, the proposed scheme takes less computation time during trapdoor generation process which can be observed from Figure 2b. However, the trapdoor generation with [25] is less computation demanding than the proposed scheme due to less secure but more computationally efficient symmetric pairing.

Figure 3a and Figure 3b show comparisons of the processing cost required to perform a search operation over 1000 encrypted indexes between the [25], [24], and the proposed scheme respectively. From the Figure 3a and Figure 3b, it can be seen that the search time of the proposed scheme is substantially less than in [25] and [24]. This due to the fact that in the proposed scheme, for each search request the VA computes $(n_t+2)T_{exp_{G_1}} + 2T_p$ operations only once per search request and other $3T_{exp_{G_1}} + 4T_p$ operations are performed per encrypted index; while [24] and [25] need to perform $T_{exp_{\mathbb{G}_T}} + (|\mathbb{U}_{\mathbb{A}}| + 1)T_p$ and $(2n_{ei} + 3)T_p + n_{ei}T_{exp_{\mathbb{G}_1}}$ number of operations per encrypted index respectively which are computing more exponentiation and pairing operations than the proposed scheme.

8 Conclusion

In this paper, a fine-grained authorized expressive keyword search scheme has been proposed. The proposed scheme uses the benefits of CP-ABE to perform efficient keyword search over encrypted data and for authorizing users. In addition, it uses an interactive protocol for users to send trapdoors to the CSP, thus avoiding construction of secure-channels. Further, it uses a bilinear-map accumulator for providing efficient user revocation, thereby avoiding the cumbersome re-encryption operations over the encrypted data. Conjunctive keyword search functionality is added by the proposed scheme at no extra overhead. The security analysis is performed under standard complexity assumptions and it shows that the proposed scheme is semantically secure against the chosen keyword attack. The scheme is also resistant against keyword guessing attacks. Moreover, the performance analysis of the proposed scheme shows that it outperforms the closely-related works in terms of storage, communication overhead, and computational overhead.

Security, efficiency, and query expressiveness are the three main factors that define the practical usability of a keyword search scheme [4]. A user expects that the keyword search scheme provides strong security and high efficiency (in terms of computation and communication costs), as high

as for normal plaintext search. Further, the query expressiveness will provide the user to make different types of queries. In this paper, we have made an effort to maintain a balance among these three factors. We believe that there is still much work to do for improving the balance among these three factors, i.e., security, efficiency, and query expressiveness. This will lead to the use of the keyword search schemes in privacy-preserving personalized services over the Internet like online advertisements or news.

ACKNOWLEDGMENTS

This paper is an outcome of the project "A Secure and Scalable Inter-cloud Authorization Framework for Resource Sharing in Cloud Environment", funded by Science and Engineering Research Board, Govt. of India (File No: ECR/2016/001822).

REFERENCES

[1] N. Kaaniche and M. Laurent. Data security and privacy preservation in cloud storage environments based on cryptographic mechanisms. *Computer Communications*, 111:120–141, 2017.

[2] J. Tang, Y. Cui, Q. Li, K. Ren, J. Liu, and R. Buyya. Ensuring security and privacy preservation for cloud data services. *ACM Computing Survey*, 49(1):13:1–13:39, Jun 2016.

[3] F. Han, J. Qin, and J. Hu. Secure searches in the cloud. *Future Generation Computer Systems.*, 62(C):66–75, Sep. 2016.

[4] C. Bösch, P. Hartel, W. Jonker, and A. Peter. A survey of provably secure searchable encryption. *ACM Computing Survey*, 47(2):18:1–18:51, Aug 2014.

[5] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, SP '00, pages 44–, 2000.

[6] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 79–88, 2006.

[7] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li. Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, ASIA CCS '13, pages 71–82, 2013.

[8] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou. Privacy-preserving multi-keyword ranked search over encrypted cloud data. *IEEE Transactions on Parallel and Distributed Systems*, 25(1):222–233, Jan 2014.

[9] C. Guo, X. Chen, Y. Jie, F. Zhangjie, M. Li, and B. Feng. Dynamic multi-phrase ranked search over encrypted data with symmetric searchable encryption. *IEEE Transactions on Services Computing*, pages 1–1, 2017.

[10] N. H. Sultan and F. A. Barbhuiya. A secure re-encryption scheme for data sharing in unreliable cloud environment. In *Proceedings of the 2016 IEEE World Congress on Services (SERVICES)*, pages 75–80, June 2016.

[11] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li. Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud. In *Proceedings of the 2014 IEEE Conference on Computer Communications*, IEEE INFOCOM, pages 226–234, April 2014.

[12] 4shared. https://www.4shared.com/ [Online accessed: 16-Dec.-2018].

[13] MediaFire. https://www.mediafire.com/ [Online accessed: 16-Dec.-2018].

[14] MEGA. https://mega.nz/[Online accessed: 16-Dec.-2018].

[15] K. Yang and X. Jia. Expressive, efficient, and revocable data access control for multi-authority cloud storage. *IEEE Transactions on Parallel and Distributed Systems*, 25(7):1735–1744, July 2014.

[16] S. Ruj, M. Stojmenovic, and A. Nayak. Decentralized access control with anonymous authentication of data stored in clouds. *IEEE Transactions on Parallel and Distributed Systems*, 25(2):384–394, Feb 2014.

[17] S. Belguith, N. Kaaniche, A. Jemai, M. Laurent, and R. Attia. PAbAC: a Privacy preserving Attribute based framework for fine grained Access Control in clouds. In *13th International Conference on Security and Cryptography*, SECRYPT'16, pages 133 – 146, July 2016.

[18] N. H. Sultan, F. A. Barbhuiya, and N. Sarma. A universal cloud user revocation scheme with key-escrow resistance for ciphertext-policy attribute-based access control. In *Proceedings of the 10th International Conference on Security of Information and Networks*, SIN '17, pages 11–18, 2017.

[19] N. Kaaniche and M. Laurent. Attribute based encryption for multi-level access control policies. In *14th International Conference on Security and Cryptography*, volume 6 of *SECRYPT'17*, pages 67 – 78, July 2017.

[20] M. Sookhak, F. R. Yu, M. K. Khan, Y. Xiang, and R. Buyya. Attribute-based data access control in mobile cloud computing. *Future Generale Computer System*, 72(C):273–287, July 2017.

[21] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Proceedings of the 24th Annual International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'05, pages 457–473, 2005.

[22] V. Goyal, O. Pandey, A. Sahai, and B. Waters. 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, pages 89–98, 2006.

[23] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy (SP '07)*, pages 321–334, May 2007.

[24] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li. 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):1187–1198, April 2016.

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

[26] H. Cui, Z. Wan, R. Deng, G. Wang, and Y. Li. Efficient and expressive keyword search over encrypted data in the cloud. *IEEE Transactions on Dependable and Secure Computing*, PP(99):1–1, 2017.

[27] P. Jiang, Y. Mu, F. Guo, and Q. Wen. Secure-channel free keyword search with authorization in manager-centric databases. *Computers & Security*, 69:50 – 64, 2017.

[28] J. Baek, R. Safavi-Naini, and W. Susilo. Public key encryption with keyword search revisited. In *Proceeding of the International Conference on Computational Science and Its Applications, Part I*, ICCSA '08, pages 1249–1259, 2008.

[29] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Advances in Cryptology - EUROCRYPT 2004*, pages 506–522, 2004.

[30] L. Fang, W. Susilo, C. Ge, and J. Wang. A secure channel free public key encryption with keyword search scheme without random oracle. In *Cryptology and Network Security*, pages 248–258, 2009.

[31] W. Yau, R. C.-W. Phan, S. Heng, and B. Goi. Keyword guessing attacks on secure searchable public key encryption schemes with a designated tester. *International Journal of Computer Mathematics*, 90(12):2581–2587, 2013.

[32] Q. Huang and H. Li. An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. *Information Sciences*, 403-404:1 – 14, 2017.

[33] R.A. Popa and N. Zeldovich. Multi key searchable encryption. 2013. https://people.csail.mit.edu/nickolai/papers/popa-multikey-eprint.pdf.

[34] A. Kiayias, O. Oksuz, A. Russell, Q. Tang, and B. Wang. Efficient encrypted keyword search for multi-user data sharing. In *Proceedings of the 2016 European Symposium on Research in Computer Security*, ESORICS, pages 173–195, 2016.

[35] P. Xu, S. Tang, P. Xu, Q. Wu, H. Hu, and W. Susilo. Practical multi-keyword and boolean search over encrypted e-mail in cloud server. *IEEE Transactions on Services Computing*, pages 1–1, 2019.

[36] P. Golle, J. Staddon, and B. Waters. Secure conjunctive keyword search over encrypted data. In *Proceedings of the Applied Cryptography and Network Security*, pages 31–45, 2004.

[37] Y. Yang and M. Ma. Conjunctive keyword search with designated tester and timing enabled proxy re-encryption function for e-health clouds. *IEEE Transactions on Information Forensics and Security*, 11(4):746–759, April 2016.

[38] Y. Miao, J. Ma, X. Liu, Q. Jiang, J. Zhang, L. Shen, and Z. Liu. Vcksm: Verifiable conjunctive keyword search over mobile e-health cloud in shared multi-owner settings. *Pervasive and Mobile Computing*, 40:205 – 219, 2017.

[39] W. Zhang, S. Xiao, Y. Lin, T. Zhou, and S. Zhou. Secure ranked multi-keyword search for multiple data owners in cloud computing. In *Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, DSN '14, pages 276–286, 2014.

[40] W. Zhang, Y. Lin, and Q. Gu. Catch you if you misbehave: Ranked keyword search results verification in cloud computing. *IEEE Transactions on Cloud Computing*, PP(99):1–1, 2017.

[41] W. Zhang, Y. Lin, and G. Qi. Catch you if you misbehave: Ranked keyword search results verification in cloud computing. *IEEE Transactions on Cloud Computing*, 6(1):74–86, Jan 2018.

[42] J. Li, D. Lin, A. C. Squicciarini, J. Li, and C. Jia. Towards privacy-preserving storage and retrieval in multiple clouds. *IEEE Transactions on Cloud Computing*, 5(3):499–509, July 2017.

[43] L. Xu, S. Sun, X. Yuan, J. K. Liu, C. Zuo, and X. Chungen. Enabling authorized encrypted search for multi-authority medical databases. *IEEE Transactions on Emerging Topics in Computing*, pages 1–1, 2019.

[44] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee. Improved searchable public key encryption with designated tester. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, ASIACCS '09, pages 376–379, 2009.

[45] L. Fang, W. Susilo, C. Ge, and J. Wang. Public key encryption with keyword search secure against keyword guessing attacks without random oracle. *Information Sciences*, 238(Supplement C):221 – 241, 2013.

[46] M. Li, S. Yu, N. Cao, and W. Lou. Authorized private keyword search over encrypted data in cloud computing. In *proceedings of the 31st International Conference on Distributed Computing Systems*, pages 383–392, June 2011.

[47] T. Okamoto and K. Takashima. Hierarchical predicate encryption for inner-products. In *Advances in Cryptology – ASIACRYPT 2009*, pages 214–231, 2009.

[48] I. R. Jeong, J. O. Kwon, D. Hong, and D. H. Lee. Constructing peks schemes secure against keyword guessing attacks is possible? *Computer Communications*, 32(2):394 – 396, 2009.

[49] Y. Zheng, T. Hardjono, and J. Pieprzyk. The sibling intractable function family (SIFF): notion, construction and applications. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, 76(1):4–13, 1993.

[50] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Proceedings of the Advances in Cryptology–CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229, 2001.

[51] M. H. Au, P. P. Tsang, W. Susilo, and Y. Mu. Dynamic universal accumulators for ddh groups and their application to attribute-based anonymous credential systems. In *Proceedings of the Cryptographers' Track at the RSA Conference-Topics in Cryptology*, CT-RSA, pages 295–308, April 2009.

[52] I. Damgard and N. Triandopoulos. Supporting non-membership proofs with bilinear-map accumulators. In *Cryptology ePrint Archive, Report 2008/538*, 2008. [online]. Available: http://eprint.iacr.org/2008/538.

[53] C. Papamanthou, R. Tamassia, and N. Triandopoulos. Optimal verification of operations on dynamic sets. In *Proccedings of the 31st Annual Cryptology Conference- Advances in Cryptology*, CRYPTO 2011, pages 91–110, August 2011.

[54] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '01, pages 514–532, Dec. 2001.

[55] L. Ballard, M. Green, B. D. Medeiros, and F. Monrose. Correlation-resistant storage via keyword-searchable encryption, 2005. Technical Report TR-SP-BGMM-050507, Johns Hopkins University, Department of Computer Science.

[56] L. Ballard, S. Kamara, and F. Monrose. Achieving efficient conjunctive keyword searches over encrypted data. In *Proceedings of the 7th International Conference on Information and Communications Security*, ICICS'05, pages 414–426, 2005.

[57] T. Zhao, L. Wei, and C. Zhang. Attribute-based encryption scheme based on siff. In *Proceedings of the 2016 IEEE International Conference on Communications (ICC)*, pages 1–6, May 2016.

[58] PBC (Pairing-Based Cryptography) library. http://crypto.stanford.edu/pbc/ [Online accessed: 12-Dec.-2018].

[59] D. Freeman, M. Scott, and E. Teske. A taxonomy of pairing-friendly elliptic curves. *Journal of Cryptology*, 23(2):224–280, April 2010.

[60] P. S. Barreto, C. Costello, R. Misoczki, M. Naehrig, G. C. Pereira, and G. Zanon. Subgroup security in pairing-based cryptography. In *Proceedings of the 4th International Conference on Progress in Cryptology*, LATINCRYPT, pages 245–265, 2015.

Nazatul Haque Sultan is a doctoral candidate at Indian Institute of Information Technology Guwahati (IIITG). Mr. Sultan has done his MTech in Information Technology from Tezpur University in 2014 and BE in Computer Science and Engineering from Dibrugarh University in 2012. Mr. Sultan was a visiting researcher at Télécom SudParis, France from Nov. 2017 to Jan. 2018 and Advanced Cyber Security Engineering Research Centre at The University of Newcastle, Australia from June 2018 to Sept. 2018. He is a student member of IEEE. His research interests include Security and Privacy related to Cloud and IoT, Access Control, Searchable Encryption, Applied Cryptography and Blockchain.

Nesrine Kaaniche is a Lecturer in Cybersecurity at the Department of Computer Science, University of Sheffield, co-affiliated with the Security of Advanced Systems Research Group. Previously, she was a research member of the chair Values and Policies of Personal Information, at Télécom SudParis, Institut Polytechnique de Paris, France. She received a PhD degree on cloud storage security jointly from Sorbonne University and Telecom SudParis, France, in 2014. Her major research interests include privacy enhancing technologies and applied cryptography for distributed systems and decentralised architectures, i.e., IoT, fog, cloud, and blockchains. She served as Technical Program Committee member for several conferences, and as referee for several outstanding international journals.

Maryline Laurent works as a Professor at Télécom SudParis, Institut Mines-Télécom, France. She leads the research team R3S (Network, Systems, Services, Security) of the CNRS SAMOVAR lab of Télécom SudParis, and she is co-founder of the chair of Institut Mines-Télécom on Values and Policies of Personal Information. She is associate editor of Annals of Telecommunication journal, and editor of several books including Digital Identity Management (2015). Her research topics are related to network security and privacy applied to clouds, Internet of Things and identity management.

Ferdous Ahmed Barbhuiya is an Associate Professor at the Indian Institute of Information Technology Guwahati (IIITG). Dr. Barbhuiya has obtained his PhD as well as MTech degree from Indian Institute of Technology (IIT) Guwahati and BE degree from Dibrugarh University, all in Computer Science and Engineering. He is a member of IEEE and ACM. His research interests include Cloud Computing, Software Defined Network, Network Function Virtualization, Computer and Network Security etc.