BTG-RKASE: Privacy Preserving Revocable Key Aggregate Searchable Encryption with Fine-grained Multi-delegation & Break-The-Glass Access Control

Mukti Padhya[1][a] and Devesh C. Jinwala[2]

[1]*Department of Computer Engineering, Sardar Vallabhbhai National Institute of Technology (SVNIT), Surat, India*

[2]*Department of Computer Science and Engineering, Indian Institute of Technology, Jammu, India*

Keywords: Searchable Encryption, Data Sharing, Data Retrieval, Cloud Server, Multi-keyword Search, Multi-delegation, Revocation, Break-The-Glass Access.

Abstract: Delegation is the technique of sharing the available rights from the delegator to the delegatee for the purpose data sharing. The Key Aggregate Searchable Encryption (KASE) scheme supports delegation of search rights for any set of ciphertexts using a key of constant-size. However, three critical issues still need to be considered. Firstly, the existing KASE schemes only discuss delegation of rights from the data owner to other user. However, if a subject receiving a delegation cannot perform time-critical task on the shared data, it becomes necessary for the delegatee to further delegate their received rights to another user. Secondly, the existing delegation mechanisms tend to rely on manual processes initiated by end-users. If no authorized user exists to perform (or to delegate) a time-critical task, in such exceptional case, we require mechanism that flexibly handles emergency situations by breaking or by controlled overriding of the standard access permissions. Thirdly, the access of user in the system changes dynamically and it requires KASE to support user revocation securely while not affecting the legitimate users' access to the shared files. To address all of the above issues, we propose Revocable KASE with Break-The-Glass access control (BTG-RKASE) to provide (i)fine-grained multi-delegation of available rights from the delegatee to another user,(ii)break-the-glass access mechanism when no authorized user exists to perform (or to delegate) a time-critical task,(iii)revocation of delegated rights (even in case of multi-delegation). The security and empirical analysis shows that BTG-RKASE performs better than the existing KASE schemes.

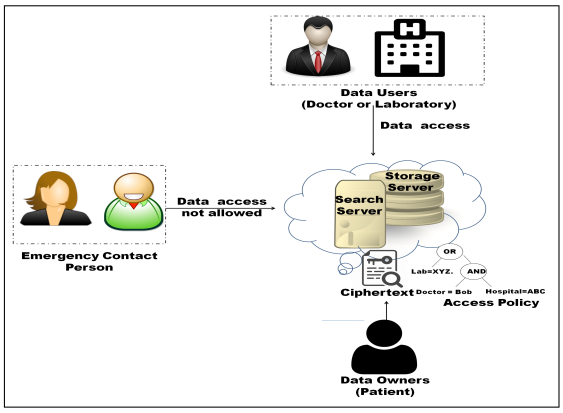
1 INTRODUCTION

The cryptography-based access control (Sandhu et al., 1996; Sahai et al., 2005) is designed to prevent unauthorized access of data stored on the untrusted cloud server. However, in some exceptional situations the tight access policies of traditional access control schemes (such as Role-Based Access Control (RBAC) (Sandhu et al., 1996), Attribute-Based Access Control (ABAC) (Sahai et al., 2005)) prevent the data accessibility. Let us consider the following example from a healthcare application to clarify the issues we aim to address in this paper.

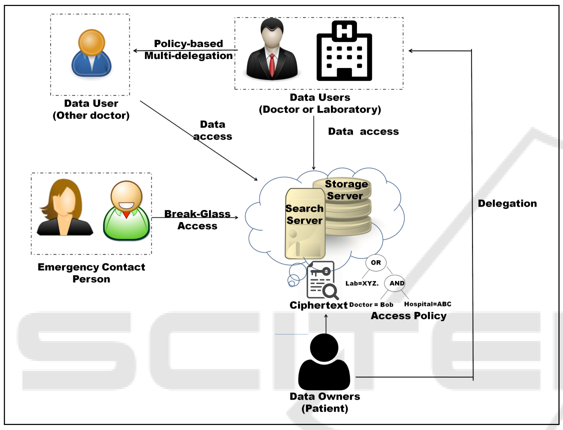
The E-Healthcare Record (EHR) storage system allows patients to store their health-related information on the cloud server. The EHR records contain the patient's personal information and his health information such as the patient's blood pressure, heartbeats etc. However, the privacy and confidentiality of that patient's medical records cannot be neglected. Therefore, to prevent unauthorized access of medical records, patient (Alice) use cryptography-based access control and define access policy for the health records (Figure 1a). For example, the access policy *(Doctor ='Bob' and Hospital='ABC') or Lab='XYZ'* allows doctor Bob from 'ABC' hospital or manager of 'XYZ' laboratory to access Alice's data. Suppose, Bob is away from work for some particular reason, such as sickness or on holiday, who has the right to access Alice's medical records in order to evaluate and treat her appropriately? As per the access policy defined by Alice, other doctors or nurses from the emergency ward can not access Alice's medical record. If we consider the sickness of the patient, data accessibility is needed to give timely treatment, at the same time we cannot ignore the privacy of patient's med-

[a] https://orcid.org/0000-0002-0498-4188

Padhya, M. and Jinwala, D.
BTG-RKASE: Privacy Preserving Revocable Key Aggregate Searchable Encryption with Fine-grained Multi-delegation Break-The-Glass Access Control.
DOI: 10.5220/0007919901090124
In *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications (ICETE 2019)*, pages 109-124
ISBN: 978-989-758-378-0



(a) Traditional approach of access control using the existing cryptography-based schemes (Sandhu et al., 1996; Sahai et al., 2005)



(b) The proposed approach using fine-grained multi-delegation and break-the-glass access control

Figure 1: Access control and data security in EHR System.

ical record and information? The question is, how can we design an access control model to provide privacy, confidentiality and availability at the same time? The possible solution to provide flexible access control and also maintaining data security is delegation of rights over Alice's medical records from Bob to other doctor.

Delegation mechanism (Crampton and Khambhammettu, 2008; Schaad and Moffett, 2002) allow subjects to transfer their available rights (data access and search), roles, tasks, or duties to another subject. Subsequently, a subject receiving a delegation (the delegatee) will act on behalf of the delegating subject (the delegator). Recent attempts at Key Aggregate Searchable Encryption (KASE) (Cui et al., 2016) allows a data owner to delegate keyword search rights on a set S of files. KASE inherits the property of Key Aggregate Encryption (KAE) (Chu et al., 2014), i.e., to delegate search rights on $|S|$ number of files, the data owner is required to share a single aggregate key with a user. In addition, a user is required to submit a single aggregate trapdoor (instead of a group of trapdoors) to the cloud to perform a keyword search across $|S|$ number of shared files. However, the limitation of the existing delegation schemes (Chu et al., 2014; Banu, 2015; Dang et al., 2016; Mahalle and Pawade, ; Patranabis et al., 2015; Cui et al., 2016; Li et al., 2016; Li et al., 2018; Zhou et al., 2018; Padhya and Jinwala, 2018) is that they only discuss delegation of rights from the data owner to other user. On the other hand, if a subject receiving a delegation cannot perform time-critical task on the shared data, it becomes necessary to further delegate the received rights from the delegatee to another user. We define such transitive delegation of rights from the delegatee to another user a *multi-delegation*. For example, in above discussed scenario of EHR system Alice can delegate search and access rights of her records to Bob using the existing delegation schemes (Chu et al., 2014; Banu, 2015; Dang et al., 2016; Mahalle and Pawade, ; Patranabis et al., 2015; Cui et al., 2016; Li et al., 2016; Li et al., 2018; Zhou et al., 2018; Padhya and Jinwala, 2018). However, multi-delegation from Bob to other doctors is not possible in the existing delegation schemes. In the existing delegation schemes (Chu et al., 2014; Banu, 2015; Dang et al., 2016; Mahalle and Pawade, ; Patranabis et al., 2015; Cui et al., 2016; Li et al., 2016; Li et al., 2018; Zhou et al., 2018; Padhya and Jinwala, 2018), the possible way to do multi-delegation is that the delegatee can share his received key having rights for dataset S with other users. However, this solution is infeasible and insecure in case of delegatee wants to share selected dataset $S' \subset S$ with other user. Further, multi-delegation can temporarily change the access control state so as to allow a subject to use another subject's privileges. It may be possible that the data owner does not want to reveal his data to any user other than the delegatee.

Furthermore, in the critical system such as healthcare industry and disaster management, even if the set of authorized entities are unavailable, the task on the shared dataset is expected to be done in reasonable time. In the healthcare industry, patients are expected to be treated in reasonable time. In case of Bob delegates his rights over Alice's medical records to John and John meets medical emergency, then how to access the patient's data?. Therefore, we solicit the answer to an interesting research question as *how to handle exceptional situation when no authorized user exists to perform (or to delegate) a time-critical task?* Motivated by use cases from the disaster management domain, the break-the-glass principle (Joint, 2004) was introduced as one approach to flexibly handle emergency situations by breaking or overriding the standard access permissions in a controlled manner. In essence, break-the-glass policies (Ferreira et al., 2006; Ardagna et al., 2008; Ardagna et al., 2010; Marinovic et al., 2011) allow a subject to perform an action under certain conditions (e.g., life-threatening situation of patient) even though he/she was not previously authorized to do so. Usually, such override accesses are monitored and documented for later reviews and audits.

However, the break-glass policies do not take into account the existence or availability of authorized users. In essence, a break-glass policy extend the set of authorized accesses for the duration of emergency and possibly allowing a unauthorized user to access the data (or resource), even though authorized subjects are available. It is difficult to address the above violations in an access control policy for the healthcare application because an overly *loose* policy might permit access to inappropriate users, but an overly *tight* policy might prevent access from the appropriate users. A potential research direction would be to find ways to the controlled overriding of access control restrictions.

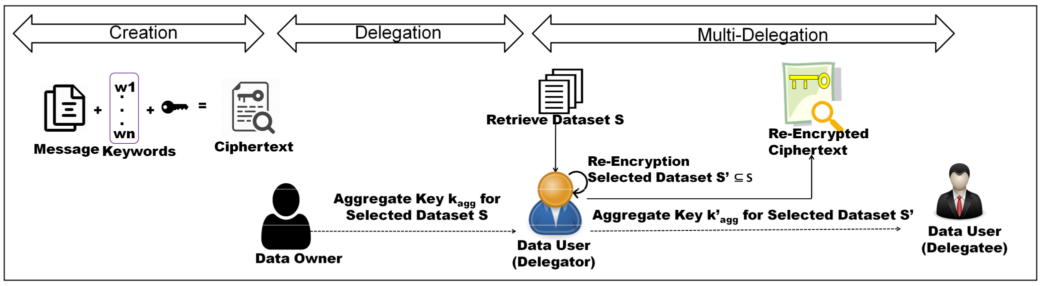
Envisioning these specific requirements in KASE, we present a KASE scheme with fine-grained multi-delegation and break-the-glass access policy. A notion of subject availability is also introduced, and the proposed scheme allows searchable group data sharing, flexible access control and data security in the following ways (Figure 1b): in normal situations, the authorized user with the delegated key can access and search over the shared dataset, provided ciphertext class is within the range of aggregate key. For the fine-grained multi-delegation, if the attributes of the delegatee satisfy the hidden access policy (defined by the data owner), the delegatee can delegate his received rights to another user. In exceptional situation (when the authorized users are not available), the break-glass access mechanism allow timely access to the data, provided the attributes of the user satisfy the break-the-glass policy. Additionally, the proposed scheme also allows revocation of delegated rights even in case of multi-delegation. The revocation refers to the process of taking away the delegated privileges. If the receiver of delegated rights leaves the system, or if the data is used differently than the data owner agreed, then data owner or delegator requires to revoke the delegated rights. Additionally, the revocation in the proposed scheme does not affects the non-revoked users, as they do not require to update their corresponding delegated keys, which greatly reduces the expensive cost of key update and the overhead of key delegate authority.

1.1 Related Work

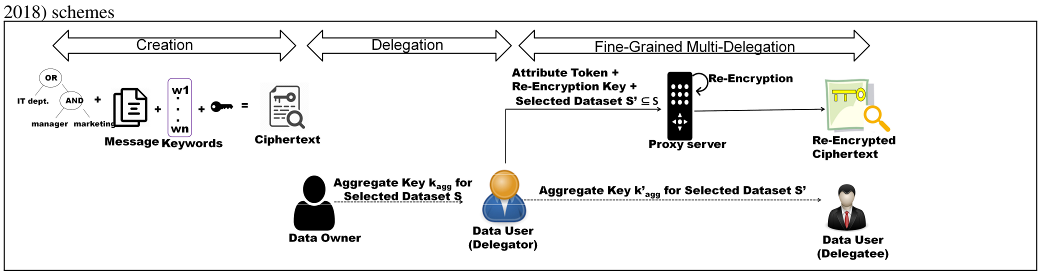
The KASE scheme (Cui et al., 2016), derives its roots from the seminal work on Key Aggregate Encryption (KAE) by Chu et.al. (Chu et al., 2014). In KASE, a data owner delegates search rights on a dataset using an aggregate key and the user can search over shared data by submitting a single aggregate trapdoor to the cloud. Despite the advantage of the secure delegation of search rights using a constant size key, KASE (Cui et al., 2016; Li et al., 2016; Li et al., 2018; Zhou et al., 2018; Padhya and Jinwala, 2018) also brings the security issues and challenges.

The KASE schemes proposed in (Cui et al., 2016; Zhou et al., 2018) do not support searching over multi-owner data using a single key of constant size. Therefore, Li *et al.* (Li et al., 2016) propose KASE to support multi-owner search using a single trapdoor and also provide verification of search results using an aggregate key. However, security proof against chosen keyword attack is not discussed for the KASE schemes (Cui et al., 2016; Li et al., 2016). Moreover, the existing KASE schemes (Cui et al., 2016; Li et al., 2016) do not provide data and trapdoor privacy, as discussed in (Kiayias et al., 2016). Zhou *et al.* show the possible insider attack on the KASE (Cui et al., 2016) scheme in (Zhou et al., 2018). Further, the KASE (Cui et al., 2016; Li et al., 2016) schemes consider the security assumption of Bilinear Diffie-Hellman Exponent (BDHE). However, SDH or its variants such as the BDHE do not guarantee the security for specific parameters, as discussed in (Cheon, 2006). In (Padhya and Jinwala, 2018), the authors proposed KASE that supports search over multi-owner data using a single aggregate key and also overcomes other security issues of the existing KASE schemes (Cui et al., 2016; Li et al., 2016; Li et al., 2018). However, the KASE scheme proposed in (Padhya and Jinwala, 2018) does not supports conjunctive keyword search and key leakage resiliency. In (Zhou et al., 2018), the authors consider Industrial IoT (IIoT) applications and propose a KASE scheme in the file-centric framework for IIoT applications. However, this approach suffers from low performance. In this scheme, each user's public key has $3n+1$ elements and secret has $n+3$ elements. Here, n is the maximum number of documents held by a data owner.

In the existing KASE (Cui et al., 2016; Li et al., 2016; Li et al., 2018; Zhou et al., 2018; Padhya and Jinwala, 2018) schemes if the delegatee needs to share his received rights on the shared dataset with another user, then he must carry out the following steps (Figure 2a): (i) first download the selected data, (ii) retrieve the plaintext through decryption, (iii) re-



(a) Traditional approach using the existing KASE (Cui et al., 2016; Li et al., 2016; Li et al., 2018; Zhou et al., 2018; Padhya and Jinwala, 2018) schemes


Creation
Delegation
Fine-Grained Multi-Delegation
OR
IT dept.
AND
manager
marketing
w1
wn
Message Keywords
Ciphertext
Attribute Token + Re-Encryption Key + Selected Dataset S' ⊆ S
Re-Encryption
Proxy server
Re-Encrypted Ciphertext
Aggregate Key k_{agg} for Selected Dataset S
Aggregate Key k'_{agg} for Selected Dataset S'
Data Owner
Data User (Delegator)
Data User (Delegatee)

(b) The proposed approach

Figure 2: Multi-Delegation.

encrypt the selected dataset using his own keys (iv) outsource the re-encrypted data to the cloud server, and (v) share the keys with the user (to whom the delegatee wants to delegate his received rights). However, this solution seems secure but such retrieval, as well as the re-encryption process, become infeasible for real-time use. Another possible solution to do multi-delegation in the existing KASE (Cui et al., 2016; Li et al., 2016; Li et al., 2018; Zhou et al., 2018; Padhya and Jinwala, 2018) schemes is as follows: the delegate can ask the data owner to generate a new aggregate key k'_{agg} for selected dataset $S' = \{1,..,n'\}$ and share it with the user. However, this solution requires both data owner and the delegatee to be involved in the delegation process and later on it may result in the tedious task for the data owner. Additionally, the delegatee cannot directly share his aggregate key k_{agg} having search rights for dataset S with other users.

Therefore, to minimize the overhead required for multi-delegation (especially the expensive cost of re-encryption of selected dataset) at the user sides, a better approach is to outsource the computation of re-encryption to the third party. We observe that Proxy Re-Encryption (PRE) (Blaze et al., 1998) can solve the issue of multi-delegation of rights with minimal required overhead at user sides. PRE allows an untrusted proxy to re-encrypt a ciphertext from being encrypted under one key to another key, without learning anything about the underlying plaintext. Furthermore, the re-encrypted ciphertext can be decrypted by the delegatee using his own key (Figure 2b). In the proposed scheme, if the attributes of the delegatee satisfy the hidden access policy (defined by the data owner), the delegatee can further delegate their received rights to other user(s) by requesting a proxy server to re-encrypt the selected data from the shared dataset.

To the best of our knowledge, ours is the first KASE scheme that meets all the following requirements all together viz.: *fine-grained multi-delegation, break-the-glass access control, revocation and privacy preservation simultaneously*.

1.2 Our Contributions

The characteristics and contributions of the BTG-RKASE are as follows:

- *Fine-grained multi-delegation:* The proposal allows a data owner to delegate access and search rights of any set of ciphertexts by sharing an aggregate key with the delegatee. In addition, BTG-RKASE support transitive delegation of rights from the delegatee to other user(s) in a fine-grained manner. Further, the expensive computation of re-encryption required for multi-delegation is outsourced to proxy server.
- *Break-the-glass access control:* In exceptional situation (when the authorized users are not available), the break-glass access mechanism allow timely access to the data, provided the attributes of the user satisfy the break-the-glass policy.
- *Revocation in multi-delegation:* The subject who delegated the rights remains accountable for the

outcome of the delegated privileges. Therefore, the proposed scheme allows the data owner and the delegator (in case of multi-delegation) to revoke any user's delegated rights in case delegatee behaves maliciously or leaves the system.

- *Multi-keyword Search.* We enhance the existing KASE schemes and improve its query expressiveness by supporting multi-keyword search over the shared dataset using a trapdoor of constant size.
- *Privacy preservation:* The proposed BTG-RKASE scheme preserves the privacy, as the hidden access policy defined by the data owner, the attribute token, keyword ciphertext and the query trapdoor do not reveal any information about related attributes or keywords.
- *Improve Query Performance.* The proposed KASE scheme allows searching over the shared dataset S using a single trapdoor Tr, that a query requester submits to the cloud server. The exiting KASE schemes (Cui et al., 2016; Li et al., 2016; Li et al., 2018; Zhou et al., 2018) require $|S|$ number of trapdoor to search over shared dataset S, as the existing KASE schemes generate adjusted trapdoors Tr_l to search over doc_l for each $l \in S$ using the given trapdoor Tr.

1.3 Outline of the Paper

The rest of the paper is organized as follows: The preliminaries are given in Section 2. The system architecture and security model are defined in Section 3. We give a concrete construction of the proposed BTG-RKASE scheme in Section 4. The security analysis of the BTG-RKASE is discussed in Section 5. The empirical evaluation of the proposed scheme is discussed in Section 6. Conclusion and future extensions are provided in Section 7. References are at the end.

2 PRELIMINARIES

In this section, we introduce computational problems, hardness assumptions and notations used throughout this paper. We rely on the standard definitions used for security of cryptographic schemes.

2.1 Computational Assumption

Definition 1 DDH Assumption: The DDH problem in group G of prime order p (according to the security parameter) is a problem for input of a tuple (g,g^a,g^b,g^c,R) where, $a,b,c \in Z_p$ be chosen at random and g be a generator of G,then to decide whether $R= g^{abc}$ or not. An algorithm A has advantage ε in solving DDH problem in G if $Adv_{DDH}(A){:=}|Pr[A(g,g^a,g^b,g^c,g^{abc})=0]-Pr[A(g,g^a,g^b,g^c,R)=0]| \geq \varepsilon(\kappa)$. We say that the DDH assumption holds in G if no Probabilistic Polynomial Time (PPT) algorithm has an advantage of at least ε in solving the DDH problem in G.

Table 1: Notations used in the proposed BTG-RKASE scheme.

Term	**Meaning**
λ	Security parameter
n	Number of documents held by a data owner
B	Bilinear map group system
SP	System parameter
doc_l	l^{th} document
$PubK$	Public parameters
pk	Public key
msk	Master secret key
sk	Secret key
$\mathcal{K}_l$	Break-glass key of document doc_l
BK_l	Encrypted break-glass key of document doc_l
rk	re-encryption key
SK	Attribute token
$Attr$	Set of user's attributes used to generate attribute token SK
l	Index of document or file
S	Subset $S \subseteq \{1,.....,n\}$ contains the indices of documents
KS	Keyword space that involves m different keywords in the system
w_i	i^{th} keyword
$w_{i,j}$	j^{th} value of keyword w_i from it's $\mathcal{N}_i$ different possible values
$\overrightarrow{Q}$	Set of query keywords attached with the query trapdoor Tr
$\overrightarrow{KW}$	Set of keywords attached with the ciphertext $C_{\overrightarrow{KW}}$
M	Message or Plaintext
C	Ciphertext
$C_{\overrightarrow{KW}}$	Keyword ciphertext
C_M	Data ciphertext
δ_l	Public information related to ciphertext of l^{th} document i.e. parameters C_1 and C_2
C^R	Re-encrypted ciphertext
A_l	Access policy attached with ciphertext C_l and defined by data owner
BTG_l	Break-the-glass access policy attached with ciphertext C_l and defined by data owner
k_{agg}	Aggregate key
Tr	Trapdoor used to search query keyword(s) within set $\overrightarrow{Q}$
R	Search result
RL_l	Revocation list of l^{th} document
$\mathcal{U}_l$	The set of revoked users' identity for l^{th} document
U_{QR}	User id of query requester
Π	Multiplicative notation
AV_{att_i}	Set of all possible values for attribute att_i
$av_{atr_{i,j}}$	j^{th} value of attribute att_i in the set AV_{att_i}
$Available_i$	The status of availability of user having id U_i (Available = 1, Not available =0)

2.2 Notations

The list of notations used throughout the paper is given in the Table 1.

3 SYSTEM ARCHITECTURE AND SECURITY MODEL

In this section we present the formal definition of the proposed BTG-RKASE scheme and formalize a security model for it.

3.1 Framework

The proposed BTG-RKASE scheme is an ensemble of randomized polynomial-time algorithms, as discussed further, here. The data owner first sets up an account on the cloud server and establishes the public system parameter via **Setup()**. At the time of registering in the system, the Trusted Authority (TA) assigns a unique id to the user. TA manages users in the system and it is fully trusted by entities in the system. The data owner generates a public, secret and master-secret keys via **Keygen()**. The data owner securely send secret key to the TA. The TA constructs attribute token via **AttrGen()**. Messages can be encrypted via **Encrypt()** algorithm. The data owner generates key for break-the-glass access via **KeyGen_BTG()**. To delegate the search and access rights to a specific subset of data classes, the data owner uses the master-secret to generate a constant size aggregate key via **Extract()**. The generated key can be passed to delegatees securely (via secure e-mails or secure device). The data owner or delegator can take away the delegated privileges via **Revocation()** algorithm. For multi-delegation of received data rights, the delegatee can generate re-encryption key via **ReKeygen()** algorithm. The generated key can be passed to proxy server, using which the proxy server can re-encrypt the requested set of data via **ReEncrypt()**. Finally, any user with an aggregate key can search over shared dataset by generating query trapdoor via **TrapdoorGen()**. On receiving the query trapdoor from the user, the cloud server runs the **Test()** algorithm to retrieve matching documents and send it further to the query requester. The user can access the ciphertext via **Decrypt()** algorithm, provided ciphertext class is within the range of aggregate key. For the break-the-glass access, the user run **BTG_Access()** algorithm, provided the attributes of the user satisfy the break-the-glass policy.

Furthermore, in the following discussion of the proposed scheme, the keyword space contains m different keywords, i.e., $KS = \{w_1,w_2,....,w_m\}$. Each keyword w_i contains $\mathcal{N}_i$ possible values and $w_{i,j}$ represents the j^{th} value of keyword w_i. Let assume that there are total X users in the system and $U = \{U_1,...,U_X\}$ is a set of all users' identities. Therefore, each cloud user can be uniquely identified by his assigned identity $U_{id} \in U$. Additionally, $\mathcal{U}_l \subset U$ represents the set of revoked users' identities for l^{th} document, where $\mathcal{U}_{l_i} \in \mathcal{U}_l$ represents an identity of user whose delegated rights needs to be revoked over l^{th} document. The revocation list $\mathcal{U}_l$ may be empty, which means there is no user needs to be revoked for l^{th} document.

- $SP \leftarrow$**Setup**$(1^\lambda, n)$**:** Takes as input the security parameter λ and the number of documents held by a data owner n. Outputs the public system parameters SP, which is omitted from the input of the other algorithms for brevity.
- $(pk,sk,msk) \leftarrow$**KeyGen():** Outputs a public, secret and master-secret key pair (pk,sk,msk) for a data owner registering in the system.
- $SK \leftarrow$**AttrGen**$(sk,Attr)$**:** Takes as input the secret key sk and the set of attributes $Attr$ of user. Outputs the attribute token SK.
- $C_l \leftarrow$**Encrypt**(pk,sk,l,M,KW,A)**:** Takes as input the the public key pk, secret key sk, document index l, message M, set of related keyword(s) KW, and the hidden access policy A. Outputs the corresponding ciphertext $C_l = (\delta_l, C_M, C_{KW}, C_A)$, where C_M is data-ciphertext, C_{KW} keyword-ciphertext, δ_l public information, C_A ciphertext with the hidden access policy.
- $BK_l \leftarrow$**KeyGen_BTG**(sk,msk,l,BTG)**:** , Takes as input the master-secret key msk, secret key sk, document index l, and the break-the-glass policy BTG. Generates a break-glass key $\mathcal{K}_l$ for document doc_l and then encrypt it using given break-the-glass policy BTG. The algorithm outputs encrypted key BK_l. If user's attribute token SK satisfies break-glass policy BTG then user can recover break-glass key $\mathcal{K}_l$ to access document doc_l.
- $k_{agg} \leftarrow$**Extract**(msk,S)**:** Takes as input the master-secret key msk of data owner and subset of data classes $S \subseteq \{1,2,...,n\}$. Outputs the aggregate key k_{agg} which aggregates the search and access rights of all encrypted messages within set S.
- $RL \leftarrow$**Revocation**$(\mathcal{U}_l,msk,l)$**:** Takes as input the set of revoked users' identities $\mathcal{U}_l$, master-secret key msk, and document index l. Outputs the revocation list $RL_l = (RL_1, RL_2)$.
- $rk \leftarrow$**ReKeygen**(pk,pk')**:** Takes as input the public key pk of data owner and public key pk' of delegatee (who wants to delegate his received rights further to other user(s)). The algorithm generates re-encryption key rk.
- $(C_l^R | \forall l \in S') \leftarrow$**ReEncrypt**$(rk,SK,S')$**:** Takes as input the re-encryption key rk, the attribute token SK and set S' of encrypted documents' indices. Outputs re-encrypted ciphertexts C_l^R iff $SK \models A_l$ $\forall l \in S'$.
- $Tr \leftarrow$**TrapdoorGen**$(k_{agg}, \overrightarrow{Q})$**:** Takes as input an aggregate key k_{agg} and a set of query keywords $\overrightarrow{Q}$. Outputs a single trapdoor Tr.
- $R \leftarrow$**Test**(Tr,S,l,U_{QR})**:** Takes as input a query trapdoor Tr, a set of shared documents' indices S, document index l, and identity of the query requester $U_{QR} \in U$. If U_{QR} is not the revoked user and $l \in S$, Test algorithm returns the search result $R \in \{0,1\}$ by following the rules shown below:

$$R = \begin{cases} 1, \text{If } \overrightarrow{Q} \models \overrightarrow{KW} \\ 0, Otherwise \end{cases}$$

Otherwise, the algorithm returns NULL.

- M ←**Decrypt(**k_{agg}, S, l, C_l**):** Takes as input the ciphertext C_l, the data class l, and the aggregate key k_{agg} corresponding to the set S. The algorithm outputs the decrypted result M if $l \in S$.
- M ←**BTG_Access(**SK, l, U_{QR}**):** Takes as input the attribute token SK, an identity of the query requester U_{QR}, and the data class l. The algorithm outputs the decrypted result M iff $SK \vDash BTG_l$ and U_{QR} is not the revoked user.

3.2 Security Model

We consider the cloud servers and data users to be *honest but curious*, i.e., they follow the given protocols honestly, but try to get some additional information beyond their authorization. However, the capacity of the data user in the system is limited by both storage space and computing power. Moreover, communication channels involving the server are assumed to be insecure.

It is required that the ciphertext does not reveal any information about the corresponding keyword(s) to an adversary. We prove this claim by the security of Indistinguishability against Chosen Keyword Attack (IND-CKA) model. Moreover, given a valid search token, the adversary can only gain information whether the search is successful by Indistinguishability against Keyword Guessing Attack (IND-KGA) model. We use the static corruption model, that means the set of corrupted users has to be defined in the initialization phase. In addition, the adversary is not allowed to ask for re-encryption key from uncorrupted users to corrupted users or vice versa. We introduce the games between an attack algorithm $\mathcal{A}$ and a challenger, both of whom are given input of n, the total number of documents. In the following games, $F(L,T)$ is a binary relation, where L is the set of attributes labeled with user's attribute token SK and T is the hidden access policy attached with ciphertext. $F(L,T) = 1$, if the attributes in L satisfy the access policy specified in T i.e. $L \vDash T$; else, $F(L,T) = 0$.

3.2.1 IND-CKA Model

Init Phase. The adversary $\mathcal{A}$ selects a set of corrupted users denoted by CoList sends it to the challenger. The adversary also commits two ciphertext access policies $\mathcal{T}_0^*$ and $\mathcal{T}_1^*$ and challenge index i_c for which he wishes to be challenged upon.
Setup. The challenger runs Setup($1^\lambda, n$) to generate system parameters. The setup algorithm also defines a keyword space ks and attribute set $ATTR$.

Query Phase 1. The adversary $\mathcal{A}$ adaptively queries $q_1, ..., q_n$ to following oracles and oracle answers in polynomial time.

- **Uncorrupted Key Generation Oracle** O_{pk}**:** Outputs a new key pair (pk, sk, msk) ←Keygen() and give pk to $\mathcal{A}$.
- **Corrupted Key Generation Oracle** O_{msk}**:** Generates (pk, sk, msk) ←Keygen() and issues key pair (pk, sk, msk) to $\mathcal{A}$
- **Attribute Token Generation Oracle** O_{SK}**:** The adversary is allowed to request for polynomially bounded number of attribute token on attributes L with the restriction that $F(L, \mathcal{T}_0^*) = F(L, \mathcal{T}_1^*) = 1$ or $F(L, \mathcal{T}_0^*) = F(L, \mathcal{T}_1^*) = 0$
- **Aggregate Key Generation Oracle** $O_{k_{agg}}$**:** On giving input of (msk, S) by the adversary, where msk is master secret key corresponding to pk and pk is from O_{msk}, then oracle returns an aggregate key k_{agg} ← Extract(msk, S). The restriction is that an adversary can not ask for the aggregate key for challenge index, i.e., if $i_c \in S$, then $O_{k_{agg}}$ returns null.
- **Re-encryption Key Generation Oracle** O_{rk}**:** On giving input of (pk, pk') by the adversary, where pk, pk' are from O_{pk} or O_{msk}, then oracle generates re-encryption key rk ←ReKeygen(pk, pk'). Here, one constraint is required that both the public key pk, pk' are either corrupted or both are uncorrupted, i.e. pk and pk' both are from O_{pk} or O_{msk}. The adversary is not allowed to ask for re-encryption key from uncorrupted users to corrupted users or vice versa.
- **Re-encryption Oracle** O_{re}**:** On giving input of (rk, S'), where re-encryption key rk ←ReKeygen(pk, pk') and pk, pk' are from O_{pk} or O_{msk}, then oracle generates re-encrypted ciphertext ($C_l^R | \forall l \in S'$) ←ReEncrypt(rk, SK, S')
- **Trapdoor Generation Oracle** O_{Tr}**:** It runs Tr ←TrapdoorGen($k_{agg}, \overrightarrow{Q}$), and returns the trapdoor Tr for $\overrightarrow{Q}$ to $\mathcal{A}$.
- **Test Oracle** O_{test}**:** On giving input of trapdoor Tr, set S and document index $l \in S$, the challenger checks k_{agg} corresponding to set S occurs in table $T_{k_{agg}}$, if yes, the oracle returns output of Test(Tr, S, l, U_{QR}) to the adversary. Otherwise, it returns $null$.

Challenge Phase.
The attacker $\mathcal{A}$ sends the challenger two equal length set of keywords $\overrightarrow{KW_0}; \overrightarrow{KW_1}$ on which it wishes to be challenged along with an index i_c, a message M from message space, a public key pk, secret key sk, and

the challenge access policy $\mathcal{T}_0^*$ and $\mathcal{T}_1^*$. Here, the restriction is that if for any attribute token generated by oracle O_{SK} in Phase 1 on an attribute list L, $F(L,\mathcal{T}_0^*) = F(L,\mathcal{T}_1^*) = 1$ then $\overrightarrow{KW_0} = \overrightarrow{KW_1}$. Additionally, the attacker had not previously asked for the query trapdoor corresponding to keywords $\overrightarrow{KW_0};\overrightarrow{KW_1}$ to the oracle O_{Tr} and pk is from the O_{pk}. The challenger chooses β randomly from {0,1} and runs Encrypt(pk,sk,i_c,M,$\overrightarrow{KW_\beta}$,$\mathcal{T}_\beta^*$) and returns keyword ciphertext C_β^* to the adversary.

Query Phase 2. Identical to that in phase 1., the adversary asks for more queries $q_{n+1},.....,q_{n_2}$ except the following oracles:

- **Trapdoor Generation Oracle O_{Tr}:** On giving input of (k_{agg}, $\overrightarrow{Q}$) by the adversary, the challenger answers same as that in phase 1, except the following cases:
  - The adversary had previously ask for the trapdoor corresponding to a set of keywords $\overrightarrow{KW_0}$ or $\overrightarrow{KW_1}$
  - $\overrightarrow{Q} = \overrightarrow{KW_0}$ or $\overrightarrow{Q} = \overrightarrow{KW_1}$

  If one of the above condition holds, then challenger returns *null*.
- **Test Oracle O_{test}:** On input of Test(Tr,S,l,U_{QR}), if $i_c = l$ or $i_c \in S$, then the challenger returns *null*. Otherwise the challenger responds as that in phase 1.

Guess. The adversary $\mathcal{A}$ outputs its guess $\beta' \in \{0,1\}$ for β and wins the game if $\beta = \beta'$.

The advantage of the adversary in this game is defined as $ADV_{\mathcal{A},K}^{\text{BTG-RKASE}} = |Pr[\beta = \beta'] - \frac{1}{2}|$, where the probability is taken over the random bits used by the challenger and the adversary.

Definition 2 (IND-CKA Security): We say that the proposed BTG-RKASE scheme hold the privacy for keyword if $ADV_{\mathcal{A},K}^{\text{BTG-RKASE}}$ is negligible with respect to the security parameter for any polynomial time adversary.

3.2.2 IND-KGA Model

Init Phase. The adversary $\mathcal{A}$ selects a set of corrupted users denoted by CoList sends it to the challenger. The adversary also commits two set of query keywords $\overrightarrow{Q_0}$ and $\overrightarrow{Q_1}$ and challenge index i_c for which he wishes to be challenged upon.

Setup. The challenger runs Setup(1^λ,n) to generate system parameters. The setup algorithm also defines a keyword space ks and attribute set $ATTR$.

Query Phase 1. The adversary $\mathcal{A}$ adaptively queries $q_1,....,q_n$ to oracles and oracle answers in polynomial time. The oracles are identical to that in the IND-CKA security model except the following:

- **Ciphertext Generation Oracle O_C:** The attacker $\mathcal{A}$ sends the challenger set of keywords $\overrightarrow{KW}$ along with an index l, a message M from message space, a public key pk, secret key sk, and access policy A. If $l \neq i_c$, then the challenger runs Encrypt(pk,sk,l,M,$\overrightarrow{KW}$,A) and returns ciphertext C_l to the adversary.

Challenge Phase. The attacker $\mathcal{A}$ sends the challenger two equal length set of keywords $\overrightarrow{Q_0};\overrightarrow{Q_1}$ on which it wishes to be challenged along with an index i_c, and an aggregate key k_{c-agg} corresponding to a set S which includes an index i_c. Here, the restriction is that the attacker had not previously asked for the ciphertext corresponding to keywords $\overrightarrow{Q_0};\overrightarrow{Q_1}$ to the oracle O_C. The challenger chooses β randomly from {0,1} and runs TrapdoorGen(k_{c-agg},$\overrightarrow{Q_\beta}$) and returns query trapdoor Tr_β^* to the adversary.

Query Phase 2. Identical to that in phase 1, the adversary asks for more queries $q_{n+1},.....,q_{n_2}$ except that

- The adversary $\mathcal{A}$ cannot ask for the aggregate key corresponding to a set S which includes an index i_c
- Ciphertext query on keyword set $\overrightarrow{Q_0}$ or $\overrightarrow{Q_1}$ under challenge index i_c is not allowed

Guess. The adversary $\mathcal{A}$ outputs its guess $\beta' \in \{0,1\}$ for β and wins the game if $\beta = \beta'$.

The advantage of the adversary in this game is defined as $ADV_{\mathcal{A},KG}^{BTG-RKASE} = |Pr[\beta = \beta'] - \frac{1}{2}|$, where the probability is taken over the random bits used by the challenger and the adversary.

Definition 3 (IND-KGA Security): We say that the proposed $BTG-RKASE$ method hold the privacy for trapdoor if $ADV_{\mathcal{A},KG}^{BTG-RKASE}$ is negligible with respect to the security parameter for any polynomial time adversary.

3.3 Availability of Subjects

The proposed scheme BTG-RKASE makes break-the-glass access decisions based on the availability of subjects. The definition of availability will be application-specific. In the context of a hospital, a doctor may have access to the patient's data, but may not be physically present in the hospital and unable to respond to any emergency situation of his patient. In the military setting, availability may simply be whether the corresponding user is still alive and present on battlefield.

To identify a set of available authorized subjects at request evaluation time, the proposed scheme maintain a set denoted by $Au_l \subset U$ represents the set of users' identities who has access right for document doc_l, where $Au_{l_i} \in Au_l$ represents an identity of the delegatee who has received the access rights over l^{th} document. The proposed scheme achieves the set Au_l updating through the cloud servers to save the computational overhead of data owner. In case of multi-delegation, the user who further delegates his available rights for doc_l, updates the set Au_l. Furthermore, the proposed scheme also use one more notation $Available_i$ to represent the status of availability of user having id U_i. If $Available_i = 1$, user having id U_i is available, otherwise he is not available ($Available_i = 0$). Therefore, if $Available_i = 0 | \forall Au_{l_i} \in Au_l$, then break-the-glass access on doc_l is possible, provided user's attributes satisfy the break-the-glass policy

4 THE PROPOSED METHOD: BTG-RKASE

In this section, we discuss the construction of the proposed scheme. The attribute space to define access policy A is $ATTR = \{att_1, att_2, .., att_\kappa\}$. Assume AV_{att_i}=$\{av_1, av_2, ...av_{\eta_i}\}$ be the set of all possible values of attribute att_i, where η_i denotes the maximum number of values for i^{th} attribute. Here, $av_{attr_{i,j}}$ represents the j^{th} value of attribute $attr_i$. The other notations used in the forthcoming discussion are already introduced in Section 2. The detailed construction of the proposed BTG-RKASE scheme is as follows:

1. **Setup** (1^λ, n)**:** The data owner runs this algorithm and publish the system parameters SP = (B, $PubK$,H_1,H_2). In the following discussion, bilinear map group system B=$(p, G, G_T, e(.,.))$, where p is the order of G and $2^\lambda \leq p \leq 2^{\lambda+1}$. g is a generator of group G. n is the number of documents D= (doc_1, doc_2, ..., doc_n) that belongs to a data owner. The random element $\alpha \in Z_p$. Algorithm 1 gives a formal description of Setup operation.

Algorithm 1: Setup (1^λ, n) $\rightarrow SP$.

```
1: $B$=$(p, G, G_T, e(.,.))$
2: $PubK \leftarrow (g, g_1, .., g_n, g_{n+2}, ..., g_{2n})$
3: $g_l \leftarrow g^{\alpha^l} \in G | \forall l \in \{1, 2, .., n, n+2, ..., 2n\}$
4: Select two one-way, collision resistant hash functions $H_1 : \{0,1\}^* \times \{0,1\}^* \rightarrow G$, $H_2 : \{0,1\}^* \times \{0,1\}^* \rightarrow G$
5: $SP \leftarrow (B, PubK, H_1, H_2)$
6: $ks \leftarrow^m \{0,1\}^*$
```

2 **Keygen:** The data owner runs this algorithm to generate a key pair (pk, sk, msk)
$pk = v$= g^γ; sk = ρ; msk = γ, where $\gamma \in Z_p$ and $\rho \in Z_p$

3 **AttrGen**(sk,$Attr$) **:** The attribute token generation algorithms takes as input the secret key sk, a set $Attr$ of attributes and generates a attribute token of constant size. Algorithm 2 gives a formal description of AttrGen operation:

Algorithm 2: AttrGen(sk,$Attr$) $\rightarrow SK$.

```
1: Select $r \in_R Z_p$
2: $S_0 \leftarrow g^r$
3: for all $av_{attr_{i,j}} \in Attr$ do
4:     $S_1 \leftarrow g^\rho (\Pi(H_2(av_{attr_{i,j}})))^r$
5: Return $SK = < S_0, S_1 >$
```

4 **Encrypt**($pk, sk, l, M, \overrightarrow{KW}$,$A$)**:** The data owner gives public key, secret key sk, message M, document index l, keywords set $\overrightarrow{KW}$, and access policy A as input. The data owner generates the ciphertext by following the steps as shown in Algorithm 3. Algorithm 3 gives a formal description of Encrypt operation.

Algorithm 3: Encrypt (pk, sk, l, M, KW, A) $\rightarrow C_l$.

```
1: Select $t \in Z_p$
2: if $l \in \{1, 2, ..., n\}$ then
3:     $C_1 \leftarrow g^t$
4:     $C_2 \leftarrow (v \cdot g_l)^t$
5:     $C_3 = C_{\overrightarrow{KW}} \leftarrow \Pi(H_1(w_{i,j}))^t | \forall w_{i,j} \in \overrightarrow{KW}$
6:     $C_4 = C_M \leftarrow M \cdot e(g_1, g_n)^t | M \in G_T$
7:     for all $av_{attr_{i,j}} \in A$ do
8:         $C_5 = (\Pi(H_2(av_{attr_{i,j}})))^t$
9:     $C_6 = l \cdot e(g^t, g^\rho)$
10:    $C_A = (C_5, C_6)$
11: $C_l = (\delta_l, C_{\overrightarrow{KW}}, C_M, C_A) = (C_1, C_2, C_3, C_4, C_5, C_6)$
12: Data owner stores ciphertext $C_l$ on the cloud server.
```

5. **KeyGen_BTG**(sk, msk, l, BTG)**:** The data owner generates the break-glass key by following the steps as shown in Algorithm 4. Algorithm 4 gives a formal description of KeyGen_BTG operation.

Algorithm 4: KeyGen_BTG(sk, msk, l, BTG) $\rightarrow BK_l$.

```
1: Select $\beta \in Z_p$
2: if $l \in \{1,2,...,n\}$ then
3:     $\mathcal{K}_l \leftarrow g_l^{\gamma}$
4:     $bk_1 = g^{\beta}$
5:     for all $av_{attr_{i,j}} \in BTG$ do
6:         $bk_2 = (\Pi(H_2(av_{attr_{i,j}})))^{\beta}$
7:     $bk_3 = \mathcal{K}_l \cdot e(g^{\beta}, g^{\rho})$
8: $BK_l \leftarrow (bk_1, bk_2, bk_3)$
```

6. **Extract** (msk,S)**:** For any subset $S \subseteq \{1,...,n\}$ which contains all the indices of document sets, the algorithm outputs aggregate key k_{agg} by computing:
   $k_{agg} = \Pi_{j \in S} g_j^{\gamma}$
   The data owner securely sends k_{agg} and a set S to the user, to delegate the keyword search and data access rights of the ciphertexts within set S.

7. **Revocation(**$\mathcal{U}_l$**,**msk**,**l**):** The algorithm outputs revocation list RL_l for document l, on giving input of master-secret key and a set of revoked users' identities $\mathcal{U}_l$ whose delegate rights needs to be revoked over document doc_l. The l^{th} document must not be either accessed or searched by any user $\mathcal{U}_{l_i} \in \mathcal{U}_l$. The cloud server checks the user's authorization using RL_l at the time of receiving any request for document l (such as access or search request). Algorithm 4 gives a formal description of Revocation operation.

Algorithm 5: Revocation($\mathcal{U}_l$,msk,l) $\rightarrow RL_l$.

```
1: Select $r \in Z_p$
2: $RL_1 \leftarrow g_l^{r\gamma}$
3: $RL_2 \leftarrow (\Pi_{\mathcal{U}_{l_i} \in \mathcal{U}_l} \mathcal{U}_{l_i}^{\gamma})^r$
4: $RL_l \leftarrow (RL_1, RL_2)$
5: Data owner stores revocation list $RL_l$ on the cloud
   server.
```

8. **ReKeygen(**pk, pk'**):** This algorithm outputs re-encryption key rk by computing:
   $rk = pk'/pk = g^{\gamma'}/g^{\gamma}$

9. **ReEncrypt(**rk, SK, S'**):** On receiving the re-encryption request from DU, proxy server first matches user attribute token SK with the hidden access policy $A_l|\forall l \in S'$ defined by the data owner. If the attribute token SK satisfies the access policy A_l, then the cloud server re-encrypt C_l using the re-encryption key rk and calculates re-encrypted ciphertext C_l^R by following the steps as shown in Algorithm 6. Algorithm 6 gives a formal description of ReEncrypt operation.

Algorithm 6: ReEncrypt (rk, SK, S') $\rightarrow C_l^R|\forall l \in S'$.

```
1: for all $l \in S'$ do
2:     $Y_l \leftarrow \frac{C_6 \cdot e(S_0, C_5)}{e(C_1, S_1)}$
3:         ▷ If $SK \vDash A_l$, then $Y_l = l$ otherwise it gets
   any random value which indicates $SK \nvDash A_l$.
4:     if $Y_l = l$ then
5:         $C_1^R \leftarrow C_1, C_2^R \leftarrow C_2 * rk$
6:         $C_3^R \leftarrow C_3, C_4^R \leftarrow C_4$
7:         $C_5^R \leftarrow C_5, C_6^R \leftarrow C_6$
8:     Stores the ciphertext $C_l^R$ on the cloud server
```

10. **TrapdoorGen(**k_{agg}, $\overrightarrow{Q}$**):** The user who wants to perform the keyword search over the shared data, runs this algorithm and generates a single aggregate trapdoor Tr. If the set S of documents are in the range of an aggregate key k_{agg}, then the user having trapdoor Tr can search over any document in the range of set S. Algorithm 7 gives a formal description of TrapdoorGen() operation.

Algorithm 7: TrapdoorGen(k_{agg}, Q)$\rightarrow Tr$.

```
1: Select $b \in Z_p$
2: $Tr_0 \leftarrow \{k_{agg} \cdot \Pi(H(w_{i,j}))^b\}_{w_{i,j} \in \overrightarrow{Q}}$
3: $Tr_1 \leftarrow g^b$
4: $Tr \leftarrow (Tr_0, Tr_1)$
5: The user submits $(Tr, S)$ to the cloud server.
```

11. **Test** (Tr, S, l, U_{QR})**:** To check if document doc_l contains query keyword(s) within set $\overrightarrow{Q}$ or not, the cloud server follows steps as shown in Algorithm 8. Algorithm 8 gives a formal description of Test operation. In the following discussion $U_{QR} \in U$ is an identity of query requester and $\mathcal{U}_l$ is a set of revoked users' identities corresponding to list RL_l.

12. **Decrypt** (k_{agg},S, l, C_l)**:** If $l \notin S$, Decrypt algorithm outputs NULL. Otherwise, returns the output:
    $M = C_4$ e(k_{agg} pub_{1_l} pub_{2_l} , C_1) /e(pub_3, C_2) e(pub_4, C_1)
    $\{pub_{1_l}\}_{l \in S} = \{\Pi_{j \in S}\, g_{j+l}\}_{l \in S}$
    $\{pub_{2_l}\}_{l \in S} = \{\Pi_{j \in S, j \neq l}\, g_{n+1-j+l}\}_{l \in S}$
    $pub_3 = \Pi_{j \in S} g_j$
    $pub_4 = \Pi_{j \in S} g_{n+1-j+l}$
    For efficiency consideration, the parameters $(\{pub_{1_l}, pub_{2_l}\}_{l \in S}, pub_3, pub_4)$ for the set S is computed only once.

Algorithm 8: Test $(Tr, S, l, U_{QR}) \rightarrow R \in \{0,1\}$.

```
1: if Revoked($U_{QR}, RL_l, \mathcal{U}_l, l$) $\neq 1$ then
2:     if $l \in S$ then
3:         $pub'_1 = \Pi_{j \in S}\, g_{j+l}$
4:         $pub'_2 = \Pi_{j \in S} g_j$
5:         $R$ = e($Tr_0 \cdot pub'_1, C_1$) / e($pub'_2, C_2$) e($C_{KW_1}, Tr_1$)
6:         if $R = 1$ then
7:             Add $doc_l$ to the search result list
8:             Return $R$
9: if $l \notin S$ then
10:     Return Failure
11: if Revoked($U_{QR}, RL_l, \mathcal{U}_l, l$) $= 1$ then
12:     Return Failure
13: FUNCTION Revoked($U_{QR}, RL_l, \mathcal{U}_l, l$)
14: $pub' \leftarrow \Pi_{\mathcal{U}_{l_i} \in \mathcal{U}_l, \mathcal{U}_{l_i} \neq U_{QR}} \mathcal{U}_{l_i}$
15: $Y \leftarrow e(RL_{l_2}, g_l)/e(U_{QR} \cdot pub', RL_{l_1})$
16: Return $Y$
```

13. **BTG_Access(SK, l, U_{QR}):** On giving input of the attribute token SK, identity of query requester U_{QR}, and the data class l, the cloud server follows steps as shown in Algorithm 9. Algorithm 9 gives a formal description of BTG_Access operation.

Algorithm 9: BTGAccess(SK, l, U_{QR})$\rightarrow M$.

```
1: if Revoked($U_{QR}, RL_l, \mathcal{U}_l, l$) $\neq 1$ then
2:     if $Available_i = 0 | \forall Au_{l_i} \in Au_l$ then
3:         Take the attribute token $SK = (S_0, S_1)$ of user $U_{QR}$ and match $SK$ with the break-glass policy of $l^{th}$ document $BTG_l$.
4:         if $SK \models BTG_l$ then
5:             $\mathcal{K}_l \leftarrow \frac{bk_3 \cdot e(S_0, bk_2)}{e(bk_1, S_1)}$
6:             Using $\mathcal{K}_l$ user $U_{QR}$ can access $C_l$.
7:             $M$ = $C_4$ e($\mathcal{K}_l$ $pub_{1_l}$ $pub_{2_l}$ , $C_1$) /e($pub_3$, $C_2$) e($pub_4$, $C_1$)
8:                 ▷ Here, while computing parameters ($pub_{1_l}, pub_{2_l}, pub_3, pub_4$), the set $S$ contains only one element, i.e., $l$ itself.
9:     elseReturn Failure
10: if Revoked($U_{QR}, RL_l, \mathcal{U}_l, l$) $= 1$ then
11:     Return Failure
12: FUNCTION Revoked($U_{QR}, RL_l, \mathcal{U}_l, l$)
13: $pub' \leftarrow \Pi_{\mathcal{U}_{l_i} \in \mathcal{U}_l, \mathcal{U}_{l_i} \neq U_{QR}} \mathcal{U}_{l_i}$
14: Return $Y \leftarrow e(RL_{l_2}, g_l)/e(U_{QR} \cdot pub', RL_{l_1})$
```

4.1 Analysis of the BTG-RKASE

The correctness of Test algorithm can be realized as follows:

If the query requester U_{QR} is revoked user, then

$$= e(RL_2, g_l)/e(U_{QR} \cdot pub', RL_1)$$

$$= e((\Pi_{\mathcal{U}_{l_i} \in \mathcal{U}_l} \mathcal{U}_{l_i})^{\gamma r}, g_l)/e(U_{QR} \cdot \Pi_{\mathcal{U}_{l_i} \in \mathcal{U}_l, \mathcal{U}_{l_i} \neq U_{QR}} \mathcal{U}_{l_i}, g_l^{\gamma r}) \quad = 1$$

Therefor, if identity of query requester is within set of revoked users then Test algorithm outputs failure. Then, user is no longer permitted to search on the l^{th} document.

Now, consider the case $U_{QR} \notin \mathcal{U}_l$, i.e., query requester is not the revoked user. If the user's query trapdoor Tr matches with the keyword ciphertext $C_{\overrightarrow{KW}}$, then Test algorithm outputs:

e($Tr_0 \cdot pub'_1, C_1$) / e(pub'_2, C_2) e(C_{KW_1}, Tr_1)

$$= \frac{e(Tr_0 \cdot pub'_1, C_1)}{e(pub'_2, C_2)\; e(C_{KW_1}, Tr_1)}$$

$$= \frac{e(k_{agg} \cdot \Pi(H(w_{i,j}))^b \cdot \Pi_{j \in S} g_{j+l}, g^t)}{e(\Pi_{j \in S} g_j, (v \cdot g_l)^t)\; e(\Pi(H(w_{i,j})))^t, g^b)}$$

$$= \frac{e(k_{agg}, g^t)\; e(\Pi(H(w_{i,j})))^b, g^t)\; e(\Pi_{j \in S} g_{j+l}, g^t)}{e(\Pi_{j \in S} g_j, g^{\gamma t})\; e(\Pi_{j \in S} g_j, g_l^t)\; e(\Pi(H(w_{i,j})))^t, g^b)}$$

$$= 1$$

The correctness of BTG_Access algorithm can be realized as follows:

If the user's attribute token SK satisfies the break-glass policy BTG of the document doc_l, then BTG_Access algorithm outputs

$$\frac{bk_3 \cdot e(S_0, bk_2)}{e(bk_1, S_1)}$$

$$= \frac{\mathcal{K}_l \cdot e(g,g)^{\beta\rho}\; e(g^r, (\Pi(H_2(av_{attr_{i,j}})))^{\beta})}{e(g^{\beta}, g^{\rho}(\Pi(H_2(av_{attr_{i,j}})))^r)}$$

$$= \mathcal{K}_l$$

Using $\mathcal{K}_l$ user can access document doc_l. BTG_Access outputs as follows:

C_4 e($\mathcal{K}_l$ pub_{1_l} pub_{2_l} , C_1) /e(pub_3, C_2) e(pub_4, C_1)

$$= \frac{C_4 \quad e(g_l^{\gamma} \cdot \Pi_{j \in S} g_{j+l} \cdot \Pi_{j \in S, j \neq l} g_{n+1-j+l}, g^t)}{e(\Pi_{j \in S} g_j, (v \cdot g_l)^t)\; e(\Pi_{j \in S} g_{n+1-j+l}, g^t)}$$

$$= \frac{C_4 \quad e(g_l^{\gamma}, g^t)\; e(\Pi_{j \in S} g_{j+l}, g^t)\; e(\Pi_{j \in S, j \neq l} g_{n+1-j+l}, g^t)}{e(\Pi_{j \in S} g_j, g^{\gamma t})\; e(\Pi_{j \in S} g_j, g_l^t)\; e(\Pi_{j \in S} g_{n+1-j+l}, g^t)}$$

$$= \frac{M \quad e(g_1, g_n)^t \quad \frac{e(\Pi_{j \in S} g_{n+1-j+l}, g^t)}{e(g_{n+1}, g^t)}}{e(\Pi_{j \in S} g_{n+1-j+l}, g^t)}$$

$$= \frac{M \quad e(g_1, g_n)^t}{e(g_n, g_1)^t}$$

$$= M$$

5 SECURITY ANALYSIS

Theorem 1. If the decisional DDH assumption hold in (G, G_T), then the proposed BTG-RKASE scheme preserves the keyword privacy in the random oracle model.

Proof.

We consider a challenger $\mathcal{C}$, a simulator $\mathcal{SIM}$ and a polynomial-time adversary $\mathcal{A}$. We assume that the adversary $\mathcal{A}$ has a non-negligible advantage $\varepsilon(l)$ to break the privacy of our scheme. Then, we can construct simulator $\mathcal{SIM}$ that breaks the decisional DDH problem ς= $(g,\ g^a,\ g^b,\ g^c,\ R)$ with the probability $Pr|A\text{breaks decisional DDH}| \geq 1/2 + \frac{\varepsilon(1-(q_{re}+q_{test}+q_{dec})\cdot\tau - Pr|A\text{breaks break-the-glass policy}|}{2e\cdot q_{Tr}}$

Here, q_{re}, q_{test}, q_{Tr} are the number of queries to re-encryption oracle, test oracle, and trapdoor generation oracle, respectively. τ is the maximum probability that any given signature verification token is output by G and it is assumed to be very negligible. Moreover, $Pr|A\text{breaks break-the-glass policy}|$ is also very negligible as per our assumption.

On DDH input (g, g^a, g^b, g^c, R), simulator $\mathcal{SIM}$ aims to decide if $R = g^{abc}$.

The challenger $\mathcal{C}$ generates $a, b, c, z \in_R Z_p$, bilinear groups G, G_T with prime order p and the mapping $G \times G \rightarrow G_T$. g is a generator of group G. The challenger $\mathcal{C}$ computes X as follows:

$$R = \begin{cases} g^{abc}, (X=0) \\ g^{z}, (X=1) \end{cases}$$

The challenger gives instance $(g,\ g^a,\ g^b, g^c,\ R) \in G_T$ to simulator $\mathcal{SIM}$.

$\mathcal{SIM}$ interacts with $\mathcal{A}$ ($\mathcal{SIM}$ simulates the $\mathcal{C}$ for $\mathcal{A}$) and starts the simulation as follows.

Init Phase. The adversary $\mathcal{A}$ selects a set of corrupted users denoted by CoList sends it to the challenger. The adversary also commits two ciphertext access policies $\mathcal{T}_0^*$ and $\mathcal{T}_1^*$ and challenge index i_c for which he wishes to be challenged upon.

Setup. The simulator $\mathcal{SIM}$ generates system parameters SP=$(g,\ g_1, \dots, g_n, g_{n+2}, \dots, g_{2n})$.

Here, $g_l = (g^a)^{\alpha^l} \in G$ for $l = \{1, 2, \dots, n, n+2, \dots, 2n\}$.

Moreover, $\mathcal{SIM}$ simulates the hash oracles for keyword and attribute as follows:

- O_{H_1} : Given a keyword w_i, having value $w_{i,j}$, the hash function proceeds as follows:
  - If $w_{i,j}$ has not been queried before, then $\mathcal{SIM}$ toss a random coin $c_i \in \{0,1\}$ with the probability that $Pr|c_i = 0| = 1/(q_T+1)$, where q_T is very large number. We require that q_T should be larger than the number of oracle queries for $O_{k_{agg}}, O_{Tr}, O_{test}$. If $c_i = 0$, then selects $f \in Z_p$ and computes $T^*_{w_{i,j}} = (g^f)^c$. Otherwise, computes $T^*_{w_{i,j}} = (g^f)^{bc}$. Add the tuple $< w_i, w_{i,j}, c_i, T^*_{w_{i,j}} >$ to table T_{kw} and return $T^*_{w_{i,j}}$.
  - Otherwise, retrieve $T^*_{w_{i,j}}$ from table T_{kw} with respect to $w_{i,j}$ and return $T^*_{w_{i,j}}$
- O_{H_2} : Given a attribute value $av_{attr_{i,j}}$, it proceeds as follows:
  - If $av_{attr_{i,j}}$ has not been queried before, then toss a random coin $c_i \in \{0,1\}$. If $c_i = 0$, then selects a random value $h \in_R Z_p$ and computes $T_{av_{attr_{i,j}}}$ = $(g^h)^c$. Otherwise, computes $T_{av_{attr_{i,j}}} = (g^h)^{bc}$. Add the tuple $< attr, c_i, v_{i,j}, T_{av_{attr_{i,j}}} >$ to table T_{attr} and return T_{attr}.
  - Otherwise, retrieve $T_{av_{attr_{i,j}}}$ from table T_{attr} with respect to $attr$ and return $T_{av_{attr_{i,j}}}$

Query Phase 1. The simulator $\mathcal{SIM}$ constructs following oracles and adversary $\mathcal{A}$ can adaptively queries $q_1, \dots., q_n$ to these oracles in polynomial for multiple times.

- **Uncorrupted Key Generation Oracle O_{pk}:** On input an index i, if $corrupted_i = 1$, then $\mathcal{SIM}$ sets public-key $pk_i = v = g^{\gamma_i}$ where $\gamma_i \in_R Z_p$; otherwise, $\mathcal{SIM}$ computes $pk_i = v = (g^c)^{\gamma_i}$. Finally, $\mathcal{SIM}$ records the tuple $< pk_i, \gamma_i, corrupted_i >$ in table T_k and responds $\mathcal{A}$ with pk_i and $sk_i = \rho_i$ where $\rho_i \in_R Z_p$. Table T_k is used to records the outputs of O_{pk} and O_{msk} and these records will be used in other oracles to response the queries.
- **Corrupted Key Generation Oracle O_{msk}:** On input of pk_i, $\mathcal{SIM}$ checks whether pk_i occurs in T_k, if not, $\mathcal{SIM}$ terminates. Otherwise, if $corrupted_i = 0$, $\mathcal{SIM}$ terminates. If $corrupted_i = 1$, $\mathcal{SIM}$ responds $\mathcal{A}$ with $(pk_i = g^{\gamma_i}, msk_i = \gamma_i, sk_i = \rho_i)$, and records the tuple $< pk_i, \gamma_i, corrupted_i >$ in table T_k
- **Attribute Token Generation Oracle O_{SK}:** Whenever $\mathcal{A}$ makes its i^{th} attribute token generation query for the set L_i of attributes such that $F(L_i, \mathcal{T}_0^*) = F(L_i, \mathcal{T}_1^*) = 1$ or $F(L_i, \mathcal{T}_0^*) = F(L_i, \mathcal{T}_1^*) = 0$, i.e., $F(L_i, \mathcal{T}_0^*) = F(L_i, \mathcal{T}_1^*)$. Specifically, $\mathcal{A}$ is allowed to issue a valid attribute token generation query which can satisfy challenge access policy, with the restriction that the token should satisfy both the challenge access structure $\mathcal{T}_0^*$ and $\mathcal{T}_1^*$. The simulator $\mathcal{SIM}$ selects a random values $r_i \in_R Z_p$ and computes the attribute token as follows:

$SK =< S_0, S_1 >= < g^{r_i}, g^{\rho_i}(\Pi(H_2(av_{attr_{j,k}})))^{r_i} > |\forall av_{attr_{j,k}} \in L_i$

- **Aggregate Key Generation Oracle $O_{k_{agg}}$:** On giving input of (msk_i, S) by the adversary,where msk_i is master secret key corresponding to pk_i, the simulator checks that key pair (pk_i, msk_i) occurs in table T_k, if not, $\mathcal{SIM}$ reports failure and terminates. Otherwise, it returns an aggregate key $k_{agg_i} \leftarrow$ Extract(msk_i, S) and add the tuple $< k_{agg_i}, msk_i, S >$ in table $T_{k_{agg}}$. The adversary $\mathcal{A}$ cannot ask for the aggregate key corresponding to a set S that includes an index i_c.
- **Re-encryption Key Generation Oracle O_{rk}:** On giving input of (pk_i, pk_j) by the adversary, $\mathcal{SIM}$ checks whether pk_i, pk_j occur in T_k,if not, $\mathcal{SIM}$ terminates. Otherwise $\mathcal{SIM}$ does the following operations:
  - If $corrupted_i = corrupted_j$, $\mathcal{SIM}$ responds $\mathcal{A}$ with pk_j/pk_i
  - If $corrupted_i \neq corrupted_j$, $\mathcal{SIM}$ aborts
- **Re-encryption Oracle O_{re}:** On giving input of (rk, S'), where re-encryption key $rk \leftarrow$ReKeygen(pk, pk'), $\mathcal{SIM}$ checks whether pk_i, pk_j occur in T_k and pk, pk' are from O_{pk} or O_{msk},if not, $\mathcal{SIM}$ terminates. Otherwise, $\mathcal{SIM}$ performs as following:
  - If $corrupted_i = corrupted_j$, $\mathcal{SIM}$ responds $\mathcal{A}$ with re-encrypted ciphertext $(C_l^R | \forall l \in S') \leftarrow$ ReEncrypt(rk, SK, S')
  - If $corrupted_i \neq corrupted_j$, $\mathcal{SIM}$ aborts
- **Trapdoor Generation Oracle O_{Tr}:** On input $(k_{agg_i}, \overrightarrow{Q})$, the simulator proceeds as follows:
  - It queries O_{H_1} $\forall w_{j,k} \in \overrightarrow{Q}$ and obtain $(w_{j,k}, c_j, w_{j,k}, T^*_{w_{j,k}})$
  - If $corrupted_i = 1$, set trapdoor $Tr \leftarrow TrapdoorGen(k_{agg_i}, \overrightarrow{Q})$
  - If $corrupted_i = 0 \wedge c_j = 1$, set trapdoor $Tr \leftarrow TrapdoorGen(k_{agg_i}, \overrightarrow{Q})$
  - Otherwise, report failure and terminate.
- **Test Oracle O_{test}:** On giving input of $(Tr \leftarrow TrapdoorGen(k_{agg_i}, \overrightarrow{Q}), S, l)$, $\mathcal{SIM}$ checks k_{agg_i} occurs in table $T_{k_{agg}}$, if yes, $\mathcal{SIM}$ returns output of Test(Tr, S, l, U_{QR}) to the adversary. Otherwise, $\mathcal{SIM}$ returns *null*.

Challenge Phase. The attacker $\mathcal{A}$ sends the challenger two equal length set of keywords $\overrightarrow{KW_0}; \overrightarrow{KW_1}$ on which it wishes to be challenged along with an index i_c, a message M from message space, a public key pk^*, secret key sk, and the challenge access policy $\mathcal{T}_0^*$ and $\mathcal{T}_1^*$. Here, the restriction is that if for any attribute token generated in Phase 1 on an attribute list L, $F(L, \mathcal{T}_0^*) = F(L, \mathcal{T}_1^*) = 1$ then $\overrightarrow{KW_0} = \overrightarrow{KW_1}$. If pk^* is not in table T_k or $corrupted^* = 1$, then simulator $\mathcal{SIM}$ terminates. If $H_1(\overrightarrow{KW_0}) = 1$ and $H_1(\overrightarrow{KW_1}) = 1$, then the simulator returns 'failure' and aborts. Otherwise, the simulator $\mathcal{SIM}$ chooses $d \in \{0, 1\}$ as follows:

- If $H_1(\overrightarrow{KW_0}) = 1$ and $H_1(\overrightarrow{KW_1}) = 0$, then set $d = 1$
- If $H_1(\overrightarrow{KW_0}) = 0$ and $H_1(\overrightarrow{KW_1}) = 1$, then set $d = 0$
- otherwise, let $d \leftarrow^R \{0, 1\}$

The challenger runs Encrypt($pk, sk, i_c, M, \overrightarrow{KW_d}, \mathcal{T}_d^*$) and returns ciphertext C_d^* to the adversary.

The Challenger $\mathcal{C}$ sets $t = a$ and computes ciphertext for $\overrightarrow{KW_d}$ and access policy $\mathcal{T}_d^*$ as follows:

$C_1^* = g^a$; $C_2^* = (v \cdot g_l)^a$

$C_3^* = C_{\overrightarrow{KW}} = \Pi(H_1(w_{i,j}))^a | \forall w_{i,j} \in \overrightarrow{KW_d}$

$C_4^* = C_M = M\, \mathrm{e}(g_1, g_n)^a$

$C_5^* = \Pi(H_2(av_{attr_{i,j}})))^a | \forall attr_{i,j} \in \mathcal{T}_d^*$

$C_6^* = i_c \cdot e(g^a, g^\rho)$

Query Phase 2. Same as Phase 1, the adversary asks for more queries $q_{n+1}, \ldots\ldots, q_{n_2}$ except the following oracles:

- **Trapdoor Generation Oracle O_{Tr}:** On giving input of $(k_{agg}, \overrightarrow{Q})$ by the adversary, the challenger answers same as that in phase 1, except the following cases:
  - The adversary had previously ask for the trapdoor corresponding to a set of keywords $\overrightarrow{KW_0}$ or $\overrightarrow{KW_1}$
  - $\overrightarrow{Q} = \overrightarrow{KW_0}$ or $\overrightarrow{Q} = \overrightarrow{KW_1}$

  If one of the above condition holds, then challenger returns *null*.
- **Test Oracle O_{test}:** On input of Test(Tr, S, l, U_{QR}), if $i_c = l$ or $i_c \in S$, then the challenger returns *null*. Otherwise the challenger responds as that in phase 1

Guess. The adversary $\mathcal{A}$ outputs its guess $d' \in$ {0,1} for d.

If $d' = d$, then $\mathcal{SIM}$ outputs $X' = 0$. If $d' \neq d$, then $\mathcal{SIM}$ outputs $X' = 1$.

This completes the simulation.

If $\mathcal{SIM}$ does not report failure and it receives DDH instances as input, the challenge ciphertext is a valid encryption of $(pk^*, i_c, M, \overrightarrow{KW_d}, \mathcal{T}_d^*)$ and $\mathcal{A}$ guesses $d' = d$ with the advantage l. If $\mathcal{SIM}$ does not receives DDH instances as input, the challenge ciphertext does not contain information about $\overrightarrow{KW_d}$ since keyword ciphertext is uniformly distributed in G_T and independent of d. The adversary $\mathcal{A}$ guesses $d' = d$ with exactly 1/2 probability.

We analyze the probability that the simulator does not report failure or abort during the simulation.

When $\mathcal{A}$ asks for trapdoor to oracle O_{Tr}, it happens $H_1(kv_{w_{j,k}}) = 0$ for some keyword $w_{j,k}$ and it makes simulator to respond with failure. The probability that the simulator $\mathcal{SIM}$ aborts is $Pr[H_1(kv_{w_{j,k}}) = 0]= 1/(q_{Tr}+1)$. Therefore, as $\mathcal{A}$ makes at most q_{Tr} oracle queries, the probability of the simulator not reporting failure and aborting is $(1-1/(q_{Tr}+1))^{q_{Tr}} \geq 1/e$

The only situation that $\mathcal{SIM}$ reports failure during challenge phase is that $H_1(\overrightarrow{KW_0}) = 1$ and $H_1(\overrightarrow{KW_1}) = 1$. Since $Pr[H_1(\overrightarrow{KW_d}) = 0]= 1/(q_{Tr}+1)$ for $d \in \{0,1\}$ and the values of $H_1(\overrightarrow{KW_0})$ and $H_1(\overrightarrow{KW_1})$ are independent of $\mathcal{A}$. Furthermore, these two values are independent of each other. Hence, we have

$Pr|H_1(\overrightarrow{KW_0}) = 1 \wedge H_1(\overrightarrow{KW_1}) = 1| = (1-1/(q_{Tr}+1))^2 \leq 1-1/q_{Tr}$

Hence, the probability that the simulator has no failure state is at least $1/q_{Tr}$.

Therefore, the simulator simulates without failure with the probability at least $1/e+1/q_{Tr}$.

Let us analyze the advantage of the simulator solving DDH problem on condition that the simulation completes perfectly.

Based on this, there will be two cases as follows:

case(i): if $X = 0$, then Z = g^{abc} and hence challenge ciphertext is a correct ciphertext of keyword $\overrightarrow{KW_d}$. Then, the probability of $\mathcal{A}$ guessing correctly $d = d'$ is $1/2+\varepsilon$.

case(ii): if $X = 1$, then the challenge ciphertext is independent of $\overrightarrow{KW_0}$ and $\overrightarrow{KW_1}$, so that $\mathcal{A}$ can not obtain any information of d. Then, the probability of $\mathcal{A}$ guessing correctly $d = d'$ is $1/2$.

From (i) and (ii), it follows that $\mathcal{SIM}$'s advantage in this DDH game can be computed as:

$$= \frac{1}{2}(\frac{1}{2}+\varepsilon) + \frac{1}{2}\frac{1}{2} \qquad = \frac{1}{2}+\frac{\varepsilon}{2}$$

Therefore, if the $\mathcal{A}$ has a non-negligible advantage ε in the above game then we can build a simulator ($\mathcal{SIM}$) which can break the DDH problem with non negligible quantity $= (\frac{1}{e}+\frac{1}{q_{Tr}})\frac{\varepsilon}{2}$, which is an intractable problem. □

Theorem 2. If the decisional DDH assumption hold in (G, G_T), then the proposed BTG-RKASE scheme preserves the trapdoor privacy in the random oracle model.

Proof. The proof of Theorem 2 is similar to the proof of Theorem 1.

6 EMPIRICAL EVALUATION

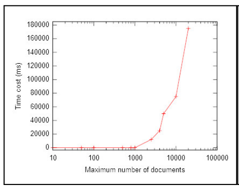
In this section, we evaluate and analyze the computational cost of different algorithms of the proposed scheme.

6.1 Implementation Details

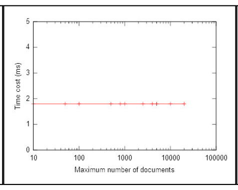
In order to evaluate the performance, we implement the prototype of our scheme and conduct the experiments using Java Pairing Based Cryptographic (JPBC) library (De Caro and Iovino, 2011). We conducted experiments as follows: client implementations were executed in a personal computer with 64 bit Intel(R) Core(TM) i5 -7200U CPU @ 2.50 GHz with Windows10 OS; server implementations were deployed in the Amazon AWS cloud, using an EC2 M5 large instance. Communications were performed on a 10MB/s connection, with 26.932ms round-trip time. We used the official Medicare.gov Hospital Compare datasets provided by the Centers for Medicare and Medicaid Services (Hos, 2019). Each document is fixed to 10 KB with random words chosen from a dictionary and the query is also containing some random keywords. We use Type A pairing for evaluation and it is the fastest (symmetric) pairing among all types of curves. Type A pairings are constructed on the curve $y^2 = x^3 + x$ over the field F_q for some prime $q = 3 \bmod 4$.

6.2 Experimental Results

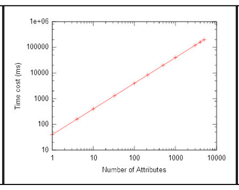
The time cost of Setup() is linear in the maximum number of documents belonging to the data owner (Figure 3a). Figure 3b shows the computational cost of KeyGen() approximate to a constant. The computational cost of AttrGen() is linear in number of attributes, as shown in Figure 3c. The results given in Figure 3d and Figure 3e show that the computational overhead of Encrypt() is linear in the number of keywords to be attached with the ciphertext and number of attributes in the access policy. Figure 3f show that the computational overhead of KeyGen_BTG() is linear in the number of attributes in the break-glass policy. Figure 3g shows the computational time of Extract() is linear in the number of shared documents. The computational cost of Revocation() is linear in the number of revoked users, as



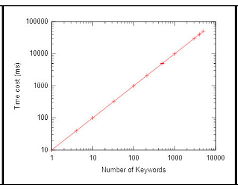
(a) Time cost of Setup()



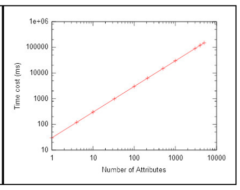
(b) Time cost of KeyGen()



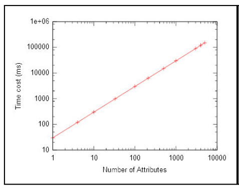
(c) Time cost of AttrGen()



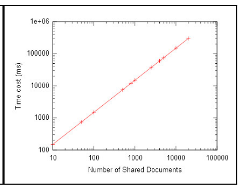
(d) Time cost of Encrypt() based on number of keywords



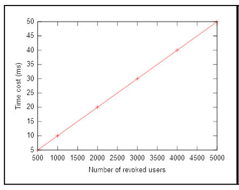
(e) Time cost of Encrypt() based on number of attributes in access policy



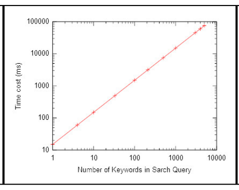
(f) Time cost of KeyGen_BTG() based on number of attributes in break-glass policy



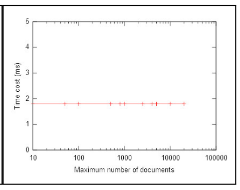
(g) Time cost of Extract()



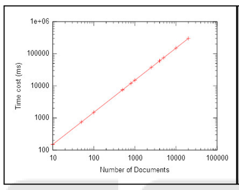
(h) Time cost of Revocation()



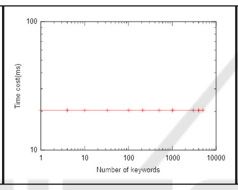
(i) Time cost of TrapdoorGen()



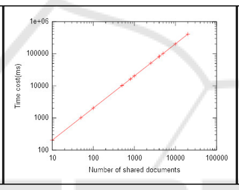
(j) Time cost of ReKeygen()



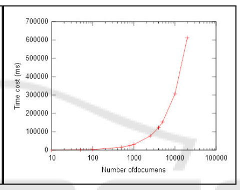
(k) Time cost of ReEncrypt()



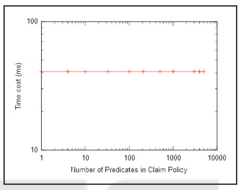
(l) Time cost of Test() based on number of keywords



(m) Time cost of Test() based on number of shared documents



(n) Time cost of Decrypt()



(o) Time cost of BTG_Access()

Figure 3: The computational cost of BTG-RKASE algorithms().

shown in Figure 3h. Figure 3i shows the computational time of TrapdoorGen() is linear in the number of keywords within search query set. Figure 3j shows the computational time of ReKeygen() approximate to a constant. Figure 3k shows that the computational time of ReEncrypt() is linear in number of documents to be re-encrypted. The execution time of Test() is linear in the number of shared documents (Figure 3m), whereas in case of multi-keyword search the computational time of Test() remains constant with respect to the number of keywords in the query set (Figure3l). Figure 3n show that the cost of decryption is in linear with the number of documents. Figure 3o show the computational time of verification of break-the-glass policy is constant as it require only 2 pairing operations, independent of number of attributes.

7 CONCLUSIONS AND FUTURE EXTENSION

The main contribution of this paper is to provide a mechanism which can handle emergency situations where no authorized user exists to perform (or to delegate) a time-critical task. The proposed BTG-RKASE system supports three ways for searchable group data sharing and accessing encrypted dataset: delegation, policy-based fine-grained multi-delegation and break-the-glass access. In normal situations, the authorized user with the delegated key can access the shared dataset, provided ciphertext class is within the range of aggregate key. In fine-grained multi-delegation mode, if the attributes of the delegatee satisfy the hidden access policy (defined by the data owner), the delegatee can delegate his received rights to another user. The expensive computation of re-encryption required for multi-delegation is outsourced to proxy server without compromising the data confidentiality. In exceptional situation (when the authorized users are not available), the break-glass access mechanism allow timely access to the data, provided the attributes of the user satisfy the break-the-glass policy. The proposed scheme also allows revocation of delegated rights even in case of multi-delegation. The other contribution of the proposed approach is that it supports multi-keyword search over the shared dataset using a single trapdoor (instead of multiple trapdoors). Our performance evaluation and simulation results indicated that the proposed method is efficient and secure. In future, we plan to design an trust and risk-based auto-delegation mechanism which can respond automatically to the absence of appropriately authorized users, after finding the legitimacy of the user (based on trust and risk value) for emergency access.

ACKNOWLEDGMENTS

The second author acknowledges the support received from Indian Institute of Technology Jammu under the Faculty Professional Development Grant and the first author acknowledges the Visveswaraya PhD Support Scheme at Sardar Vallabhbhai National Institute of Technology, Surat.

REFERENCES

(2019). Hospital compare. *available at https://data.medicare.gov/data/hospital-compare, accessed on February 13, 2019.*

Ardagna, C. A., Di Vimercati, S. D. C., Foresti, S., Grandison, T. W., Jajodia, S., and Samarati, P. (2010). Access control for smarter healthcare using policy spaces. *Computers & Security*, 29(8):848–858.

Ardagna, C. A., di Vimercati, S. D. C., Grandison, T., Jajodia, S., and Samarati, P. (2008). Regulating exceptions in healthcare using policy spaces. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 254–267. Springer.

Banu, A. S. (2015). Efficient data sharing in cloud medium with key aggregate cryptosystem. *Networking and Communication Engineering*, 7(3):118–121.

Blaze, M., Bleumer, G., and Strauss, M. (1998). Divertible protocols and atomic proxy cryptography. *Advances in Cryptology—EUROCRYPT'98*, pages 127–144.

Cheon, J. H. (2006). Security analysis of the strong diffie-hellman problem. In *Advances in Cryptology - EUROCRYPT 2006*, pages 1–11. Springer Berlin Heidelberg.

Chu, C.-K., Chow, S. S., Tzeng, W.-G., Zhou, J., and Deng, R. H. (2014). Key-aggregate cryptosystem for scalable data sharing in cloud storage. *IEEE transactions on parallel and distributed systems*, 25(2):468–477.

Crampton, J. and Khambhammettu, H. (2008). On delegation and workflow execution models. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 2137–2144. ACM.

Cui, B., Liu, Z., and Wang, L. (2016). Key-aggregate searchable encryption (kase) for group data sharing via cloud storage. *IEEE Transactions on computers*, 65(8):2374–2385.

Dang, H., Chong, Y. L., Brun, F., and Chang, E.-C. (2016). Practical and scalable sharing of encrypted data in cloud storage with key aggregation. In *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, pages 69–80. ACM.

De Caro, A. and Iovino, V. (2011). jpbc: Java pairing based cryptography. In *Computers and communications (ISCC), 2011 IEEE Symposium on*, pages 850–855. IEEE.

Ferreira, A., Cruz-Correia, R., Antunes, L., Farinha, P., Oliveira-Palhares, E., Chadwick, D. W., and Costa-Pereira, A. (2006). How to break access control in a controlled manner. In *Computer-Based Medical Systems, 2006. CBMS 2006. 19th IEEE International Symposium on*, pages 847–854. IEEE.

Joint, N. (2004). Break-glass: An approach to granting emergency access to healthcare systems. *COCIR/JIRA Security And Privacy Committee (SPC)*.

Kiayias, A., Oksuz, O., Russell, A., Tang, Q., and Wang, B. (2016). Efficient encrypted keyword search for multi-user data sharing. In *European Symposium on Research in Computer Security*, volume NA, pages 173–195. Springer.

Li, T., Liu, Z., Jia, C., Fu, Z., and Li, J. (2018). Key-aggregate searchable encryption under multi-owner setting for group data sharing in the cloud. *International Journal of Web and Grid Services*, 14(1):21–43.

Li, T., Liu, Z., Li, P., Jia, C., Jiang, Z. L., and Li, J. (2016). Verifiable searchable encryption with aggregate keys for data sharing in outsourcing storage. In *Information Security and Privacy*, pages 153–169. Springer International Publishing.

Mahalle, R. V. and Pawade, P. P. A review of secure data sharing in cloud using key aggregate cryptosystem and decoy technology. *International Journal of Science and Research (IJSR)*, 3.

Marinovic, S., Craven, R., Ma, J., and Dulay, N. (2011). Rumpole: a flexible break-glass access control model. In *Proceedings of the 16th ACM symposium on Access control models and technologies*, pages 73–82. ACM.

Padhya, M. and Jinwala, D. (2018). Mulkase - a novel approach for key aggregate searchable encryption for multi-owner data. *Frontiers of Information Technology Electronic Engineering*, -1(-1).

Patranabis, S., Shrivastava, Y., and Mukhopadhyay, D. (2015). Dynamic key-aggregate cryptosystem on elliptic curves for online data sharing. In *International Conference in Cryptology in India*, pages 25–44. Springer.

Sahai, A., Waters, B., et al. (2005). Fuzzy identity-based encryption. In *Eurocrypt*, volume 3494, pages 457–473. Springer.

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

Schaad, A. and Moffett, J. D. (2002). Delegation of obligations. In *Policies for Distributed Systems and Networks, 2002. Proceedings. Third International Workshop on*, pages 25–35. IEEE.

Zhou, R., Zhang, X., Du, X., Wang, X., Yang, G., and Guizani, M. (2018). File-centric multi-key aggregate keyword searchable encryption for industrial internet of things. *IEEE Transactions on Industrial Informatics*.