Data Science and Engineering (2020) 5:391–418
https://doi.org/10.1007/s41019-020-00136-y

R-OO-KASE: Revocable Online/Offline Key Aggregate Searchable Encryption

Mukti Padhya[1] · **Devesh C. Jinwala**[1]

Received: 31 October 2019 / Revised: 30 January 2020 / Accepted: 16 July 2020 / Published online: 6 August 2020

Abstract
The existing Key Aggregate Searchable Encryption (KASE) schemes allow searches on the encrypted dataset using a single query trapdoor, with a feature to delegate the search rights of multiple files using a constant size key. However, the operations required to generate the ciphertext and decrypt it in these schemes incur higher computational costs, due to the computationally expensive pairing operations in encryption/decryption. This makes the use of such schemes in resource-constrained devices, such as Radio Frequency Identification Devices, Wireless Sensor Network nodes, Internet of Things nodes, infeasible. Motivated with the goal to reduce the computational cost, in this paper, we propose a Revocable Online/Offline KASE (R-OO-KASE) scheme, based on the idea of splitting the encryption/decryption operations into two distinct phases: online and offline. The offline phase computes the majority of costly operations when the device is on an electrical power source. The online phase generates final output with the minimal computational cost when the message (or ciphertext) and keywords become known. In addition, the proposed scheme R-OO-KASE also offers multi-keyword search capability and allows the data owners to revoke the delegated rights at any point in time, the two features are not supported in the existing schemes. The security analysis and empirical evaluations show that the proposed scheme is efficient to use in resource-constrained devices and provably secure as compared to the existing KASE schemes.

Keywords Searchable encryption · Data sharing · Data retrieval · Cloud server · Multi-keyword search · Online/offline encryption · Revocation

Mathematics Subject Classification 94A60 · 68P25

1 Introduction

Cloud computing services are often resorted to, with an aim to reduce the overhead of data management and data processing at the user side. However, when the data are outsourced and stored on a remote cloud, it is often desired to encrypt the same in order to protect the data from unauthorized access. One of the issues associated with encryption is that the accessibility and usability of encrypted data are definitely lowered since the latter would require the decryption of data before being put to use. There are two distinct threads of research pursued in the literature with respect to addressing this issue of improving the usability of the encrypted data viz. (1) carrying out arbitrary computations on the encrypted data using homomorphic encryption [8] or (2) devising operation by which the encrypted data can, at least, be searched for the desired keyword value, to be present or not. Our focus in this paper is on the latter, i.e., on *Searchable Encryption* (SE) [28]. The SE schemes, typically consist of three distinct entities with distinct roles as shown in Fig. 1. An SE cryptosystem allows a server to search given query keyword(s) on encrypted data on behalf of the user without learning information about the plaintext data.

However, secure sharing of the search rights for the selected dataset is not an easy task for the data owner, since it is often desired to encrypt different document sets using different encryption keys, for confidentiality and privacy considerations. Therefore, the sharing of search rights of the dataset using the existing SE methods [2, 5, 9, 28–30] require efficient management and distribution of more than one key. Specifically, to delegate the search and access rights

✉ Mukti Padhya
mukti.padhya@yahoo.in

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

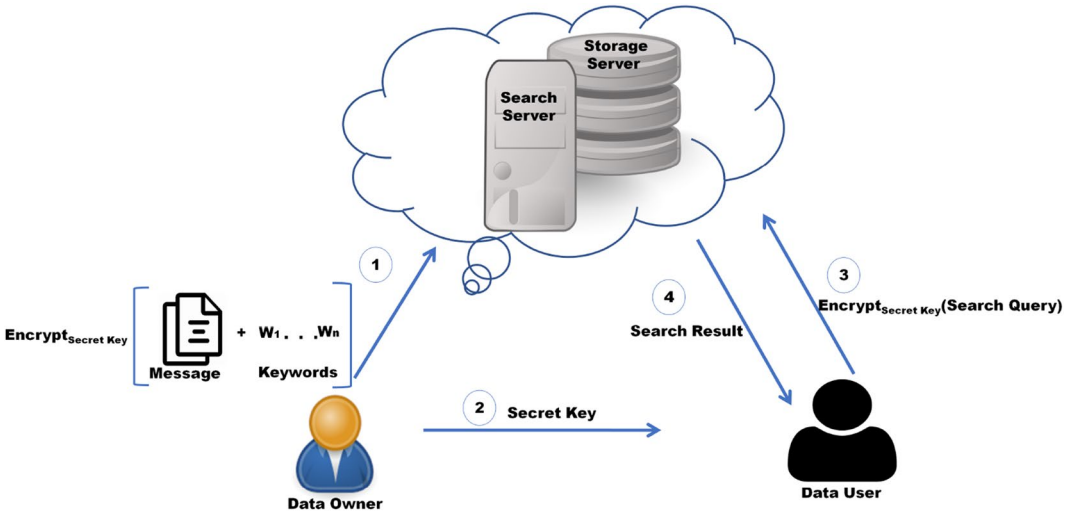


Fig. 1 The system model of SE. Steps: (1) The data owner uploads a ciphertext (i.e., encrypted message + related keywords list) onto the storage server. (2) The data owner shares a searchable secret key with the user. (3) The data user constructs a searchable trapdoor using a secret key. The data user sends a trapdoor to the server. (4) The search server applies a trapdoor on the ciphertext. The server returns the search results to the query requesting user

over set S of files encrypted under different keys, a data owner is required to share $|S|$ number of keys. This solution is not practically deployable for two major reasons. Firstly, the number of secret keys grows linearly with the number of files, and it incurs storage as well as communication overhead of $O(|S|)$. Therefore, a data owner is required to store, manage and distribute a large number of keys, proportional to the number of shared files. In addition, an authorized user is required to generate and submit multiple trapdoors in order to search through all of the shared data. Secondly, if a key assigned to a user needs to be revoked later, then the data owner is required to re-encrypt the corresponding subset of data and distribute a new set of keys to the authorized users. This makes the scheme inefficient and difficult to scale.

The most efficient proposition pertaining to our problem statement, to the best of our knowledge, is made in [4]. Key-Aggregate Searchable Encryption (KASE) [4] combines the searchable encryption scheme with group data sharing. KASE [4] is proposed to reduce the number of keys required to be shared for the delegation of search rights over the set of data. KASE inherits the property of Key Aggregate Encryption (KAE) [3], i.e., to delegate search rights on dataset S, the data owner requires 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 server for searching a keyword over $|S|$ number of shared files. Despite the advantage of the secure delegation of search rights using a key of constant size, the existing KASE schemes [4, 13, 18–24, 31, 32, 34, 35] still face issues when deploying them into real-time scenarios, as we discuss further, here.

1.1 Motivation

In a practical data sharing system based on cloud storage, the user can upload or retrieve the data from a different possible devices such as Personal Computer (PC), mobile device, sensor nodes (especially in the Internet of Things (IoT) applications) [4]. The constant size of the aggregate key makes the KASE scheme more suitable for devices having limited storage resources, such as smart cards, IoT sensors. However, the existing KASE schemes [4, 13, 18–21, 31] use two pairing operations to generate a keyword ciphertext for each keyword to be attached with the document. Similarly, for decryption of ciphertext pairing operations are required. The authors of [25] show that for different security levels (i.e., for 80 bits, 112 bits and 128 bits) and prime order groups, the pairing operation (on random group elements) requires significantly more time and power resource as compared to the group exponentiation (of a random group element with a random exponent) and multiplication operation. Therefore, the limited power resource of sensor nodes in IoT deployment does not support the expensive computational cost of the existing KASE schemes [4, 13, 18–21, 31]. The expensive computational cost of encryption and decryption phases make the existing KASE [4, 13, 18–21, 31] schemes infeasible to use in power-constrained devices. Therefore, the KASE schemes [22, 34, 35] propose different solutions to generate keyword ciphertext without using expensive pairing operations and make the schemes practical for the

resource-limited environment. However, if the data owner wants to attach a set of keywords $\overrightarrow{KW} = \{KW_1, \ldots, KW_q\}$ with a document doc_l, then the encryption algorithm in the existing KASE [4, 13, 18–22, 24, 31, 32, 34, 35] schemes generate $|\overrightarrow{KW}|$ different keyword ciphertexts. Formally, to attach a set of keywords $\overrightarrow{KW}$ with a document doc_l, the storage overhead of resultant ciphertext C_l is $O(|\overrightarrow{KW}|)$, with a communication cost of $O(|\overrightarrow{KW}|)$, as a user has to store $|\overrightarrow{KW}|$ number of keyword ciphertexts to the cloud server [36]. Therefore, if the data owner frequently uploads dataset from the power-constrained device, the required computational, communication overhead and its impact on power-constrained device negate the advantages of KASE [4, 13, 18–22, 24, 31, 32, 34, 35]. Hence, devising the KASE scheme to work on resource-limited devices, with the battery as the only source of power is non-trivial.

In addition, the existing KASE schemes [4, 13, 18–22, 24, 31, 32, 34] do not support revocation of delegated rights. The revocation refers to the process of taking away the delegated privileges. As the access of users 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. If the receiver of delegated rights leaves the system, or if the data is used differently than the data owner agreed, the delegated rights need to be revoked by the data owner. Therefore, Zhou et al. [35] propose the solution for revocable KASE. However, the KASE scheme proposed in [35] requires the data owner to generate and distribute the new set of keys to the non-revoked users in each time period. Specifically, the data owner maintains a list of authorized users. The authorized users registered in the data owner's list can receive a new pair of authorized keys from the data owner. Therefore, it consumes more overhead at the data owner side to generate as well as distribute keys periodically to each authorized user of the system. This paper also aims to design revocable KASE scheme which will be efficient for resource-limited environment.

There are different revocation schemes proposed in the literature viz. strong and weak revocations, cascading and non-cascading revocations, rule-based, role-based and user-based revocations, direct and indirect revocations (for further details see [33]). However, in the previous revocation schemes, if the shared documents are modified or if the delegated rights for the shared documents needs to be revoked, the data owner is required to outsource the list of revoked users (or the list of revocation rule or revocation policy) and the corresponding revoked document id (docID) set. However, the idea of uploading the revocation list in the plain form on the cloud server can trigger security threats (*one can change docID or revoked user's identity in the list*). We cannot ignore the user's privacy and security of the revocation list in the public cloud storage system.

Furthermore, the existing KASE schemes only support a single keyword search. To perform a conjunctive keyword search using the existing KASE schemes [4, 13, 18–22, 31, 32, 34, 35], the user requires to submit different trapdoors for each individual keyword to the server. The server performs a search for each of the keywords separately and returns the intersection of all the results. This approach leaks information about which documents contain each individual keyword and may allow the server to learn information about the documents and it's related keywords. Specifically, for searching a set of keywords $\overrightarrow{Q} = \{Q_1, \ldots, Q_p\}$ over shared dataset, the existing KASE schemes [4, 13, 18–22, 31, 32, 34, 35] require $|\overrightarrow{Q}| = p$ number of trapdoors. This results in a communication cost of $O(|\overrightarrow{Q}|)$ and a computational cost of $O(|\overrightarrow{Q}|)$ at both the cloud server and user (query generator) sides. Therefore, to improve the system usability, query expressiveness and system performance (in terms of accuracy, communication and computational cost), KASE must support multiple keywords search using a single query trapdoor.

The issues in the existing KASE schemes makes it non-trivial to design a KASE scheme that meets the following requirements viz. : *revocation of delegated rights, multi-keyword search using a single trapdoor and minimal online computational overhead with minimal energy usage at both the data owner and the users' sides simultaneously*.

1.2 Our Contributions

The contributions of this paper are summarized as follows:

- *Energy Efficient KASE* Utilizing the notion of KASE, we design a KASE scheme that is suitable for resource-constrained devices, as we split costly operations of encryption and decryption into two phases: online and offline. In the offline phase, the user performs expensive pairing and exponentiation operations required in the encryption/decryption. In the online phase, i.e., when the device is moving on (not connected to power source), the user can generate the final output with the minimal computational cost. Additionally, instead of using expensive pairing operation, we use exponentiation operation to generate the keyword ciphertext.
- *Multi-Keyword Search* We enhance the existing KASE schemes and improve their query expressiveness by supporting multi-keyword searches over the shared dataset using a trapdoor of constant size.
- *Revocation* With the proposed scheme, we offer the revocation of delegated rights, without affecting other users in the system. The user is not allowed to search the encrypted data by the old trapdoor computed from the old secret key if his search privileges are revoked. The proposed scheme supports fine-grained revocation of the

delegated rights on document level, instead of coarse-grained all-or-nothing access.

- *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 existing KASE schemes [4, 13, 18–21, 31, 32, 34, 35] require $|S|$ number of trapdoors to search over shared dataset S, as the existing KASE schemes generate Tr_l for each $l \in S$ using the given trapdoor Tr. Here, Tr_l is the adjusted trapdoor that is used to search over document doc_l in the KASE schemes [4, 13, 18–21, 31, 32, 34, 35]. (Detailed comparison of performance analysis is done in Sect. 7)
- *Efficiency* In the proposed R-OO-KASE scheme, the size of the aggregate key is constant, i.e., it consists of single-group elements irrespective of the number of shared documents. Similarly, the query trapdoor and keyword ciphertext are also of constant size irrespective of the number of keywords.

To the best of our knowledge, ours is the first scheme which offers all the above-mentioned features all together.

1.3 Outline of the Paper

The rest of the paper is organized as follows: First, we review some background knowledge and related work in Sect. 2. The preliminaries are given in Sect. 3. The problem statement and overview of the proposed scheme, framework, the system model and the security model are defined in Sect. 4. We give a concrete construction of the proposed R-OO-KASE scheme in Sect. 5. The security analysis of the R-OO-KASE scheme is discussed in Sect. 6. The theoretical and empirical analysis is made in Sects. 7 and 8, respectively. Conclusion and future extensions are provided in Sect. 9. References are at the end.

2 Related Work

Key Aggregate Encryption (KAE) [3], derives its roots from the seminal work on broadcast encryption by Boneh et.al. [1]. KAE may essentially be considered as a dual notion of broadcast encryption [1]. In broadcast encryption, a single ciphertext is broadcast among multiple users, each of whom may decrypt the same using their own individual private keys. In KAE, a single aggregate key is distributed among multiple users in order to delegate access rights on the dataset. For broadcast encryption, the focus is on having shorter ciphertexts and low overhead individual decryption keys, while in KAE, the focus is on having short ciphertexts and low overhead aggregate keys.

In KAE, ciphertexts are associated with an index i, given by data owner at the time of encryption. Therefore, if data owner wants to delegate access rights of set S (set of ciphertexts' indices) of ciphertexts, then he can generate a single key k_{agg} of constant size by aggregating secret keys of all the ciphertexts in the set S. The user can decrypt any ciphertext using a single aggregate key if the index of ciphertext is within set S.

Further, to retrieve selected data from outsourced dataset and simultaneously delegate the search rights of selected dataset using a single aggregate key, the first solution for KASE is proposed in [4]. In a KASE scheme, the data owner requires sharing a single aggregate key to a user for delegating search rights over a set of documents and the user requires submitting a single trapdoor for searching over the shared dataset.

The CLW16 scheme proposed in [4] is the first solution for KASE. However, the formal security proof against keyword guessing attack (to prove trapdoor privacy) and chosen keyword attack (to prove keyword ciphertext privacy) are not given for the CLW16 scheme. Moreover, the CLW16 scheme is insecure against cross-pairing attack and do not provide trapdoor privacy, as discussed in [14]. Zhou et al. in [34] show the attack on CLW16 scheme, in which the attacker can guess the authorized user's key with the help of insider adversary. Further, the CLW16 scheme is not scalable because the system parameters in the Cui's scheme [4] are strictly bounded by the value of n (number of documents belong to the data owner). Specifically, if the data owner generates ciphertexts beyond predefined limit n, he must request additional key pairs in such case. If the data owner wants to delegate the search rights of ciphertexts $\{C_1, C_n, C_{n+1}\}$, then he is required to share two different aggregate keys, i.e., k_{agg1} for $\{C_1, C_n\}$ and k_{agg2} for C_{n+1}. Therefore, the predefined bound on the maximum number of possible ciphertext classes at the time of system setup makes the CLW16 scheme impractical for real-time use. In the CLW16 scheme, when a user submits aggregate trapdoor Tr to the cloud server to carry out searching over a set S of files, the server first requires to run Adjust algorithm. The adjust algorithm generates trapdoor Tr_l for each index $l \in S$ using submitted trapdoor Tr before searching. Here, Tr_l is the actual trapdoor that is used to search over document doc_l. This trapdoor transformation adds additional computational cost at server side before searching over dataset S. The CLW16 scheme does not support searching over multi-owner data using a single key of constant size. Formally, in a multi-owner setting, a user can have multiple aggregate keys $\{k_{agg_1}, \ldots, k_{agg_\kappa}\}$ received from different data owners. Therefore, to search across multi-owner dataset, the CLW16 scheme requires user to submit $\{Tr_1, \ldots, Tr_\kappa\}$ different trapdoors to the cloud server. This results in a system with $O(\kappa)$

communication overhead on the user side and $O(\kappa)$ storage as well as computational overhead on the server side.

The TZPCZJ16 scheme proposed in [19] provides the solution to search over multi-owners' data using a single trapdoor and also allows verification of search result using an aggregate key. However, TZPCZJ16 scheme requires the auxiliary values having size in linear with the number of data owners while searching over multi-owner data. Further, the formal security proof against keyword guessing attack and chosen keyword attack are not given for the TZPCZJ16 scheme. The TZPCZJ16 scheme follows the same construction as the CLW16 scheme. Therefore, the cross-pairing attack is possible on the TZPCZJ16 scheme as well.

The TZCZJ18 scheme proposed in [18] also provides the solution to search over multi-owners' data using a single trapdoor. However, TZCZJ18 scheme requires the auxiliary values having size in linear with the number of data owners while searching over multi-owner data. Further, the TZCZJ18 scheme is not scalable and requires trapdoor transformation at server side before searching on requested dataset. Additionally, if the data owner wants to attach a set of keywords $\overrightarrow{KW} = \{KW_1, \ldots KW_q\}$ with a document doc_l, then the computational cost of the encryption algorithm in the existing KASE [4, 18, 19] schemes increase linearly with the number of keywords attached with the ciphertext. Formally, to attach a set of keywords $\overrightarrow{KW}$ with a document doc_l, the storage overhead of resultant ciphertext C_l is $O(|\overrightarrow{KW}|)$, with a communication cost of $O(|\overrightarrow{KW}|)$. The user has to store $|\overrightarrow{KW}|$ number of keyword ciphertexts to the cloud server [36]. The expensive pairing operations used to generate a keyword ciphertext in the existing KASE [4, 18, 19] schemes drain more energy and makes the schemes infeasible to use in power-constrained devices.

The ZZDWYG18 scheme proposed in [34] consider Industrial IoT (IIoT) application and propose KASE scheme in the file-centric framework for the IIoT application. The sensors in IIoT deployment usually have extremely limited hardware resource and do not support computation cost of pairing operations. Therefore, Zhou et al. [34] propose KASE scheme without using pairing computation operations in the encryption phase. However, the KASE scheme proposed in [34] 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.

The ZZWYL18 scheme proposed in [35] provides solution for fine-grained right revocation at document level in IoT environment. The ZZWYL18 scheme prove the keyword confidentiality, query privacy and forward secrecy. The ZZWYL18 scheme generates keyword ciphertext without using expensive pairing operations and makes the scheme practical for resource limited environment.

The PJ18 scheme proposed in [22] supports search over multi-owners' data using a single trapdoor of constant size. The PJ18 scheme also overcomes security issues of the existing KASE schemes [4, 18, 19] and prove the security against cross-pairing attack. Furthermore, the PJ18 scheme is scalable in such a way that the value n is kept variable and an aggregate key is independent of the value of n. The PJ18 scheme allows searching over the shared dataset S using a single trapdoor Tr that a query requester submits to the cloud server. The scheme does not use Adjust algorithm for trapdoor transformation at the server side. The PJ18 scheme generates keyword ciphertext without using expensive pairing operations and makes the scheme practical for resource limited environment. The PJ18 scheme also discuss the scenario of federated cloud and shows how to use their scheme for delegation of search rights if data are stored on the federated cloud.

The ZY18 scheme proposed in [21] provides solution for verification of search result using an aggregate key and user authentication. The cloud server can verify the legality of data user by authenticating whether data user's identity is contained in the authorized users' identity set. However, the solution proposed in [21] is not secure against impersonation attack. The ZY18 scheme does not give any formal proof to prove the security of the scheme. The ZY18 scheme is not scalable as the system parameters are bounded by the value of n. Further, the usage of Adjust algorithm in the ZY18 scheme increases server search time because of trapdoor transformation process. The ZY18 scheme uses two pairing operations to generate a keyword ciphertext for each keyword to be attached with the document, which makes the scheme impractical to be used in resource constraint environment.

The ZTPCJ18 scheme proposed in [20] provides verification of search result using an aggregate key and also allows search over multi-owner data using a single trapdoor of constant size. However, the ZTPCJ18 is suffering from same limitation as TZPCZJ16 scheme. The ZTPCJ18 scheme requires the auxiliary values having size in linear with the number of data owners while searching over multi-owner data. Further, the formal security proof against keyword guessing attack and chosen keyword attack are not given for the ZTPCJ18 scheme.

Yao et al. propose the first lattice-based KASE scheme in [32]. The lattice-based KASE [32] scheme provides security against quantum computing attacks and potential efficiency. Padhya et al. [23] propose a revocable KASE scheme with Break-The-Glass access control. The KASE [23] scheme provides a mechanism that can handle emergency situations where no authorized user exists to perform (or to delegate) a time-critical task. The KASE scheme proposed in [24] supports the conjunctive range and sort query on the encrypted dataset and enhances the query expressiveness of

the existing KASE [4, 13, 18–22, 32, 34, 35] schemes. As compared to the previous solutions, the KASE [24] scheme supports multi-dimensional, multi-keyword searches on the encrypted dataset using a single trapdoor. Wang et al. propose the verifiable KASE scheme in [31].

To the best of our knowledge, none of the existing KASE schemes is practically applicable for resource-constrained environment.

3 Preliminaries

In this section, we review some basic assumptions and cryptology concepts that we use throughout the paper.

3.1 Bilinear Map

A pairing is a bilinear map defined over elliptic curve subgroups. Let G_1 and G_2 be two multiplicative cyclic elliptic curve subgroups of the same prime order p. Let G_T be a multiplicative group, also of order p with identity element 1. A mapping $e : G_1 \times G_2 \rightarrow G_T$ is said to be a bilinear map if it satisfies the following properties:

1. *Bilinearity* for all $P_1 \in G_1, Q_1 \in G_2, u, v \in Z_p^*$, we have $e(P_1^u, Q_1^v) = e(P_1, Q_1)^{uv}$.
2. *Non-degeneracy* if P and Q be the generators for G_1 and G_2 respectively, then $e(P, Q) \neq 1$.
3. *Computability* there is an efficient algorithm to compute $e(P_1, Q_1)$ for any $P_1 \in G_1, Q_1 \in G_2$.

3.2 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 $Adv_{DDH}(A) := |Pr[A(g, g^a, g^b, g^c, g^{abc}) = 0] - Pr[A(g, g^a, g^b, g^c, R) = 0]| \geq \epsilon(\kappa)$. 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.

3.3 Notations

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

Table 1 Notations used in the proposed R-OO-KASE 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	lth document
$PubK$	Public parameters
pk	Public key
msk	Master secret key
l	Index of document or file
S	Subset $S \subseteq \{1, \ldots, n\}$ contains the indices of documents
KS	Keyword space that involves m different keywords in the system
w_i	ith keyword
$w_{i,j}$	jth 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
IC_l	Intermediate ciphertext of lth document
$C_{l_{\overrightarrow{KW}}}$	Keyword ciphertext of lth document
C_{l_M}	Data ciphertext
$IC_{l_{i,j}}$	Intermediate keyword ciphertext for $w_{i,j}$ keyword value
δ_l	Public information related to ciphertext of lth document i.e., parameters C_1 and C_2
k_{agg}	Aggregate key
Tr	Trapdoor used to search query keyword(s) within set $\overrightarrow{Q}$
R	Search result
RL_l	Revocation list of lth document
$\mathcal{U}_l$	The set of revoked users' identities for lth document
U_{QR}	Identity of query requester
Π	Multiplicative notation

4 System Architecture and Security Model

In this section, we begin by discussing the overview of the proposed scheme. Then, we define the system model of the proposed R-OO-KASE scheme. We formally define the framework of the proposed scheme. Finally, we outline the game-based framework for formally proving the security of the proposed scheme.

4.1 Problem Statement and Overview

For reducing the computation burden at both data owners and users' side, several techniques for outsourcing computation of encryption and decryption operations are proposed in [10, 15–17, 37]. However, for outsourcing computations, the required communication cost is high, especially

for resource-limited devices. Therefore, this paper aims to design a secure and efficient KASE scheme that allows the user to perform encryption and decryption operations on power-constrained devices.

In this paper, we propose a KASE scheme to reduce the computational overhead at both the data owners and users' side, by splitting the encryption and decryption operations into two phases: online and offline, as shown in Fig. 2. The proposed scheme allows ciphertext preparation work done offline. Formally, in the proposed scheme, the resource-limited devices perform the expensive pairing and exponentiation operations in offline mode (when devices are on electrical power source), whereas performing the multiplication operations in online mode (when devices are deployed at real-time applications). In the proposed system, the offline phase will generate the intermediate ciphertext for each keyword in the system using the only public key. When the encryption algorithm later needs to encrypt a set $\overrightarrow{KW}$ of keywords, it selects keyword ciphertext for each $KW_i \in \overrightarrow{KW}$ from the precomputed intermediate ciphertext and aggregates it to generate a keyword ciphertext $C_{\overrightarrow{KW}}$. The keyword ciphertext in the proposed scheme is having a constant size. The online encryption requires one multiplication in Z_p for each keyword to be attached with the ciphertext instead of costly pairing operations. We remark that the work done in the offline phase is roughly equivalent to the work of the regular encryption algorithm in the existing KASE schemes [4, 18–22, 34, 35]. Moreover, in the proposed KASE scheme, the communication cost of ciphertext is constant and independent of number of keywords attached with the ciphertext. The constant communication cost of the ciphertext makes the proposed R-OO-KASE scheme practically applicable in the resource-constrained environment, with the battery as the only source of power. In a similar way, we reduce the computation overhead at the user side by precomputing the

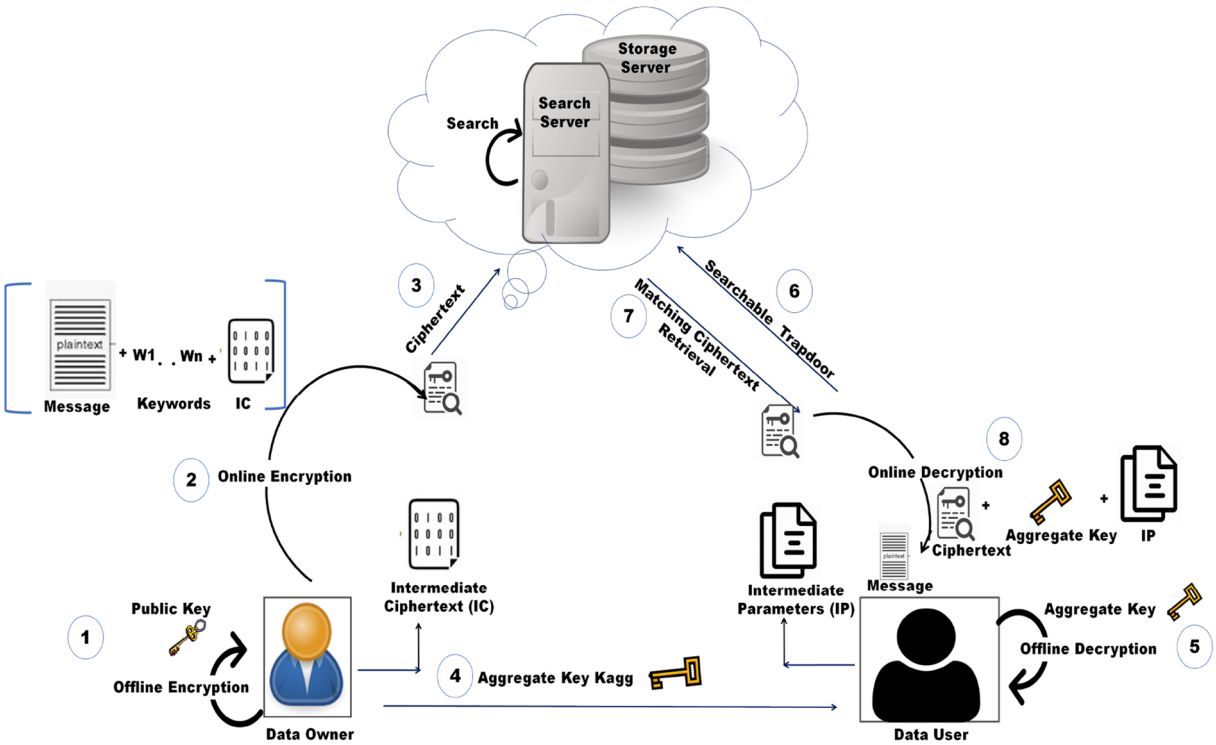


Fig. 2 The system model of online/offline KASE. Steps: (1) In the offline encryption phase, the data owner computes intermediate ciphertexts using the only public key. Most of the high computational cost operations (i.e., pairing, exponentiation) without knowing the message and keywords are done at this phase. (2) At the time, the data owner receives the message and it's related keywords set for the encryption, the data owner selects one intermediate ciphertext and runs Online Encrypt() algorithm. The data owner generates a final ciphertext C for a given message and keywords set, with minimal computation overhead. (3) The data owner sends the ciphertexts onto the storage server. (4) The data owner delegates search and access rights over the selected dataset by generating and sharing an aggregate key with other user(s). (5) Using an aggregate key, the data user can precompute the parameters that are required to decrypt the ciphertexts within a range of the shared dataset. (6) To search over the shared dataset, the data user generates a query trapdoor using an aggregate key and a set of query keywords. The data user sends the query trapdoor to the search server. (7) The server performs a search over stored encrypted data using a given query trapdoor. The server returns search results to the query requester. The user retrieves matching documents that are satisfying given the search query from the storage server. (8) In the online phase, the data user decrypts the ciphertexts using an aggregate key and precomputed parameters (which are generated in the offline phase)

parameters required for decryption in the offline phase and keeping the online computation task very less. In the proposed R-OO-KASE scheme, the online phase of the decryption requires three pairing operations to be done.

Another potential advantage of splitting work this way is that in some applications, the online and offline work can be performed on different devices. Hence, one might perform the offline tasks for several encryptions on a high-end server and store these intermediate ciphertexts on a sensor device such that the resource-constrained device is not required to perform full encryption.

The proposed scheme also allows the data owner to revoke delegated rights of selected users within set $\mathcal{U}_l$ over lth document, without affecting the users' rights over the rest of the shared documents. The data owner generates the revocation list RL_l for lth document using his master-secret key msk and set of revoked users' identities $\mathcal{U}_l$. The cloud server checks the user's authorization each time when he receives search and data access requests for any stored document. If the identity of the query requester U_{QR} matches with the revocation list RL_l, then the failure state is returned to the user, otherwise, the cloud server performs the search process over lth document. Furthermore, the proposed scheme supports fine-grained revocation of the delegated rights on document level, instead of coarse-grained all-or-nothing access. Additionally, the data owner generates a revocation list using the master-secret key, and there is a very negligible probability that an adversary can get the master-secret key of the data-owner. Thus, any malicious user without having the master-secret key cannot modify or generate a new revocation list. The revocation in the proposed scheme does not affect the non-revoked users, as they do not require to update their corresponding delegated keys, which greatly reduces the expensive cost of key updates and the overhead of key delegate authority. The proposed scheme also preserves user privacy along with the revocation of delegated rights.

4.2 Assumptions

The keyword space contains m different keywords, i.e., $KS = \{w_1, w_2, \ldots, w_m\}$. Each keyword w_i contains $\mathcal{N}_i$ possible values and $w_{i,j}$ represents the jth value of keyword w_i. However, the number of keywords are not bounded, one can add new keywords after system initialization. Let we assume that there are total X users in the system and $U = \{U_1, \ldots, U_X\}$ 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 lth document, where $\mathcal{U}_{l_i} \in \mathcal{U}_l$ represents an identity of user whose delegated rights needs to be revoked over lth document. The revocation list $\mathcal{U}_l$ may be empty, which means there is no user whose delegated rights needs to be revoked over lth document.

4.3 System Model

The proposed R-OO-KASE scheme mainly consists of four parties: (i) the trusted authority, (ii) the cloud server, (iii) the data owner and (iv) the users (Fig. 3).

The interactions among involved parties in the proposed R-OO-KASE scheme are as follows:

(1) The data owner who wants to outsource his data $\{doc_1, \ldots, doc_n\}$ on the cloud server and share multiple files with others through the cloud server, first performs Setup() to initialize the system parameters. At the time of system initialization, the trusted authority assigns every cloud user a unique identity $U_{id} \in U$. The data owner executes KeyGen() to generate a public-master secret key pair (pk, msk). The public key pk is used for encrypting keywords and message. The master-secret key msk for delegation of search rights is kept private by the owner.

(2) In the offline phase (when the resource-limited device is connected to the power source), the data owner executes Offline_Encrypt() algorithm using a public key and generates intermediate ciphertexts $\{IC_l\}_{l\in[1,n]}$, without knowing the message and keyword(s) to be encrypted. The intermediate ciphertext $\{IC_l\}_{l\in[1,n]}$ contains keyword ciphertexts $\{IC_{l_{i,j}}\}_{i\in[1,m],j\in[1,\mathcal{N}_i]}$ for each keyword in the system $\{w_{i,j}\}_{i\in[1,m],j\in[1,\mathcal{N}_i]} \in KS$ along with the auxiliary values $\{IC_{l_1}, IC_{l_2}, IC_{l_3}\}$ that are required to generate final ciphertext C_l in the online encryption phase.

(3) To encrypt the lth document, i.e., doc_l, the data owner chooses lth intermediate ciphertext IC_l and runs Online_Encrypt() algorithm for generating final ciphertext C_l. To encrypt a set $\overrightarrow{KW}$ of keywords, the Online_Encrypt() algorithm selects $\{IC_{l_{i,j}}\}_{w_{i,j}\in\overrightarrow{KW}}$ and aggregates it to make a single element of keyword ciphertext $C_{l_{\overrightarrow{KW}}}$. Specifically, the online encryption requires one multiplication in Z_p for each keyword to be attached with the ciphertext. Furthermore, using auxiliary values of intermediate ciphertext $\{IC_{l_1}, IC_{l_2}, IC_{l_3}\}$, the Online_Encrypt() algorithm generates public information δ_l and data ciphertext C_{l_M}. The data owner outsources resultant ciphertext $C_l = (\delta_l, C_{l_{\overrightarrow{KW}}}, C_{l_M})$ to the cloud server.

(4) The data owner is capable to grant search and access rights of the selected set S of documents to other user(s) by sharing an aggregate key of constant size. The aggregate key k_{agg} is the single secret key using which an authorized user can retrieve and access all the shared documents. As k_{agg} is having a constant size, it is easy for the data owner to share an aggregate

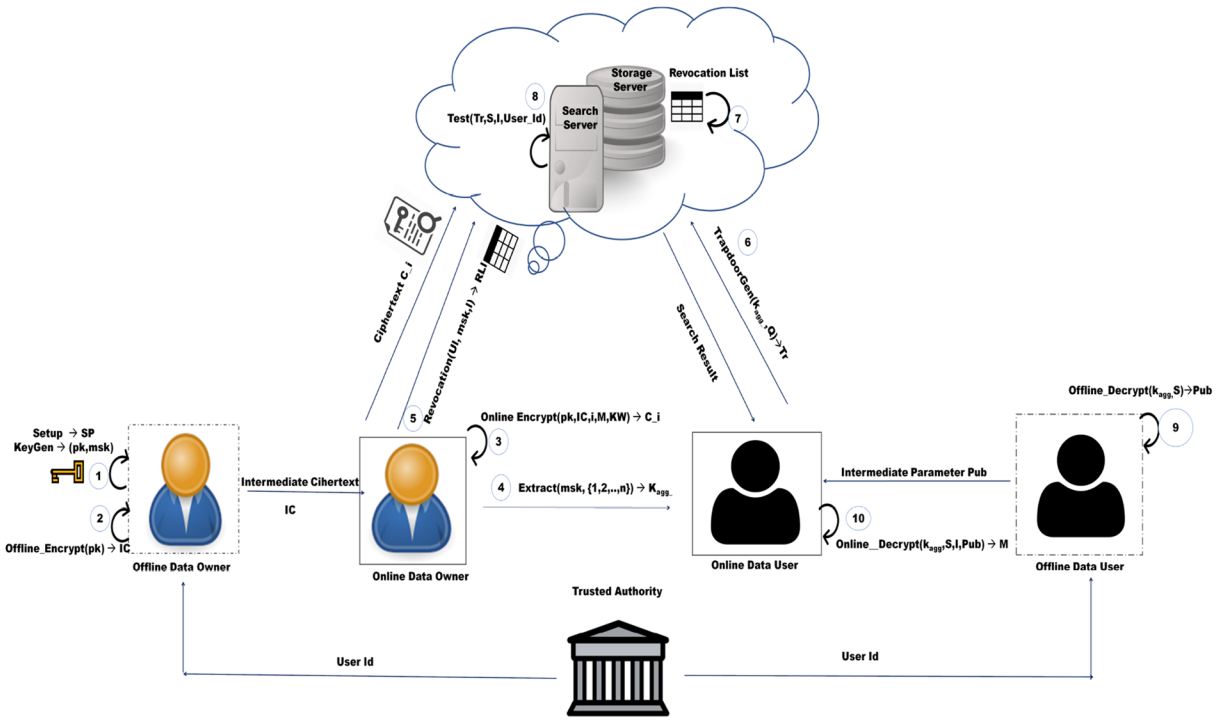


Fig. 3 System model of the proposed R-OO-KASE scheme

key with other user through a secure communication channel and with small communication costs.

(5) The data owner can revoke delegated rights of selected users within set $\mathcal{U}_l \subset U$ over lth document by generating encrypted revocation list RL_l. The data owner uploads the revocation list on the cloud server. If and only if the privileges of the query requester have not been revoked, he can search and access the ciphertext C_l from the cloud server to obtain plaintext.

(6) An authorized user can search over shared dataset S within scope of an aggregate key k_{agg} using a single trapdoor Tr. The user can generate a query trapdoor Tr using the shared aggregate key k_{agg} and set of query keyword(s) $\overrightarrow{Q}$. A query trapdoor Tr represents search query within set $\overrightarrow{Q}$ to the server. The user submits the trapdoor Tr to the cloud server.

(7) The cloud server provides massive storage and computation resources to the users. At the time of receiving data access or search request for any document l stored on the cloud server, the server checks the user's authorization using a revocation list RL_l. If the identity of the query requester U_{QR} matches with the revocation list RL_l, then the failure state is returned to the user, otherwise, the cloud server performs the search process over lth document. With the failure state, the user will not be able to proceed further and search (or access) the document l. If a user $\mathcal{U}_{l_i} \in \mathcal{U}_l$ was previously authorized to access documents within set S using an aggregate key k_{agg}, now, he cannot further access document $l \in S$, due to the published revocation list. However, the user can search over other shared documents within set $S - \{l\}$.

(8) The cloud server performs the keyword search on behalf of the user, using the submitted trapdoor Tr and returns the search results to the user. The search result contains a true or false value for each document $l \in S$, indicating whether a document contains keywords within the query set $\overrightarrow{Q}$ or not. Then, the query requester only requires to download matching documents that are satisfying search query, instead of downloading all the shared documents.

(9) In the offline mode, the resource-constrained device used by the data user executes Offline_Decrypt() and generates the parameters $pub = (\{pub_{1\_l}, pub_{2\_l}\}_{l \in S}, pub_3, pub_4)$, that are used to decrypt the ciphertexts within set S.

(10) An authorized user runs Online_Decrypt() algorithm to decrypt the ciphertext, using an aggregate key k_{agg} and parameters pub generated by Offline_Decrypt().

4.4 The R-OO-KASE Framework

The proposed R-OO-KASE scheme is an ensemble of randomized polynomial-time algorithms, as discussed in this section. The data owner first sets up an account on the cloud server and establishes the public system parameters via

Setup(). At the time of registering in the system, the Trusted Authority (TA) assigns a unique identity 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/master-secret key pair via **KeyGen()**. Messages can be encrypted via **Offline_Encrypt()** and **Online_Encrypt()** algorithms by anyone who has the public key of the data owner. To delegate the search and access rights to a specific subset of data, the data owner uses the master-secret key and generates a constant size aggregate key via **Extract()**. The generated key can be passed to the delegatees securely (via secure e-mails or secure devices). The data owner can take away the delegated privileges via the **Revocation()** algorithm. Finally, any user with an aggregate key can search over the shared dataset by generating a query trapdoor via **Trapdoor-Gen()**. On receiving a query trapdoor from the user, the cloud server runs the **Test()** algorithm to retrieve matching documents and sends it further to the query requester. The user can decrypt the ciphertext and access the plaintext via **Offline_Decrypt()** and **Online_Decrypt()** algorithms, provided the ciphertext is within the scope of an aggregate key.

We now describe each of the algorithms involved in R-OO-KASE:

- $SP \leftarrow$**Setup**$(1^{\lambda}, n)$: Takes as input the security parameter λ and n the number of documents held by a data owner. Outputs the public system parameters SP, which are omitted from the input of the other algorithms for brevity.
- $(pk, msk) \leftarrow$**KeyGen():** Outputs a public and master-secret key pair (pk, msk) for a data owner registering in the system.
- $\{IC_l\}_{l\in[1,n]} \leftarrow$**Offline_Encrypt**$(pk)$: Takes as input the public key pk and outputs the intermediate ciphertext $\{IC_l\}_{l\in[1,n]}$.
- $C_l \leftarrow$**Online_Encrypt**$(pk, IC_l, l, M, \overrightarrow{KW})$: Takes as input the public key pk, lth intermediate ciphertext IC_l, document index l, message M, and a set of related keyword(s) $\overrightarrow{KW}$. Outputs the corresponding ciphertext $C_l = (\delta_l, C_{l_{\overrightarrow{KW}}}, C_{l_M})$, where δ_l is public information, $C_{l_{\overrightarrow{KW}}}$ keyword-ciphertext and C_{l_M} data ciphertext. Then, the data owner stores the ciphertext C_l on the cloud server.
- $k_{agg} \leftarrow$**Extract**(msk, S): Takes as input the master-secret key msk of the data owner and subset of data classes $S \subseteq \{1, 2, \ldots, n\}$. Outputs the aggregate key k_{agg} which aggregates the search and access rights of all encrypted messages within set S.
- $RL_l \leftarrow$**Revocation**$(\mathcal{U}_l, msk, l)$: Takes as input the set of users' identities $\mathcal{U}_l$ whose delegated rights needs to be revoked over lth document and, master-secret key msk of the data owner. Outputs the revocation list $RL_l = (RL_{l_1}, RL_{l_2})$.
- $Tr \leftarrow$**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. This algorithm is run by the user who holds an aggregate key k_{agg} for document set S and wants to perform search over documents within set S. Then, the user should submit Tr and S to the server.
- $R \leftarrow$**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 $U_{QR} \in U$. The Test algorithm outputs true (1) or false (0) to denote whether the document doc_l contains the set of keywords $\overrightarrow{Q}$. 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:

$$R = \begin{cases} 1, \text{ If } \vec{Q} \vDash \overrightarrow{KW} \\ 0, \text{ otherwise} \end{cases}$$

Otherwise, the algorithm returns NULL.

Let $\overrightarrow{Q} = \{Q_1, Q_2, \ldots, Q_p\} \in KS$ be the set of keywords attached with the trapdoor $Tr \longleftarrow$ TrapdoorGen$(k_{agg}, \overrightarrow{Q})$ and $\overrightarrow{KW} = \{KW_1, KW_2, \ldots, KW_q\} \in KS$ be the set of keywords labeled with the ciphertext $C_l \longleftarrow$ Online_Encrypt$(pk, IC_l, l, M, \overrightarrow{KW})$. Here, p is the number of keywords in the search query and q is the number of keywords attached with the ciphertext.

Test$(Tr, S, l, U_{QR}) = 1$ *iff* $p = q$ and $\overrightarrow{Q} = \overrightarrow{KW}$

Test$(Tr, S, l, U_{QR}) = 0$ *iff* $\overrightarrow{Q} \neq \overrightarrow{KW}$.

- $pub \leftarrow$ **Offline**_*Decrypt*(k_{agg}, S): Takes as input the aggregate key k_{agg} corresponding to the set S and generates the parameters $pub = (\{pub_{1\_l}, pub_{2\_l}\}_{l\in S}, pub_3, pub_4)$. The parameters pub are further used in Online_Decrypt() to decrypt $\{C_l\}_{l\in S}$. For efficiency consideration, the parameters pub for the set S is computed only once.
- $M \leftarrow$ **Online**_*Decrypt*$(k_{agg}, S, l, C_l, pub)$: Takes as input the ciphertext C_l, ciphertext index l, the aggregate key k_{agg} corresponding to the set S, and parameters pub generated from Offline_Decrypt(). The algorithm outputs the decrypted result M iff $l \in S$.

4.5 Security and Functional Goals

The proposed KASE scheme aims to achieve the following security and functional goals: The *compactness* of the scheme is to ensure that the size of the aggregate key should be independent of the number of documents in the scope of the key. Similarly, the size of the query trapdoor should be independent of the number of keywords in the query set. For the *correctness*, the proposed scheme should get correct result, whether ciphertext C_l of any message M with an index l contain keyword(s) $\overrightarrow{Q}$ or not, when giving trapdoor Tr which represents query for keyword(s) within set $\overrightarrow{Q}$. The *privacy* of the scheme ensures that the cloud server or any

third party should not get any additional information for which they are not authorized. The *revocation* property of the scheme ensures that if delegated rights of any user are revoked on the document, then he is no longer permitted to search or access the same. The *controlled searching* ensures that an authorized user or the cloud server cannot perform searches on the documents for which search or access rights are not delegated. Moreover, an authorized user cannot create new aggregate keys for another set of documents from the known one. For the *efficiency*, the above-mentioned goals of privacy and functionality should be achieved with lower bandwidth, computation and storage overhead. The proposed R-OO-KASE scheme must be secure against the *collision resistance* attack. In order to get additional information other than each aggregate key individually contains, a collision attack is carried out by combining multiple aggregate keys.

Definition 2 *(Compactness)* The proposed R-OO-KASE scheme is compact if for any set of documents $|S| = x$ having indices of x different ciphertexts, then $k_{agg} \longleftarrow$ Extract(S, msk) outputs a single aggregate key having constant size. Moreover, for any set of query keywords $\overrightarrow{Q}$, $Tr \longleftarrow$ TrapdoorGen($k_{agg}, \overrightarrow{Q}$) outputs a single trapdoor having constant size.

Definition 3 *(Correctness)* The proposed R-OO-KASE scheme is correct if for any document doc_l containing set of query keywords $\overrightarrow{Q}$, $C_l \longleftarrow$ Online_Encrypt($pk, IC_l, l, M, \overrightarrow{KW}$) and $Tr \longleftarrow$ TrapdoorGen($k_{agg}, \overrightarrow{Q}$), then $1 \longleftarrow$ Test(Tr, S, l, U_{QR}) iff $\overrightarrow{Q} \vDash \overrightarrow{KW}$.

Definition 4 *(Privacy)* The proposed R-OO-KASE scheme is privacy preserving if for any set of query keywords $\overrightarrow{Q}$, searchable trapdoor Tr representing query keywords $\overrightarrow{Q}$ and adversary $\mathcal{A}$ running in PPT, $C_l \longleftarrow$ Online_Encrypt($pk, IC_l, l, M, \overrightarrow{KW}$), $k_{agg} \longleftarrow$ Extract(S, msk) and $Tr \longleftarrow$ TrapdoorGen($k_{agg}, \overrightarrow{Q}$), then the $Pr[\mathcal{A}(param, pk, S, Tr, \delta_l, C_{\overrightarrow{KW}}) = \overrightarrow{Q}]$ and $Pr[\mathcal{A}(param, pk, S, k_{agg}, \delta_l, C_{\overrightarrow{KW}}) = \overrightarrow{KW}]$ are negligible.

Definition 5 *(Revocation)* The proposed R-OO-KASE scheme is revocable if for a set of revoked users' identities $\mathcal{U}_l$ and document index l, the data owner having master-secret key generates a revocation list $RL_l \longleftarrow Revocation(\mathcal{U}_l, msk, l)$ and any query requester having user id $U_{QR} \in \mathcal{U}_l$ submits the trapdoor to the cloud server $Tr \longleftarrow$ TrapdoorGen($k_{agg}, \overrightarrow{Q}$), then $\perp \longleftarrow$ Test(Tr, S, l, U_{QR}).

Definition 6 *(Controlled Searching)* The proposed R-OO-KASE scheme provides controlled searching if for dataset S and set of query keywords $\overrightarrow{Q}$, $k_{agg} \longleftarrow$ Extract(S, msk), $Tr \longleftarrow$ TrapdoorGen($k_{agg}, \overrightarrow{Q}$) and $l \notin S$, then $\perp \longleftarrow$ Test(Tr, S, l, U_{QR}).

Definition 7 *(Efficiency)* The proposed R-OO-KASE scheme is efficient if $O(Computation\_Time_{Online\_Encrypt})$, $O(Computation\_Time_{Extract})$ and $O(Computation\_Time_{Revocation})$ at the data owner side and $O(Computation\_Time_{TrapdoorGen})$, $O(Computation\_Time_{Online\_Decrypt})$ at the user side is $\theta(1)$ or $\theta(c)$ for some constant c.

4.6 Security Model

We consider the cloud servers and the 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 the storage space and the computing power. Moreover, communication channels involving the server are assumed to be insecure.

The goals of an adversary considered for the proposed scheme are as follows:

- Retrieve the information about the keywords related to the ciphertext.
- Gain the information about the query keyword(s) $\overrightarrow{Q}$ by looking at the search trapdoor Tr.
- Retrieve the plaintext or try to search over the ciphertext using a single or combination of aggregate keys such that none of them have rights for it.

To prove the privacy of the proposed R-OO-KASE scheme against defined attacks, we consider two security notions: ciphertext privacy and trapdoor privacy.

- *Ciphertext privacy* The keyword ciphertext does not reveal any information about the corresponding keyword to the attacker who is the unauthorized user. We prove this claim by the security of Indistinguishability against the Chosen Keyword Attack (IND-CKA) model. IND-CKA model ensures that an adversary cannot obtain the relationship between the challenge ciphertext and the corresponding keyword.
- *Trapdoor privacy* The trapdoor does not reveal any information about the corresponding keyword to the attacker who does not possess the authorization keys. We prove this claim by the Indistinguishability against Keyword Guessing Attack (IND-KGA) model. IND-KGA model ensures that an adversary cannot find the relationship between the challenge trapdoor and the corresponding keyword. The size of the guessing keyword dictionary equals to the number of keywords in the attacked file.

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.

4.6.1 IND-CKA Model

We need to ensure that Test(Tr, S, l, U_{QR}) does not reveal any information about keywords $\overrightarrow{KW}$ labeled with the ciphertext $C_{l_{\underrightarrow{KW}}}$ unless trapdoor Tr is available. We define security against an active adversary $\mathcal{A}$ who is able to obtain trapdoor Tr for any set of keywords $\overrightarrow{Q}$ of his choice. Even under such attack the attacker should not be able to distinguish an encryption of a keyword $\overrightarrow{KW_0}$ from an encryption of a keyword $\overrightarrow{KW_1}$ for which he did not obtain the trapdoor. We say that the proposed R-OO-KASE scheme hold the privacy for keyword if no polynomial bounded adversary $\mathcal{A}$ has a non-negligible advantage against the challenger in the following IND-CKA game. The game proceeds as follows:

Init Phase

The adversary $\mathcal{A}$ selects a challenge index i_c for which he wishes to be challenged upon.

Setup

The challenger runs Setup($1^\lambda, n$) to generate the system parameters and KeyGen() to obtain a public/ master-secret key pair (pk, msk) . It issues public key pk to the adversary $\mathcal{A}$.

Query Phase 1

The adversary $\mathcal{A}$ adaptively queries $q_1, \ldots, q_{n'}$ to following oracles and oracle answers in polynomial time.

- *Aggregate key generation oracle* $\mathcal{O}_{k_{agg}}$ On giving inputs of (msk, S) by the adversary, where msk is master secret key corresponding to pk, oracle returns an aggregate key $k_{agg} \leftarrow$ Extract(msk, S). The restriction is that an adversary cannot ask for the aggregate key for challenge index, i.e., if $i_c \in S$, then $\mathcal{O}_{k_{agg}}$ returns null. The oracle adds aggregate key (k_{agg}, S) in table $T_{k_{agg}}$.
- *Offline_Encryption oracle* $\mathcal{O}_{Off\_Encrypt}$ On giving input of pk by the adversary, the oracle generates intermediate ciphertext $\{IC_l\}_{l\in[1,n]}$ using the Offline_Encrypt(pk) algorithm.
- *Revocation oracle* $\mathcal{O}_{RL}$ On giving inputs of ($\mathcal{U}_l, msk, l$) by the adversary, where msk is master secret key corresponding to pk and document index $l \leq n$, then oracle returns a revocation list $RL_l \leftarrow$Revocation($\mathcal{U}_l, msk, l$) to the $\mathcal{A}$.
- *Trapdoor generation oracle* $\mathcal{O}_{Tr}$ On giving inputs of ($k_{agg}, \overrightarrow{Q}$), the challenger checks k_{agg} occurs in table $T_{k_{agg}}$, if yes, the oracle runs $Tr \leftarrow$TrapdoorGen($k_{agg}, \overrightarrow{Q}$), and returns the trapdoor Tr for query set $\overrightarrow{Q}$ to $\mathcal{A}$.
- *Test oracle* $\mathcal{O}_{test}$ On giving inputs of (Tr, S, l, U_{QR}), the challenger checks authorization of user U_{QR}, if the delegated rights of user U_{QR} is not revoked and $i_c \notin S$, the oracle returns output of Test(Tr, S, l, U_{QR}) to the adversary. Otherwise, it returns *null*.
- *Offline_Decryption oracle* $\mathcal{O}_{Off\_Decrypt}$ The oracle computes the parameters $pub \leftarrow$Offline_Decrypt(k_{agg}, S) that are used to decrypt the ciphertext within given set S.
- *Online_Decryption oracle* $\mathcal{O}_{On\_Decrypt}$ On giving inputs of (k_{agg}, S, l, C_l, pub), the challenger checks if k_{agg} occurs in table $T_{k_{agg}}$, if yes, then the oracle returns $M \leftarrow$Online_Decrypt(k_{agg}, S, l, C_l, pub) 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. Here, the restriction is that the attacker had not previously asked for the query trapdoor corresponding to keywords $\overrightarrow{KW_0}$;$\overrightarrow{KW_1}$ to the oracle $\mathcal{O}_{Tr}$. The challenger chooses β randomly from $\{0, 1\}$ and runs Online_Encrypt($pk, IC_{i_c}, i_c, M, \overrightarrow{KW_\beta}$). The challenger returns keyword ciphertext C^*_β to the adversary.

Query Phase 2

The adversary $\mathcal{A}$ asks for more queries $q_{n'+1}, \ldots, q_{n'_2}$ to oracles and oracle answers in polynomial time. The oracles are identical to that in the query phase 1 except the following:

- *Trapdoor generation oracle* $\mathcal{O}_{Tr}$ On giving inputs 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 $\overrightarrow{KW_1}$
  - $\overrightarrow{Q} \vDash \overrightarrow{KW_0}$ or $\overrightarrow{Q} \vDash \overrightarrow{KW_1}$

  If one of the above condition holds, then challenger returns *null*.
- *Test oracle* $\mathcal{O}_{test}$ 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 $ADV^{R-OO-KASE}_{\mathcal{A},K} = |Pr[\beta = \beta'] - \frac{1}{2}|$, where the probability is taken over the random bits used by the challenger and the adversary.

Definition 8 *(IND-CKA Security)* We say that the proposed R-OO-KASE scheme is CKA secure if for any polynomial time adversary $\mathcal{A}$, we have that $|ADV^{R-OO-KASE}_{\mathcal{A},K}| \leq \epsilon$.

4.6.2 IND-KGA Model

We need to ensure that Tr does not reveal any information about the corresponding query keywords $\overrightarrow{Q}$ to the attacker who does not possess the authorization key k_{agg}. We define security against an active adversary $\mathcal{A}$ who is able to obtain almost all the trapdoor Tr except the two specified set of query keywords $\overrightarrow{Q_0}$ and $\overrightarrow{Q_1}$. Even under such attack the attacker cannot find the relationship of the challenge trapdoor and the corresponding keyword. We say that the proposed R-OO-KASE scheme hold the security against keyword guessing attack if no polynomial bounded adversary $\mathcal{A}$ has a non-negligible advantage against the challenger in the following IND-KGA game. The game proceeds as follows:

Init Phase

The adversary $\mathcal{A}$ selects a challenge index i_c for which he wishes to be challenged upon.

Setup

The challenger runs Setup($1^\lambda, n$) to generate the system parameters and KeyGen() to obtain a public/ master-secret key pair (pk, msk) . It issues public key pk to the adversary $\mathcal{A}$.

Query Phase 1

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

- *Online_Encryption oracle* $\mathcal{O}_{On\_Encrypt}$ The attacker $\mathcal{A}$ sends the challenger a set of keywords $\overrightarrow{KW}$ along with an index l, a message M from the message space, a public key pk. If $l \neq i_c$, then the challenger runs Online_Encrypt($pk, IC_l, l, M, \overrightarrow{KW}$) and returns ciphertext C_l to the adversary.
- *Test oracle* $\mathcal{O}_{test}$ On giving input of (Tr, S, l, U_{QR}), the challenger checks $i_c \notin S$ and $l \neq i_c$, 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 of query keyword $\overrightarrow{Q_0}$;$\overrightarrow{Q_1}$ 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 keyword $\overrightarrow{Q_0}$;$\overrightarrow{Q_1}$ to the oracle $\mathcal{O}_{On\_Encrypt}$. 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

The adversary $\mathcal{A}$ asks for more queries $q_{n'+1}, \ldots, q_{n'_2}$ to oracles and oracle answers in polynomial time. The oracles are identical to that in the query phase 1 except the following:

- The adversary $\mathcal{A}$ cannot ask for the aggregate key corresponding to a set S which includes an index i_c
- Ciphertext query on set of keywords $\overrightarrow{Q_0}$ 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 $ADV^{R-OO-KASE}_{\mathcal{A},KG} = |Pr[\beta = \beta'] - \frac{1}{2}|$, where the probability is taken over the random bits used by the challenger and the adversary.

Definition 9 *(IND-KGA Security)* We say that the proposed R-OO-KASE method hold the privacy for trapdoor if $|ADV^{R-OO-KASE}_{\mathcal{A},KG}| \leq \epsilon$ is negligible with respect to the security parameter for any polynomial time adversary.

5 R-OO-KASE : Revocable Online/offline Key Aggregate Multi-Keyword Searchable Encryption over Multi-owners' Data

In this section, we discuss the construction of the proposed scheme. In the following discussion, Z_p is a group of large prime order p. Group G and G_T are cyclic multiplicative group of prime order p. The other notations used in the forthcoming discussion are already introduced in Sect. 3.

The detailed construction of the proposed R-OO-KASE scheme is as follows:

A. *System Initialization* The data owner first sets up an account on the cloud server. At the time of system initialization, TA assigns a unique user identity $U_{id} \in G$ to each cloud user. Let we assume that there are total X users in the system and $U = \{U_1, \ldots U_X\}$ is a set of all users' identities. For each $U_{id} \in U$, $\{U_{id} = g^{\beta^{id}}\}_{id \in [1,X]}$, where $\beta \in Z_p$. If a new user gets registered in the system, a unique user identity $U_{X+1} = g^{\beta^{X+1}}$ will be assigned to him and U_{X+1} will be added to the set U . The data owner establishes the public system parameter via **Setup()** and generates a public/master-secret key pair via **KeyGen()**.

1. *Setup* $(1^\lambda, n)$ The data owner runs this algorithm and publishes the system parameters $SP = (B, PubK, H)$. 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 the data owner. The random secret element $\alpha \in Z_p$.

Algorithm 1 Setup $(1^{\lambda}, n) \rightarrow SP$

1: Generate a bilinear map group system $B=(p,G,G_T,e(.,.))$
2: Generate public parameters $PubK \leftarrow (g, g_1, ..,g_n,g_{n+2}, ...,g_{2n})$, where, $g_l = g^{\alpha^l} \in G \ \forall l \in \{1,2,..,n,n+2,\ldots,2n\}$
3: Select a one-way, collision resistant hash function $H : \{0,1\}^* \times \{0,1\}^* \rightarrow G$
4: $SP \leftarrow (B, PubK, H)$
5: $KS \leftarrow^m \{0,1\}^*$

2. *KeyGen()* The data owner runs this algorithm to generate a key pair (*pk*, *msk*)

$$pk = v = g^{\gamma}; msk = \gamma, \text{ where } \gamma \in Z_p$$

The public key *pk* is used for encrypting keywords and messages. The master-secret key *msk* for delegation of search rights is kept private by the owner.

B. *Storage Scenario* The data owner executes offline encryption algorithm (**refer to Offline_Encrypt()**) using only public key and perform the majority of costly operations before knowing a message and keywords to be encrypted. Further, the online encryption (**refer to Online_Encrypt()**) is performed once the message and keywords to be encrypted are known.

3. *Offline_Encrypt* (*pk*) The data owner gives public key $pk = v = g^{\gamma}$ as input and generates intermediate ciphertext $\{IC_l\}_{l\in[1,n]}$ using the following offline encryption algorithm.

Algorithm 2 Offline Encrypt $(pk) \rightarrow \{IC_l\}_{l\in[1,n]}$

1: **for** $l \leftarrow 1$ to n **do**
2: Select $t_l \in Z_p$
3: $IC_{l_1} \leftarrow g^{t_l}$
4: $IC_{l_2} \leftarrow (v \cdot g_l)^{t_l}$
5: $IC_{l_3} \leftarrow e(g_1, g_n)^{t_l}$
6: **for all** $w_{i,j} \in KS$ **do**
7: $IC_{l_{i,j}} = \{(H(w_{i,j}))^{t_l}\}_{i\in[1,m],j\in[1,\mathcal{N}_i]}$
8: $IC_l = \{IC_{l_1}, IC_{l_2}, IC_{l_3}, \{IC_{l_{i,j}}\}_{i\in[1,m],j\in[1,\mathcal{N}_i]}\}_{l\in[1,n]}$

4. *Online_Encrypt* $(pk, IC_l, l, M, \overrightarrow{KW})$ In the online phase, the data owner selects *l*th intermediate ciphertext IC_l to generate the ciphertext C_l for *l*th document.

Algorithm 3 Online Encrypt $(pk, IC_l, l, M, KW) \rightarrow C_l$

1: **if** $l \in \{1,2,\ldots,n\}$ **then**
2: $\delta_l = (C_{l_1}, C_{l_2}) \leftarrow (IC_{l_1}, IC_{l_2})$
3: $C_{l_{\overrightarrow{KW}}} = C_{l_3} \leftarrow \Pi_{w_{i,j}\in\overrightarrow{KW}}(IC_{l_{i,j}})$
4: $C_{l_M} = C_{l_4} \leftarrow M \cdot IC_{l_3}$, where $M \in G_T$
5: $C_l = (\delta_l, C_{l_{\overrightarrow{KW}}}, C_{l_M}) = (C_{l_1}, C_{l_2}, C_{l_3}, C_{l_4})$
6: Data owner stores ciphertext C_l on the cloud server.

The data ciphertext C_{l_M} is generated by multiplying message *M* with the term $e(g_1, g_n)^{t_l}$. The term $g_{n+1} = g^{\alpha^{n+1}}$ is missing in the public parameters *PubK*. Therefore, an attacker cannot compute the value of $e(C_{l_1}, g_{n+1})$ to get the value of $e(g_1, g_n)^{t_l}$. An attacker cannot get any information related to message *M*.

Algorithm 3 clearly indicates that the Online_Encrypt() requires only multiplication operations in linear with the number of keywords attached with the ciphertext.

C. *Sharing Scenario* In order to delegate search and access rights on the selected set of documents *S* to other users, the data owner generates an aggregate key **(refer to Extract())**. Using an aggregate key an authorized user can retrieve and access all the shared documents.

5. *Extract* (*msk*, *S*) The data owner runs this algorithm using master-secret key $msk = \gamma$, subset $S \subseteq \{1, \ldots, n\}$ which contains documents' indices. 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 in order to delegate the keyword search rights of ciphertexts in the range of set *S*.

D. *Revocation Scenario* If the shared documents are modified or if the delegated rights for the shared documents need to be revoked, the data owner can generate revocation list **(refer to Revocation())**.

6. *Revocation*$(\mathcal{U}_l, msk, l)$ The proposed scheme allows the data owner to revoke delegated rights of selected users within set $\mathcal{U}_l$ over *l*th document, without affecting the users' rights over rest of the shared documents. The data owner generates the revocation list RL_l for *l*th document using his master-secret key *msk* and set of revoked users' identities $\mathcal{U}_l$. The cloud server checks the user's authorization each time when he received a search and data access request for any stored document. If the identity of the query requester U_{QR} matches with the revocation list RL_l, then the failure state is returned to the user, otherwise, the cloud server performs the search process over *l*th document. Suppose, after storing the revocation list RL_l on the cloud server, the data owner wants to revoke rights of users within set $\mathcal{U}_l'$ along with the users in set $\mathcal{U}_l$. The

data owner will generate new revocation list $RL_l^{'}$ considering revoked users set $\mathcal{U}_l \bigcup \mathcal{U}_l^{'}$. The data owner will upload a new revocation list $RL_l^{'}$ to the cloud server. On receiving new revocation list for *l*th document from the data owner, the cloud server will store a new revocation list $RL_l^{'}$ and discard the old one RL_l.

Algorithm 4 Revocation($\mathcal{U}_l$,msk,l) $\rightarrow RL_l$

1: Select $r \in Z_p$
2: $RL_{l_1} \leftarrow g_l^{r\gamma}$
3: $RL_{l_2} \leftarrow (\Pi_{\mathcal{U}_{l_i} \in \mathcal{U}_l} \mathcal{U}_{l_i}^{\gamma})^r$
4: $RL_l \leftarrow (RL_{l_1}, RL_{l_2})$
5: Data owner stores revocation list RL_l on the cloud server.

E. *Search Scenario* The user who holds an aggregate key k_{agg} for dataset *S* and wants to retrieve documents having a set of keywords $\overrightarrow{Q}$ from the shared dataset, generates the trapdoor *Tr* for query keywords **(refer to TrapdoorGen())**. Then, the user submits the trapdoor *Tr* and *S* to the server. The cloud server performs the keyword search on behalf of the user, using the submitted trapdoor *Tr* and returns the search results to the user **(refer to Test())**.

7. *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 scope of an aggregate key k_{agg}, then the user having trapdoor *Tr* can search over any document in the range of set *S*.

Algorithm 5 TrapdoorGen(k_{agg}, Q)$\rightarrow Tr$

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

8. *Test* (Tr, S, l, U_{QR}) On receiving the trapdoor *Tr* from the query requester, the cloud server runs Test() algorithm to check if document doc_l contains query keyword(s) $\overrightarrow{Q}$ or not. 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.

Algorithm 6 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: $R = \mathrm{e}(Tr_0 \cdot pub_1', C_{l_1})$ / $\mathrm{e}(pub_2', C_{l_2})$ $\mathrm{e}(C_{l_{\overrightarrow{KW}}}, Tr_1)$
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

Correctness:

The correctness of Test algorithm can be realized as follows:

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

$$= e(RL_{l_2}, g_l)/e(U_{QR} \cdot pub', RL_{l_1})$$
$$= e((\Pi_{\mathcal{U}_{l_i} \in \mathcal{U}_l} \mathcal{U}_{l_i})^{\gamma r}, g_l)/e(U_{QR} \cdot p\Pi_{\mathcal{U}_{l_i} \in \mathcal{U}_l, \mathcal{U}_{l_i} \neq U_{QR}} \mathcal{U}_{l_i}, g_l^{\gamma r})$$
$$= 1$$

Therefore, if the query requester is revoked user 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:

$$= \frac{e(Tr_0 \cdot pub_1', C_{l_1})}{e(pub_2', C_{l_2})\ e(C_{l_{\overrightarrow{KW}}}, Tr_1)}$$
$$= \frac{e(k_{agg} \Pi_{w_{i,j} \in \overrightarrow{Q}} (H(w_{i,j}))^b \cdot \Pi_{j \in S} g_{j+l}, g^{t_l})}{e(\Pi_{j \in S} g_j, (v \cdot g_l)^{t_l})\ e(\Pi_{w_{i,j} \in \overrightarrow{KW}} (H(w_{i,j}))^{t_l}, g^b)}$$
$$= \frac{e(k_{agg}, g^{t_l})\ e(\Pi_{w_{i,j} \in \overrightarrow{Q}} (H(w_{i,j}))^b, g^{t_l})\ e(\Pi_{j \in S} g_{j+l}, g^{t_l})}{e(\Pi_{j \in S} g_j, g^{\gamma t_l})\ e(\Pi_{j \in S} g_j, g_l^{t_l})\ e(\Pi_{w_{i,j} \in \overrightarrow{KW}} (H(w_{i,j}))^{t_l}, g^b)}$$
$$= 1$$

F. *Online/Offline Decryption* The cloud server sends search results to the query requester. The search result contains a true or false value for each document $l \in S$, indicating whether document contains keywords within query set $\overrightarrow{Q}$ or not. Then, the query requester only requires to download matching documents that are satisfying search query, instead of downloading all

the shared documents. The user having an aggregate key can decrypt the retrieved data. The offline phase of the decryption (**refer to Offline_Decrypt()**) computes the parameters $pub = (\{pub_{1\_l}, pub_{2\_l}\}_{l\in S}, pub_3, pub_4)$, which are used to decrypt $\{C_l\}_{l\in S}$. In the online phase (**refer to Online_Decrypt()**), the ciphertext is decrypted with minimal computation overhead.

9. *Offline_Decrypt* (k_{agg}, S) The user who holds an aggregate key k_{agg} for dataset S can precompute the parameters that are required to decrypt ciphertexts within range of set S, in offline mode. This algorithm generates the parameters $pub = (\{pub_{1\_l}, pub_{2\_l}\}_{l\in S}, pub_3, pub_4)$. 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.

$$\{pub_{1\_l}\}_{l\in S} = \{\Pi_{j\in S}\, g_{j+l}\}_{l\in S}$$

$$\{pub_{2\_l}\}_{l\in S} = \{\Pi_{j\in S, j\neq l}\, g_{n+1-j+l}\}_{l\in S}$$

$$pub_3 = \Pi_{j\in S} g_j$$

$$pub_4 = \Pi_{j\in S} g_{n+1-j+l}$$

10. *Online_Decrypt* $(k_{agg}, S, l, C_l, pub)$ If $l \notin S$, Online_Decrypt() algorithm outputs NULL. Otherwise, returns the output:

$$M = C_{l_4}\, \mathrm{e}(k_{agg}\ pub_{1\_l}\ pub_{2\_l}\ , C_{l_1})\ /\ \mathrm{e}(pub_3, C_{l_2})\ \mathrm{e}(pub_4, C_{l_1})$$

Correctness:

$$C_{l_4}\ \mathrm{e}(k_{agg}\ pub_{1\_l}\ pub_{2\_l}\ , C_{l_1})\ /\ \mathrm{e}(pub_3, C_{l_2})\ \mathrm{e}(pub_4, C_{l_1})$$

$$= \frac{C_{l_4}\quad e(k_{agg}\cdot \Pi_{j\in S} g_{j+l}\cdot \Pi_{j\in S, j\neq l} g_{n+1-j+l}, g^{t_l})}{e(\Pi_{j\in S} g_j, (v\cdot g_l)^{t_l}) e(\Pi_{j\in S} g_{n+1-j+l}, g^{t_l})}$$

$$= \frac{C_{l_4}\quad e(\Pi_{j\in S} g_{j+l}, g^{t_l})\ e(\Pi_{j\in S, j\neq l} g_{n+1-j+l}, g^{t_l})}{e(\Pi_{j\in S} g_j, g_l^{t_l})\ e(\Pi_{j\in S} g_{n+1-j+l}, g^{t_l})}$$

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

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

$$= M$$

6 Security Analysis

We prove the security of the proposed scheme against the IND-CKA, IND-KGA and cross-pairing attack in the generic group model using the DDH hardness assumption.

Theorem 1 *The proposed R-OO-KASE scheme is IND-CKA secure, assuming that the DDH problem is hard to solve.*

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 ϵ to break the privacy of our scheme.

Then, we can construct a simulator $\mathcal{SIM}$ that breaks the decisional DDH problem $\varsigma = (g, g^a, g^b, g^c, R)$ with the advantage $\frac{\epsilon}{2}(1 - \frac{N^2}{p})$.

Here, we assume that for trapdoors $Tr_{\overrightarrow{Q}}$ which denotes search query of keyword set $\overrightarrow{Q}$ and $Tr_{\overrightarrow{Q'}}$ which denotes query of keyword set $\overrightarrow{Q'}$; $Tr_{\overrightarrow{Q}} \neq Tr_{\overrightarrow{Q'}}$. If there exists $\overrightarrow{Q}$ and $\overrightarrow{Q'}, \overrightarrow{Q} \neq \overrightarrow{Q'}$ such that $Tr_{\overrightarrow{Q}} = Tr_{\overrightarrow{Q'}}$, then $Tr_{\overrightarrow{Q}}$ can search over keyword-ciphertext $C_{\overrightarrow{KW'}}$, where $Tr_{\overrightarrow{Q'}} \vDash C_{\overrightarrow{KW'}}$ and $Tr_{\overrightarrow{Q}}| \neq C_{\overrightarrow{KW'}}$. This assumption hold with the probability

$$\frac{p(p-1)\ldots(p-(N-1))}{p^N} > \frac{(p-(N-1))^N}{p^N}$$
$$= (1 - \frac{N-1}{p})^N > (1 - \frac{N(N-1)}{p}) > (1 - \frac{N^2}{p})$$

where N is number of possible keywords in the system and p is the order of group G.

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 $G \times G \to G_T$. g is a generator of group G. The challenger $\mathcal{C}$ computes v as follows:

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

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 challenge index i_c for which he wishes to be challenged upon.

Setup

The simulator $\mathcal{SIM}$ generates public parameters $PubK = (g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$.

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

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

- $\mathcal{O}_H(w_{i,j})$: 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 $\mathcal{O}_{k_{agg}}, \mathcal{O}_{Tr}, \mathcal{O}_{test}$. 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 $T^*_{w_{i,j}}$

$\mathcal{SIM}$ sets public-key $pk = v = (g^c)^u$ where $u \in_R Z_p$. Finally, $\mathcal{SIM}$ records the tuple $< pk, u >$ in table T_k. Table T_k is used to records the tuple $< pk, u >$ and these records will be used in other oracles to response the queries. The simulator $\mathcal{SIM}$ sends the public key pk to $\mathcal{A}$.

Query phase 1

The simulator $\mathcal{SIM}$ constructs following oracles and adversary $\mathcal{A}$ can adaptively queries $q_1, \ldots, q_{n'}$ to these oracles in polynomial for multiple times.

- *Aggregate key generation oracle* $\mathcal{O}_{k_{agg}}$ On giving inputs of (msk, S) by the adversary, where msk is master secret key corresponding to pk, the simulator checks that key pair (pk, msk) occurs in table T_k, if not, $\mathcal{SIM}$ reports failure and terminates. Otherwise, it returns an aggregate key $k_{agg} \leftarrow \text{Extract}(msk, S)$ and add the tuple $< k_{agg}, msk, S >$ 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. The $\mathcal{SIM}$ add aggregate key (k_{agg}, S) in table $T_{k_{agg}}$.
- *Offline_Encryption oracle* $\mathcal{O}_{Off\_Encrypt}$ On giving input of pk by the adversary, the oracle generates intermediate ciphertext $\{IC_l\}_{l\in[1,n]}$ using the Offline_Encrypt(pk) algorithm.
- *Revocation oracle* $\mathcal{O}_{RL}$ On giving inputs of $(\mathcal{U}_l, msk, l)$ by the adversary, where msk is master secret key corresponding to pk and document index $l \le n$, then oracle returns a revocation list $RL_l \leftarrow$Revocation$(\mathcal{U}_l, msk, l)$ to the $\mathcal{A}$.
- *Trapdoor generation oracle* $\mathcal{O}_{Tr}$ On giving inputs $(k_{agg}, \overrightarrow{Q})$, the simulator first checks k_{agg} occurs in table $T_{k_{agg}}$, if yes, then it proceeds as follows:
  - For each $w_{i,j} \in \overrightarrow{Q}$, the simulator queries $\mathcal{O}_H(w_{i,j})$ and obtain $(w_i, c_i, w_{i,j}, T^*_{w_{i,j}})$
  - If $c_i = 1$, set trapdoor $Tr \leftarrow TrapdoorGen(k_{agg}, \overrightarrow{Q})$
  - Otherwise, report failure and terminate.
- *Test oracle* $\mathcal{O}_{test}$ On giving inputs of (Tr, S, l, U_{QR}), the challenger checks authorization of user U_{QR}, if the delegated rights of user U_{QR} is not revoked and $i_c \notin S$, the oracle returns output of Test(Tr, S, l, U_{QR}) to the adversary. Otherwise, it returns *null*.
- *Offline_Decryption oracle* $\mathcal{O}_{Off\_Decrypt}$ The oracle computes the parameters $pub \leftarrow$Offline_Decrypt(k_{agg}, S) that are used to decrypt the ciphertext within given set S corresponding to given aggregate key k_{agg}.
- *Online_Decryption oracle* $\mathcal{O}_{On\_Decrypt}$ On giving inputs of $(k_{agg}, S, l, C_l, pub)$, the challenger checks if k_{agg} occurs in table $T_{k_{agg}}$, if yes, then the oracle returns $M \leftarrow$Online_Decrypt$(k_{agg}, S, l, C_l, pub)$ to the adversary. Otherwise, it returns *null*.

Challenge Phase

$\mathcal{A}$ outputs two set of keywords $\overrightarrow{KW_0}, \overrightarrow{KW_1}$ on which it wishes to be challenged, message M, public key pk and index i_c. We consider only the case where $\overrightarrow{Q}| \neq \overrightarrow{KW_0} \bigwedge \overrightarrow{Q}| \neq \overrightarrow{KW_1}$.

The reason for this is

if $\overrightarrow{KW_0} = \overrightarrow{KW_1}$ OR $(\overrightarrow{Q} \vDash \overrightarrow{KW_0} \bigwedge \overrightarrow{Q} \vDash \overrightarrow{KW_1})$, then $\mathcal{SIM}$ simply aborts and takes a random guess. The probability that the simulator $\mathcal{SIM}$ aborts is $Pr[abort] = N^2/p$

Otherwise, $\mathcal{SIM}$ randomly chooses a bit $b \in \{0,1\}$ and compute keyword-ciphertext for $\overrightarrow{KW_b}$.

Challenger $\mathcal{C}$ sets $t = a$ and computes ciphertext by running Online_Encrypt$(pk, IC_{i_c}, i_c, M, \overrightarrow{KW_b})$

$C^*_1 = g^a$; $C^*_2 = (v \cdot g_{i_c})^a$

$C^*_3 = C_{\overrightarrow{KW}} = \Pi_{w_{i,j} \in \overrightarrow{KW_b}}(H(w_{i,j}))^a$

$C^*_4 = C_M = M\, e(g_1, g_n)^a$

Query phase 2

The adversary $\mathcal{A}$ asks for more queries $q_{n'+1}, \ldots, q_{n'_2}$ to oracles and oracle answers in polynomial time. The oracles are identical to that in the query phase 1 except the following:

- *Trapdoor generation oracle* $\mathcal{O}_{Tr}$ On giving inputs 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* $\mathcal{O}_{test}$ 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

$\mathcal{A}$ outputs a guess $b' \in_R \{0, 1\}$.

If $b' = b$, then $\mathcal{SIM}$ outputs $v' = 0$. If $b' \neq$ b , then $\mathcal{SIM}$ outputs $v' = 1$. Based on this, there will be two cases as follows:

case (i) if $v = 0$, then $Z = g^{abc}$, and hence, challenge ciphertext is a correct ciphertext of keyword KW_b.

- $\therefore \mathcal{A}$ outputs $b' = b$ with an advantage ϵ
- $\therefore Pr[b' = b | v = 0 \wedge \overline{abort}] = 1/2 + \epsilon$
- $\therefore Pr[v' = v | v = 0 \wedge \overline{abort}] = 1/2 + \epsilon$ because $\mathcal{SIM}$ guesses $v' = 0$ when $b' = b$

case (ii) if $v = 1$, then the challenge ciphertext is independent of $\overrightarrow{KW_0}$ and $\overrightarrow{KW_1}$, so that $\mathcal{A}$ cannot obtain any information of b.

- $\therefore \mathcal{A}$ outputs $b' \neq b$ with NO knowledge
- $\therefore Pr[b' \neq b | v = 1 \wedge \overline{abort}] = 1/2$
- $\therefore Pr[v' = v | v = 1 \wedge \overline{abort}] = 1/2$ because $\mathcal{SIM}$ guesses $v' = 1$ when $b' \neq b$

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

$$
\begin{aligned}
&Pr[v' = v] - \frac{1}{2} \\
&\quad = Pr[v = 0]Pr[v' = v | v = 0] + Pr[v = 1]Pr[v' = v | v = 1] - \frac{1}{2} \\
&\quad = \frac{1}{2}Pr[v' = v | v = 0] + \frac{1}{2}Pr[v' = v | v = 1] - \frac{1}{2} \\
&\quad = \frac{1}{2}\{Pr[v' = v | v = 0] + \Pr[v' = v | v = 1] - 1\} \\
&\quad = \frac{1}{2}\{Pr[abort]Pr[v' = v | v = 0 \wedge abort] + Pr[\overline{abort}]Pr[v' = v | v = 0 \wedge \overline{abort}] + Pr[abort]Pr[v' = v | v = 1 \wedge abort] \\
&\qquad + Pr[\overline{abort}]Pr[v' = v | v = 1 \wedge \overline{abort}] - 1\}
\end{aligned}
$$

Because the event "abort" is independent of DDH challenge, we have

$$
\begin{aligned}
&Pr[v' = v | v = 0 \wedge abort] = Pr[v' = v | v = 1 \wedge abort] = \frac{1}{2} \\
&\quad = \frac{1}{2}\left\{\frac{N^2}{p}\frac{1}{2} + \left(1 - \frac{N^2}{p}\right)\left(\frac{1}{2} + \epsilon\right) + \frac{N^2}{p}\frac{1}{2} + \left(1 - \frac{N^2}{p}\right)\frac{1}{2} - 1\right\} \\
&\quad = \frac{1}{2}\left\{\left(1 - \frac{N^2}{p}\right)\epsilon\right\} \\
&\quad = \frac{\epsilon}{2}\left(1 - \frac{N^2}{p}\right)
\end{aligned}
$$

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 $= \frac{\epsilon}{2}(1 - \frac{N^2}{p})$, which is an intractable problem. □

Theorem 2 *The proposed R-OO-KASE scheme is IND-KGA secure under the DDH assumption, assuming that the DDH problem is hard to solve.*

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 $\epsilon(l)$ to break the privacy of our scheme. Then, we can construct simulator $\mathcal{SIM}$ that breaks the decisional DDH problem $\varsigma = (g, g^a, g^b, g^c, R)$ with the advantage $\epsilon(l)/2$

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 v as follows:

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

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 challenge index i_c for which he wishes to be challenged upon.

Setup

The simulator $\mathcal{SIM}$ generates public parameters $PubK = (g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$.

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

Moreover, $\mathcal{SIM}$ simulates the hash oracle to map the given keyword with unique value. The hash oracle is identical to that in the IND-CKA security model.

$\mathcal{SIM}$ sets public key $pk = v = (g^c)^u$ where $u \in_R Z_p$. Finally, $\mathcal{SIM}$ records the tuple $< pk, u >$ in table T_k. Table T_k is used to records the tuple $< pk, u >$, and these records will be used in other oracles to response the queries. The simulator $\mathcal{SIM}$ sends the public key pk to $\mathcal{A}$.

Query Phase 1

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

- *Online_Encryption oracle* $\mathcal{O}_{On\_Encrypt}$ 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. If $l \neq i_c$, then the challenger runs Online_Encrypt$(pk, IC_l, l, M, \overrightarrow{KW})$ and returns ciphertext C_l to the adversary.
- *Test oracle* $\mathcal{O}_{test}$ On giving input of (Tr, S, l, U_{QR}), the challenger checks $i_c \notin S$ and $l \neq i_c$, 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 of query keyword $\overrightarrow{Q_0}; \overrightarrow{Q_1}$ 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 keyword $\overrightarrow{Q_0}; \overrightarrow{Q_1}$ to the oracle $\mathcal{O}_{On\_Encrypt}$.

Otherwise, $\mathcal{SIM}$ randomly chooses a bit $u \in \{0,1\}$ and computes Trapdoor Tr_u^*.

The challenger runs TrapdoorGen$(k_{c_{agg}}, \overrightarrow{Q_u})$ and computes trapdoor Tr_u^* for $\overrightarrow{Q_u}$ as follows:

$Tr_0 \leftarrow k_{c_{agg}} \cdot \Pi_{w_{i,j} \in \overrightarrow{Q_u}} (H(w_{i,j}))^a$

$Tr_1 \leftarrow g^a$

Query Phase 2

The adversary $\mathcal{A}$ asks for more queries $q_{n'+1}, \ldots, q_{n'_2}$ to oracles and oracle answers in polynomial time. The oracles are identical to that in the query phase 1 except the following:

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

Guess $\mathcal{A}$ outputs a guess $u' \in_R \{0, 1\}$.

If $u' = u$ $(Z = g^{abc})$, then $\mathcal{SIM}$ outputs $v' = 0$. If $u' \neq u$, then $\mathcal{SIM}$ outputs $v' = 1$ to indicate trapdoor is a random element. Therefore, $\mathcal{A}$ gains no information about v, in turn, $Pr[u \neq u' | v = 1] = 1/2$. As, the simulator $\mathcal{SIM}$ guesses $v' = 1$ when $u \neq u'$, $Pr[v = v' | v = 1] = 1/2$.

If $v = 0$, then $\mathcal{A}$ is able to view the valid trapdoor components with advantage $\epsilon(l)$, a negligible quantity in security parameter in l. Therefore, $Pr[u = u' | v = 0] = 1/2 + \epsilon(l)$. Similarly, simulator $\mathcal{SIM}$ guesses $v' = 0$ when $u = u'$, in turn, $Pr[v = v' | v = 0] = 1/2 + \epsilon(l)$.

The overall advantage of the simulator in DDH game is

$$\begin{aligned} &\frac{1}{2} Pr[v = v' | v = 1] + \frac{1}{2} Pr[v = v' | v = 0] - \frac{1}{2} \\ &= \frac{1}{2}\frac{1}{2} + \frac{1}{2}\left(\frac{1}{2} + \epsilon(l)\right) - \frac{1}{2} \\ &= \frac{\epsilon(l)}{2} \end{aligned}$$

Therefore, if the $\mathcal{A}$ has a non-negligible advantage $\epsilon(l)$ in the above game, then we can build a simulator $(\mathcal{SIM})$ which can break the DDH problem with non negligible quantity $\epsilon(l)/2$, which is an intractable problem. □

Theorem 3 *The proposed KASE is secure against cross-pairing attack.*

Proof An attacker $\mathcal{A}$ (the curious cloud server or an authorized user) may try to learn more information from the stored encrypted data beyond their authorization. With the knowledge of public parameters $PubK$=$(g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}) \in G$ and public information δ_l= (C_{l_1}, C_{l_2}) where $C_{l_1} = g^{t_l}$, $C_{l_2} = (v \cdot g_l)^{t_l}$; $\mathcal{A}$ may try to get information about message M by performing the cross-pairing attack on data ciphertext $C_{l_4} = M * e(g_1, g_n)^{t_l}$. The adversary $\mathcal{A}$ may try to compute the value of $e(g_1, g_n)^{t_l}$. However, the term $g_{n+1} = g^{\alpha^{n+1}}$ is missing in the public parameters $PubK$. Therefore, an attacker cannot compute the value of $e(C_1, g_{n+1})$ to get the value of $e(g_1, g_n)^{t_l}$. Notice that $\mathcal{A}$ cannot get the value of $e(g_1, g_n)^{t_l}$ by cross-paring the available public information $PubK$ and δ_l. Moreover, data-ciphertext C_M is secured with value of t_l, $\mathcal{A}$ is unable to get the value of t_l as per the discrete logarithm problem. In addition, using the keyword-ciphertext and public information an attacker cannot launch cross-paring attack to learn whether doc_l contains keyword value $w_{i,j}$ or not.

Furthermore, in the proposed KASE scheme, both the parameters of trapdoor $Tr = (Tr_0, Tr_1)$ are protected using same value $b \in_R Z_p$. Therefore, cross-paring of two different trapdoors Tr and Tr' is not possible, as the value of b is used different to generate each trapdoor. Cross-pairing of different trapdoors can generate incompatibility. Additionally, in the proposed scheme, the aggregate key contains single group element. Therefore, an adversary cannot extract the individual secret keys or keyword from the single aggregate key and trapdoor. Hence, different users cannot collide to decrypt or search over ciphertext(s) not in the scope of their respective aggregate keys. Even an authorized user

cannot try to generate new aggregate key or trapdoor from the known ones.

Therefore, the proposed KASE is secure against cross-pairing attack. □

7 Performance and Efficiency Analysis

In this section, we analyze and compare the theoretical space, computational and communicational complexities of the proposed R-OO-KASE scheme with other related KASE schemes [4, 13, 18–24, 31, 32, 34, 35]. Our observations from the theoretical analysis are as follows:

- *Storage Overhead*
  - The comparison of storage overhead is shown in Table 2. The size of intermediate ciphertext generated from the Offline_Encrypt() algorithm in the R-OO-KASE scheme depends on the number of keywords in the keywords space and the total number of documents belonging to a data owner. If we use a curve $E(F_{2^{271}}) \times E(F_{2^{271}}) \rightarrow F_{2^{271}}$ over the binary field $F_{2^{271}}$, which is equivalent to the 80-bit security level [11]. The size of an element in group G is 542 bits using an elliptic curve with 252 bits p. The size of an element in group G can be reduced to 34 bytes using a compression technique proposed in the paper [26]. Therefore, if there are 10000 keywords in the system and data owner owns n =1000 documents, then also intermediate ciphertext requires less than $0.24GB$ storage space. Further, the size of final ciphertext generated after Online_Encrypt() remains constant in our scheme, whereas in the existing KASE schemes [4, 13, 18–22, 24, 31, 32, 34, 35] the size of ciphertext is linear in the number of keywords to be attached with the ciphertext.
  - In the existing KASE schemes [4, 13, 18–22, 24, 31, 32, 34, 35], the storage overhead of trapdoor is linear in the number of keywords in the search query. On the other hand, the storage overhead of trapdoor in the R-OO-KASE is constant and independent of the number of keywords in the search query set.
  - The size of the revocation list in the proposed scheme is constant, i.e., two group elements, and it is independent of the number of revoked users. The storage complexity of each individual revocation list RL_l (i.e., revocation list for lth document) is $O(1)$. However, the proposed approach generates a revocation list document-wise, so there will be a total n revocation list in the system if the data owner held n documents. The total storage cost of all the revocation list in the system is linear with the number of documents that a data owner holds. Even though the proposed revocation solution brought linear storage cost of $O(n)$ by generating revocation list document-wise, the idea of direct revocation used in the proposed approaches eliminates the expensive cost of updated key distribution among all the non-revoked users, as the revocation is realized by publishing the revocation list. The revocation in the proposed scheme does not affect the non-revoked users, as they do not require updating their corresponding delegated keys, which greatly reduces the expensive cost of key update and the overhead of key delegate authority. Especially, to realize search right revocation, re-encryption operations on keyword ciphertexts are not needed in our scheme. The KASE [35] scheme supports the revocation of delegated rights by distributing the new set of keys to the non-revoked users in each time period. The revocable KASE scheme proposed in [35] requires the data owner to maintain a user list to generate authorized keys for all the non-revoked users. The authorized users registered in the data owner's list can receive a new pair of authorized keys from the data owner. Therefore, the existing revocable KASE scheme [35] consumes more overhead at the data owner side to

Table 2 Comparative analysis: storage overhead

Schemes	Size of Ciphertext		Size of Trapdoor	Size of RL
	Online	Offline		
[4]	$O(\lvert\overline{KW}\rvert)$	NA	$O(\lvert\overline{Q}\rvert)$	NA
[19]	$O(\lvert\overline{KW}\rvert)$	NA	$O(\lvert\overline{Q}\rvert)$	NA
[18]	$O(\lvert\overline{KW}\rvert)$	NA	$O(\lvert\overline{Q}\rvert)$	NA
[34]	$O(\lvert\overline{KW}\rvert)$	NA	$O(\lvert\overline{Q}\rvert)$	NA
[22]	$O(\lvert\overline{KW}\rvert)$	NA	$O(\lvert\overline{Q}\rvert)$	NA
[35]	$O(\lvert\overline{KW}\rvert)$	NA	$O(\lvert\overline{Q}\rvert)$	NA
[20]	$O(\lvert\overline{KW}\rvert)$	NA	$O(\lvert\overline{Q}\rvert)$	NA
[21]	$O(\lvert\overline{KW}\rvert)$	NA	$O(\lvert\overline{Q}\rvert)$	NA
[24]	$O(\lvert\overline{KW}\rvert)$	NA	$O(\lvert\overline{Q}\rvert)$	NA
[23]	$O(1)$	NA	$O(1)$	NA
[32]	$O(\lvert\overline{KW}\rvert)$	NA	$O(\lvert\overline{Q}\rvert)$	NA
[13]	$O(\lvert\overline{KW}\rvert)$	NA	$O(\lvert\overline{Q}\rvert)$	NA
[31]	$O(\lvert\overline{KW}\rvert)$	NA	$O(\lvert\overline{Q}\rvert)$	NA
R-OO-KASE	$O(1)$	$O(nm)$	$O(1)$	$O(1)$

$\lvert\overline{KW}\rvert$ number of keywords attached with the ciphertext, $\lvert\overline{Q}\rvert$ number of keywords in the query set, n number of documents held by a data owner, m size of keywords space in the system, NA not applicable

generate as well as distribute keys periodically to each authorized user of the system.

- *Computation Overhead*
  - The comparison of computation overhead is shown in Table 3. The computational cost of revocation in the proposed scheme is linear in the number of revoked users, whereas verification of the user's authorization with the revocation list only requires two pairing operations.
  - In the proposed R-OO-KASE scheme, Offline_Encrypt() algorithm requires only one pairing operation along with the exponentiation operations proportional to the number of keywords in the system. Additionally, Online_Encrypt() algorithm in the proposed scheme requires only multiplication operations linearly dependent on the number of keywords to be attached with the ciphertext. The existing KASE methods [4, 13, 18–21, 31] use two pairing operations to generate a keyword ciphertext for each keyword to be attached with the document, whereas in the proposed scheme, we are using exponentiation to generate the keyword ciphertext. The authors of [25] shows that for different security levels (i.e., for 80 bits, 112 bits and 128 bits) and prime order groups, the group exponentiation (of a random group element with a random exponent) requires significantly less time compared to the pairing operations (on random group elements). We remark that the work done in the offline phase in the R-OO-KASE scheme is roughly equivalent to the work of the regular encryption algorithm in the existing KASE schemes [4, 13, 18–22, 24, 31, 32, 34, 35]. Therefore, the proposed R-OO-KASE scheme reduces the online computation cost of the existing KASE methods [4, 13, 18–22, 24, 31, 32, 34, 35] substantially.
  - The computational cost of trapdoor generation in case of multi-keyword search is same for all the considered KASE schemes [4, 13, 18–24, 31, 32, 34, 35] including the proposed KASE scheme. However, in case of multi-keyword search, the number of resultant trapdoors are $|\overrightarrow{Q}|$ in the existing KASE schemes [4, 13, 18–22, 24, 31, 32, 34, 35], whereas in the proposed scheme, one can perform a multi-keyword search using a single trapdoor of constant size. We are not focusing on the query expressiveness, but our aim is to support multiple keyword searches using a single trapdoor. The proposed scheme supports only the case in which $\overrightarrow{Q} = \overrightarrow{KW}$, i.e, the set of queried keywords are the same as the set of keywords encrypted. Moreover, the proposed approach provides trapdoor privacy in contrast to the existing KASE schemes [4, 18–21] that leak data related to keyword from the trapdoor as discussed in [14].
  - The computational cost of Test() algorithm in the existing schemes [4, 13, 18–22, 24, 31, 32, 34, 35] scales linearly with the number of keywords in the search query set. In contrast, in the proposed KASE scheme, the computational cost of the Test() algorithm is constant and independent of the number of keywords in the query set. Moreover, in the existing KASE schemes [4, 13, 18–21, 31, 34, 35], when an authorized user submits the trapdoor to the cloud server in order to perform search; the total time required to search a keyword at the cloud server is: Time required to transform aggregate trapdoor Tr to generate actual trapdoor Tr_l for searching over lth document + Time required to execute Test () algorithm. This trapdoor transformation process used in the existing KASE schemes [4, 13, 18–21, 31, 34, 35] adds extra computational and storage overhead at the server-side. The proposed R-OO-KASE scheme allows searching over shared documents using aggregate trapdoor without requiring to generate individual trapdoor Tr_l from the aggregate trapdoor. In this way, we overcome the extra overhead of trapdoor transformation required on the server-side. With such constant computational overhead of Test() algorithm, the proposed scheme performs better than the existing KASE schemes [4, 13, 18–21, 31, 34, 35].
  - In the proposed KASE scheme, the Online_Decrypt() only requires three pairing operations and most of the costly computation of the decryption work is shifted to the offline phase. In this way, we reduce the computation overhead at both data owners and users' sides.
- *Communication Overhead*
  - We compare the communication overhead of ciphertext and query trapdoor as shown in Table 4.
  - The communication overhead of query trapdoor in the existing KASE [4, 13, 18–22, 24, 31, 32, 34, 35] schemes is linear in the number of keywords in the query set. The communication overhead of ciphertext in the existing KASE schemes [4, 13, 18–22, 24, 31, 32, 34, 35] is linear in the number of keywords attached with the ciphertext.
  - The proposed scheme reduces communication cost as compared to the existing KASE schemes [4, 13, 18–22, 24, 31, 32, 34, 35]. The communication cost of the trapdoor in the R-OO-KASE scheme is independent of the number of keywords in the query set. Similarly, the communication cost of ciphertext in

Table 3 Comparative analysis: computation overhead

Schemes	Revocation	Encryption() **Data + Keyword** ciphertext		TrapdoorGen()	Test()	Decryption()	
		Offline	Online	Multi-keyword search	Multi-keyword search	Offline	Online
[4]	NA	NA	$O(\lvert\overrightarrow{KW}\rvert P)$	$O(\lvert\vec{Q}\rvert MUL)$	$O(\lvert\vec{Q}\rvert P)$	NA	NA
[19]	NA	NA	$O(\lvert\overrightarrow{KW}\rvert P)$	$O(\lvert\vec{Q}\rvert MUL)$	$O(\lvert\vec{Q}\rvert P)$	NA	NA
[18]	NA	NA	$O(\lvert\overrightarrow{KW}\rvert P)$	$O(\lvert\vec{Q}\rvert MUL)$	$O(\lvert\vec{Q}\rvert P)$	NA	NA
[34]	NA	NA	$O(\lvert\overrightarrow{KW}\rvert E)$	$O(\lvert\vec{Q}\rvert MUL)$	$O(\lvert\vec{Q}\rvert P)$	NA	NA
[22]	NA	NA	$O(\lvert\overrightarrow{KW}\rvert E)$	$O(\lvert\vec{Q}\rvert MUL)$	$O(\lvert\vec{Q}\rvert P)$	NA	NA
[35]	NA	NA	$O(\lvert\overrightarrow{KW}\rvert E)$	$O(\lvert\vec{Q}\rvert MUL)$	$O(\lvert\vec{Q}\rvert P)$	NA	NA
[20]	NA	NA	$O(\lvert\overrightarrow{KW}\rvert P)$	$O(\lvert\vec{Q}\rvert MUL)$	$O(\lvert\vec{Q}\rvert P)$	NA	NA
[21]	NA	NA	$O(\lvert\overrightarrow{KW}\rvert P)$	$O(\lvert\vec{Q}\rvert MUL)$	$O(\lvert\vec{Q}\rvert P)$	NA	NA
[24]	NA	NA	$O(\lvert\overrightarrow{KW}\rvert E)$	$O(\lvert\vec{Q}\rvert MUL)$	$O(\lvert\vec{Q}\rvert P)$	NA	NA
[23]	NA	NA	$O(\lvert\overrightarrow{KW}\rvert E)$	$O(\lvert\vec{Q}\rvert MUL)$	$O(3P)$	NA	NA
[32]	NA	NA	$O(\lvert\overrightarrow{KW}\rvert n^2 logn)$	$O(\lvert\vec{Q}\rvert)$	$O(\lvert\vec{Q}\rvert nlogn)$	NA	NA
[13]	NA	NA	$O(\lvert\overrightarrow{KW}\rvert P)$	$O(\lvert\vec{Q}\rvert MUL)$	$O(\lvert\vec{Q}\rvert P)$	NA	NA
[31]	NA	NA	$O(\lvert\overrightarrow{KW}\rvert P)$	$O(\lvert\vec{Q}\rvert MUL)$	$O(\lvert\vec{Q}\rvert P)$	NA	NA
R-OO-KASE	$O(\lvert\mathcal{U}\rvert MUL)$	$O(mnE)$	$O(\lvert\overrightarrow{KW}\rvert MUL)$	$O(\lvert\vec{Q}\rvert MUL)$	$O(3P)$	$O(\lvert S\rvert MUL)$	$O(3P)$

$\lvert\overrightarrow{KW}\rvert$ number of keywords attached with the ciphertext, $\lvert\vec{Q}\rvert$ number of keywords in the query set, n number of documents held by a data owner, m size of keywords space in the system, $\lvert\mathcal{U}\rvert$ number of revoked users, E exponentiation, MUL scalar multiplication, P pairing, $\lvert S\rvert$ size of set S, NA not applicable

Table 4 Comparative analysis: communication overhead

Schemes	Ciphertext	Search request (No. of Trapdoor)
[4]	$2\lvert G\rvert + \lvert\overrightarrow{KW}\rvert\lvert G_T\rvert$	$\lvert\vec{Q}\rvert\lvert G\rvert$
[19]	$2\lvert G\rvert + \lvert\overrightarrow{KW}\rvert\lvert G_T\rvert$	$\lvert\vec{Q}\rvert\lvert G\rvert$
[18]	$2\lvert G\rvert + \lvert\overrightarrow{KW}\rvert\lvert G_T\rvert$	$\lvert\vec{Q}\rvert\lvert G\rvert$
[34]	$(2 + \lvert\overrightarrow{KW}\rvert)\lvert G\rvert + \lvert G_T\rvert$	$\lvert\vec{Q}\rvert\lvert G\rvert$
[22]	$(2 + \lvert\overrightarrow{KW}\rvert)\lvert G\rvert + \lvert G_T\rvert$	$\lvert\vec{Q}\rvert\lvert G\rvert$
[35]	$(2 + \lvert\overrightarrow{KW}\rvert)\lvert G\rvert + \lvert G_T\rvert$	$\lvert\vec{Q}\rvert\lvert G\rvert$
[20]	$2\lvert G\rvert + \lvert\overrightarrow{KW}\rvert\lvert G_T\rvert$	$\lvert\vec{Q}\rvert\lvert G\rvert$
[21]	$2\lvert G\rvert + \lvert\overrightarrow{KW}\rvert\lvert G_T\rvert$	$\lvert\vec{Q}\rvert\lvert G\rvert$
[24]	$3\lvert G\rvert + \lvert\overrightarrow{KW}\rvert\lvert G_T\rvert$	$\lvert\vec{Q}\rvert\lvert G\rvert$
[23]	$4\lvert G\rvert + 2\lvert G_T\rvert$	$\lvert G\rvert$
[32]	$\lvert\overrightarrow{KW}\rvert((2m+1)log_q)$	$\lvert\vec{Q}\rvert(\tau'\sqrt{m})$
[13]	$3\lvert G\rvert + \lvert\overrightarrow{KW}\rvert\lvert G_T\rvert$	$\lvert\vec{Q}\rvert(\lvert G\rvert + 2\lvert Z_p^*\rvert)$
[31]	$2\lvert G\rvert + \lvert\overrightarrow{KW}\rvert\lvert G_T\rvert$	$\lvert\vec{Q}\rvert\lvert G\rvert$
R-OO-KASE	$3\lvert G\rvert + \lvert G_T\rvert$	$\lvert G\rvert$

n number of documents that belongs to the data owner, $\lvert\overrightarrow{KW}\rvert$ number of keywords attached with the ciphertext, $\lvert\vec{Q}\rvert$ number of keywords in the query set, q size prime number in lattice-based KASE scheme [32], mXn size of matrix in lattice-based KASE scheme [32], $\lvert G\rvert$ size of a element in group G, $\lvert G_T\rvert$ size of a element in group G_T

the R-OO-KASE scheme is independent of the number of keywords attached with the ciphertext.

- If we use 80-bit [6] security level, then the size of an element in group G is 542 bits using an elliptic curve with 252 bits p. The size of an element in group G can be reduced to 34 bytes using a compression technique proposed in the paper [26]. Therefore, in the proposed scheme, the size of ciphertext is $3\lvert G\rvert + \lvert G_T\rvert = 3*34 + 34bytes = 136bytes$ and trapdoor size is $\lvert G\rvert = 34\ bytes$. The MICA2 equipped with an ATmega128 8-bit processor clocked at 7.3728MHz, 4-KB RAM, and 128-KB ROM requires 3*27*8/12400 = 0.052 mJ to transmit one-byte message, as discuss in [27]. Therefore, communication energy consumptions of the sensor in the R-OO-KASE scheme to send ciphertext are 0.052*136 =7.072 mJ and trapdoor is 0.052*34=1.768 mJ.

7.1 Evaluation of KASE Schemes

In this section, we compare the proposed R-OO-KASE scheme with the existing KASE schemes [4, 13, 18–24, 31, 32, 34, 35] in the security, efficiency and functional respects. The comparison results are summarized in Table 5.

Table 5 A Comparative summary of the existing KASE schemes with the R-OO-KASE

Scheme	EFF	MUL	REV	MO	TT	SCA	IND-KGA	IND-CKA	CPrA	SRV
[4]	×	×	×	×	✓	×	×	×	×	×
[19]	×	×	×	✓	✓	×	×	×	×	✓
[18]	×	×	×	✓	✓	×	×	×	×	✓
[34]	×	×	×	×	✓	×	✓	✓	✓	×
[22]	×	×	×	✓	×	✓	✓	✓	✓	×
[35]	×	×	✓	×	✓	×	✓	✓	✓	×
[20]	×	×	×	✓	✓	×	×	×	×	✓
[21]	×	×	×	✓	✓	×	×	×	×	✓
[24]	×	✓	×	✓	×	✓	✓	✓	✓	×
[23]	×	✓	✓	×	×	✓	✓	✓	✓	×
[32]	×	×	×	×	✓	×	✓	✓	×	×
[13]	×	×	×	×	✓	×	✓	✓	×	×
[31]	×	×	×	×	✓	×	×	×	×	✓
R-OO-KASE	✓	✓	✓	×	×	✓	✓	✓	✓	×

EFF efficient for resource starved environment, MUL support multi-keyword search using a trapdoor, REV support revocation of delegated rights, MO search over multi-owners' data using a single query trapdoor, TT is trapdoor transformation required?, SCA scalable, IND-KGA secure against keyword guessing attack, IND-CKA secure against keyword indistinguishability attack, CPrA secure against cross-pairing attack, SRV support verification of search result using an aggregate key

In Table 5, each scheme is compared in terms of the following parameters:

- Efficiency for resource starved environment
- Support for multi-keyword search using a single trapdoor
- Support for revocation of delegated rights
- Support for search over multi-owners' data using a single trapdoor
- Requirement of trapdoor transformation before keyword search
- Scalability
- Security against IND-CKA and IND-KGA
- Security against cross-paring attack
- Support for verification of search result

From Table 5, we can observe that none of the existing KASE schemes listed in the table provide revocation of delegated rights, conjunctive keywords search using a single trapdoor and energy-efficient encryption as well as decryption algorithms for resource-limited devices. Additionally, as shown in Table 5, the KASE schemes [4, 13, 23, 24, 31, 32, 34, 35] do not support search over multi-owners' data using a single trapdoor. On the other hand, the KASE schemes proposed in [18–22] provide the solution to search over multi-owners' data using a single trapdoor. The KASE schemes proposed in [18–21, 31] allow verification of search result using an aggregate key. However, the existing KASE schemes [4, 13, 18–21, 31, 32, 34, 35] require trapdoor transformation at the time of keyword searching that adds an additional computational and storage overhead at the server-side. Further, the KASE schemes [4, 18–21, 31] do not provide formal security proof against IND-CKA and IND-KGA. The existing KASE schemes [4, 18–21, 31] are insecure against cross-pairing attack. The existing KASE schemes [4, 13, 18–21, 31, 32, 34, 35] are not scalable, i.e., if the number of files exceeds n (maximum number of documents held by a data owner), the whole system should be reestablished. The KASE [35] scheme supports revocation of delegated rights by distributing the new set of keys to the non-revoked users in each time period. However, the existing revocable KASE scheme [35] consumes more overhead at the data owner side to generate as well as distribute keys periodically to each authorized user of the system. Further, considering the requirements of resource-limited environment, none of the existing KASE[4, 13, 18–24, 31, 32, 34, 35] schemes can be applicable to work on resource-limited devices, with the battery as the only source of power.

In contrast, the proposed KASE scheme supports multi-keyword searches using a single trapdoor. The proposed R-OO-KASE scheme allows the data owner to revoke delegated rights. Additionally, the proposed scheme reduces the computation overhead of encryption and decryption, to make the scheme practically applicable in the resource-limited environment. Furthermore, the proposed KASE scheme does not require trapdoor transformation and reduces overhead at the server side at the time of keyword searching. The proposed approach is also scalable and the aggregate key is independent of the maximum number of documents held by a data owner. The proposed R-OO-KASE scheme provides security against the IND-KGA, IND-CKA and cross-pairing attack.

8 Empirical Evaluation

8.1 Implementation Details

We implement a prototype of the proposed R-OO-KASE scheme and conduct the experiments on two different platforms: the computer and the mobile device, using Java Pairing-Based Cryptographic (JPBC) library [7]. The experiments are conducted 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. In order to evaluate computational cost on the resource-constrained device, we simulate client implementations on Xiaomi Redmi 3S mobile device with Android OS 6. For evaluation, the official Medicare.gov Hospital Compare datasets provided by the Centers for Medicare and Medicaid Services [12] are used. Type A pairing is used for the evaluation, and it is the fastest (symmetric) pairing among all types of curves. The Type A pairings are constructed on the curve $y^2 = x^3 + x$ over the field F_q for some prime $q = 3 \bmod 4$.

In the proposed scheme, the Setup(), KeyGen(), Encrypt(), Extract(), Revocation(), TrapdoorGen() algorithms are executed at client-side and the Test() algorithm on the server-side. The running time is measured in milliseconds. The results represent an average of five executions.

8.2 Experimental Results

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

The time cost of Setup() is linear in the maximum number of documents belonging to the data owner (Fig. 4a). Figure 4b shows the computational cost of KeyGen() approximate to a constant. The results given in Fig. 4c show that the computational overhead of Online_Encrypt() is linear in the number of keywords to be attached with the ciphertext. The results are given in Fig. 4d, clearly indicate that when the number of keywords is greater than 500, only 10% of the work remains to be done in online encryption phase and more than 90% of the work in the keyword and message encryption is shifted to the offline phase. The computational cost of Revocation() is linear in the number of revoked users, as shown in Fig. 4e. Figure 4f shows that the computational time of Extract() is linear in the number of shared documents. Figure 4g shows the computational time of TrapdoorGen() is linear in the number of keywords within the search query set. The execution time of Test() is linear in the number of shared documents (Fig. 4i), whereas in the case of multi-keyword search, the computational time of Test() remains constant with respect to the number of keywords in the query set (Fig. 4h). Figure 4j, k shows that in the online decryption phase only 3 pairing operations are required to perform and more than 80% of the decryption work is shifted to the offline phase. This observation from empirical analysis implies that we reduce the computation overhead at both data owners and users' sides.

8.3 Comparative Analysis Results

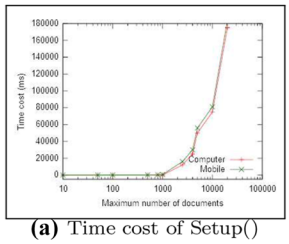
In this section, to show the trade-off in performance of the proposed R-OO-KASE scheme, the proposed solution is evaluated against the existing KASE[20, 22] schemes.

In the proposed scheme as well as in the existing KASE schemes[20, 22], the Setup(), KeyGen(), Encrypt(), Extract(), Revocation(), TrapdoorGen() algorithms are executed at client-side and the Test() algorithm on the server-side. To calculate the computation cost of the proposed Test() algorithm for the multi-keyword query, the conjunctive search queries are given. The total time required by the proposed Test() algorithm to get the results after he sends out the query with multiple keywords is measured. The running time is measured in milliseconds. The results represent an average of five executions.

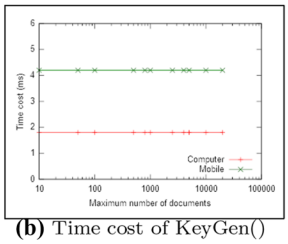
Our observations from the empirical analysis are as follows:

- Time cost of Setup() for the proposed R-OO-KASE is lesser as compared to the existing KASE [20, 22] schemes and it is in linear with the maximum number of documents belonging to the data owner (Fig. 4l).
- The results given in Fig. 4m show that the computational overhead of Encrypt() is in linear with the number of keywords attached with the ciphertext. However, the encryption cost of the proposed R-OO-KASE and the MULKASE [22] scheme is lesser as compared to the MO-VKASE[20] scheme, since the cost of pairing is much more heavy than the multiplication and exponentiation operations. The MO-VKASE uses two pairing operations to generate a keyword ciphertext, whereas, the MULKASE uses multiplication and exponentiation to generate a keyword ciphertext. The proposed Encrypt() algorithm uses one pairing operation to generate a ciphertext.
- Figure 4n shows that the computational time of Extract() is in linear with the number of shared documents for the proposed R-OO-KASE and the existing KASE schemes [20, 22]. The computational time of Extract() is same for all the considered KASE schemes.

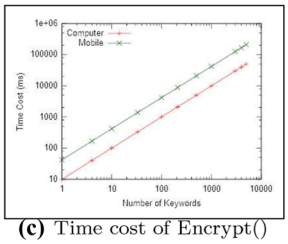
Fig. 4 The computational cost of R-OO-KASE algorithms()



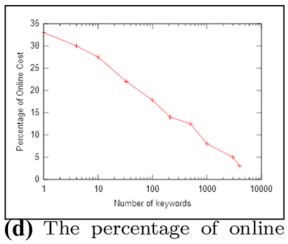
(a) Time cost of Setup()



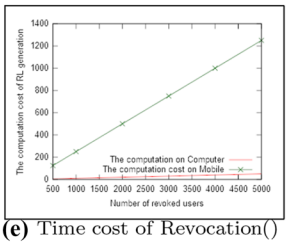
(b) Time cost of KeyGen()



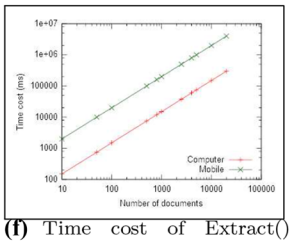
(c) Time cost of Encrypt()



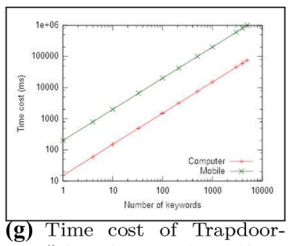
(d) The percentage of online computational overhead of Encrypt()



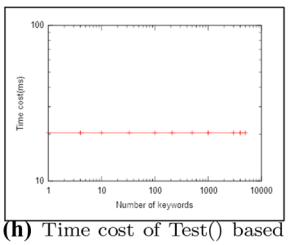
(e) Time cost of Revocation()



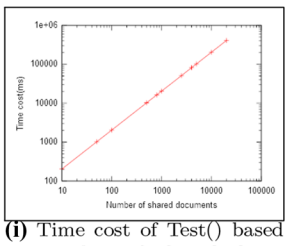
(f) Time cost of Extract() based on number of shared documents



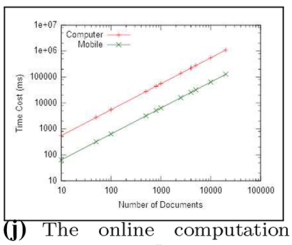
(g) Time cost of TrapdoorGen() based on number of keywords



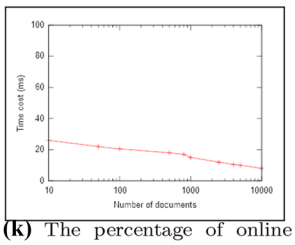
(h) Time cost of Test() based on number of keywords



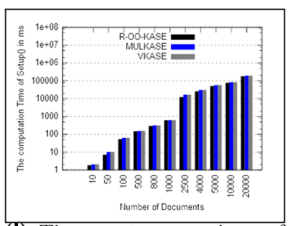
(i) Time cost of Test() based on number of shared documents



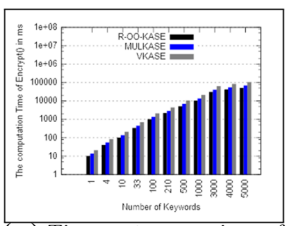
(j) The online computation time of Decrypt()



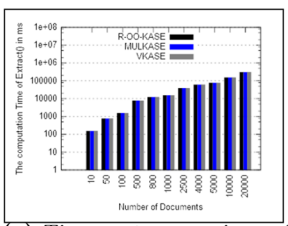
(k) The percentage of online computational overhead of Decrypt()



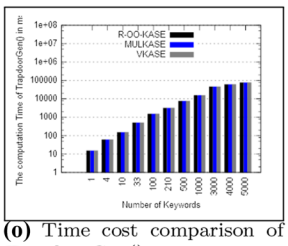
(l) Time cost comparison of Setup()



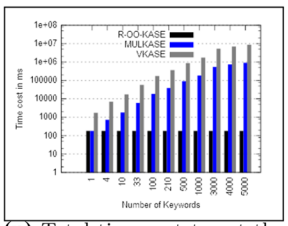
(m) Time cost comparison of Encrypt()



(n) Time cost comparison of Extract()



(o) Time cost comparison of TrapdoorGen()



(p) Total time cost to get the search results for query with multiple keywords

- Figure 4o shows that the computational time of TrapdoorGen() is linear with the number of keywords in the search query set for the proposed R-OO-KASE and the existing KASE [20, 22] schemes. However, the MUL-KASE [22] and the MO-VKASE [20] only support the exact keyword match query on a single keyword.
- Figure 4p shows the comparison result of the time cost required to send and execute a search query with multiple keywords. The total time considered for comparison is: Communication cost required to send trapdoor(s) for a query with multiple keywords + Computation cost required to execute Test() for multi-keyword query + Communication cost to receive search results. The dataset size considered for this comparison is 10, i.e., 10 documents are in the range of given query trapdoor. Therefore, the results shown in the comparison graph are the total time after executing Test() for multi-keyword query over 10 documents. Additionally, the time cost for the existing VKASE [20] scheme also includes computation cost of Adjust() algorithm, as VKASE [20] scheme requires trapdoor transformation by executing Adjust() before running Test(). The proposed R-OO-KASE and the existing MULKASE [22] schemes do not use Adjust() algorithm for trapdoor transformation before searching. Therefore, R-OO-KASE and the existing MULKASE [22] schemes take lesser computation time for Test() as compared to the VKASE [20]. Further, the existing KASE[20, 22] schemes do not support multi-keyword searches using a single trapdoor. Therefore, the total time cost required to send and execute search query with multiple keywords is less in the proposed R-OO-KASE scheme as compared to the existing KASE [20, 22] schemes. In the existing KASE [4, 18–22, 34, 35] schemes, the user requires to submit different trapdoors for each individual keyword to the server. The server performs a search for each of the keywords separately and returns the intersection of all the results. Specifically, for searching a set of keywords $\vec{Q} = \{Q_1, \ldots, Q_p\}$ over shared dataset, the existing KASE schemes [4, 18–22, 34, 35] require $|\vec{Q}| = p$ number of trapdoors. Similarly, the cloud server requires to perform $O(|\vec{Q}|)$ pairing operations to execute the multi-keyword query. Additionally, the cloud server must perform a search query over all the shared dataset. In the considered scenario, as the dataset is 10 the cloud server requires to perform $O(|\vec{Q}|)$ pairing operations over all the 10 documents to execute the multi-keyword query. Therefore, the time cost increases drastically in the existing KASE [20, 22] schemes as compared to the proposed approach. In the proposed R-O-KASE scheme, the communication and computation cost required to execute multi-keyword query is independent of the size of keywords in the query set.

8.4 Findings

- The encryption cost of the proposed R-OO-KASE scheme is lesser as compared to the MO-VKASE [20] scheme. The reason for the lesser computation cost of the R-OO-KASE scheme is that the proposed Encrypt() algorithm mainly uses the multiplication and exponentiation operations whereas the MO-VKASE [20] uses pairing operations in Encrypt() algorithm (Fig. 4m).
- The existing MO-VKASE [20] scheme takes more computation time for Test() as compared to the proposed R-OO-KASE for the exact keyword match query on non-numeric keywords. The reason for the same is the MO-VKASE [20] uses Adjust() algorithm for trapdoor transformation before running Test() algorithm and it increases search time. On the other hand, the proposed KASE scheme does not use the Adjust() algorithm. The proposed R-OO-KASE scheme allows searching over the shared dataset S using a single trapdoor Tr that a query requester submits to the cloud server (Fig. 4p).
- The computation cost of the proposed Test() is independent of the number of keywords in the search query, however, the computation cost of Test() is linear with the number of shared documents (Fig. 4h, i).

9 Conclusions and Future Extensions

In this paper, we propose the R-OO-KASE (Revocable Online/Offline KASE) scheme that is suitable for the resource-constrained environment, as we split costly operations of encryption and decryption into two phases: online and offline. In the offline phase, the user performs expensive pairing and exponentiation operations required in the encryption/decryption. In the online phase, i.e., when the device is moving on (not connected to the power source), the user can generate the final output with the minimal computational cost. We provide the performance estimates that showed over 90% of the computation operations of encryption/decryption are shifted to the offline phase. The proposed KASE scheme also supports the revocation of delegated rights in the cloud environment, realizing the key aggregation and user access control effectively. The proposed scheme supports fine-grained revocation of the delegated rights on document level, instead of coarse-grained all-or-nothing access. The idea of direct revocation used in the proposed approach eliminates the expensive cost of updated key distribution among all the non-revoked users, as the revocation is realized by publishing the revocation list. The other contribution of the proposed approach is that it offers multi-keyword searches over a shared dataset using a constant size query trapdoor. We also proved that the proposed scheme is secure against defined IND-CKA, IND-KGA and

cross-pairing attacks. In addition, the empirical analysis confirms that the proposed scheme is suitable for resource-constrained devices and improves query performance as compared to the existing KASE schemes. Overall, the proposed R-OO-KASE scheme helps to reduce the cost of bringing KASE into practice on the power-constrained device.

In the future, one can extend the scheme and enhance the system usability by providing fuzzy, semantic and ranked search on the encrypted dataset using a single trapdoor. Research on query expressiveness needs to move toward closing the gap between existing SE schemes and plaintext searches.

Funding Funding information is not applicable. No funding was received. The experiments and research comply with the current laws of the country.

References

1. Boneh D, Gentry C, Waters B (2005) Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Annual international cryptology conference, Springer, pp 258–275
2. Chang YC, Mitzenmacher M (2005) Privacy preserving keyword searches on remote encrypted data. In: ACNS, Springer, vol 5, pp 442–455
3. Chu CK, Chow SS, Tzeng WG, Zhou J, Deng RH (2014) Key-aggregate cryptosystem for scalable data sharing in cloud storage. IEEE Trans Parallel Distrib Syst 25(2):468–477
4. Cui B, Liu Z, Wang L (2016) Key-aggregate searchable encryption (kase) for group data sharing via cloud storage. IEEE Trans Comput 65(8):2374–2385
5. Curtmola R, Garay J, Kamara S, Ostrovsky R (2011) Searchable symmetric encryption: improved definitions and efficient constructions. J Comput Secur 19(5):895–934
6. Daemen J, Rijmen V (2013) The design of Rijndael: AES-the advanced encryption standard. Springer, Berlin
7. De Caro A, Iovino V (2011) jpbc: Java pairing based cryptography. In: 2011 IEEE symposium on computers and communications (ISCC), IEEE, pp 850–855
8. Gentry C, Halevi S (2011) Implementing gentry's fully-homomorphic encryption scheme. In: Annual international conference on the theory and applications of cryptographic techniques, Springer, pp 129–148
9. Goh EJ et al (2003) Secure indexes. IACR Cryptol ePrint Arch 2003:216
10. Green M, Hohenberger S, Waters B, et al (2011) Outsourcing the decryption of abe ciphertexts. In: USENIX security symposium, vol 2011
11. Gura N, Patel A, Wander A, Eberle H (2004) Comparing elliptic curve cryptography and rsa on 8-bit cpus. In: International workshop on cryptographic hardware and embedded systems, Springer, pp 119–132
12. Hospital compare (2019) https://data.medicare.gov/data/hospital-compare. Accessed 13 Feb 2019
13. Kamimura M, Yanai N, Okamura S, Cruz JP (2019) Key-aggregate searchable encryption, revisited: formal foundations for cloud applications, and their implementation. Preprint arXiv :1908.11096
14. Kiayias A, Oksuz O, Russell A, Tang Q, Wang B (2016) Efficient encrypted keyword search for multi-user data sharing. In: European symposium on research in computer security, Springer, pp 173–195
15. Lai J, Deng RH, Guan C, Weng J (2013) Attribute-based encryption with verifiable outsourced decryption. IEEE Trans Inf For Secur 8(8):1343–1354
16. Li J, Chen X, Li J, Jia C, Ma J, Lou W (2013) Fine-grained access control system based on outsourced attribute-based encryption. In: European symposium on research in computer security, Springer, pp 592–609
17. Li J, Huang X, Li J, Chen X, Xiang Y (2014) Securely outsourcing attribute-based encryption with checkability. IEEE Trans Parallel Distrib Syst 25(8):2201–2210
18. Li T, Liu Z, Jia C, Fu Z, Li J (2018) Key-aggregate searchable encryption under multi-owner setting for group data sharing in the cloud. Int J Web Grid Serv 14(1):21–43
19. Li T, Liu Z, Li P, Jia C, Jiang ZL, Li J (2016) Verifiable searchable encryption with aggregate keys for data sharing in outsourcing storage. In: Information security and privacy, Springer International Publishing, pp 153–169
20. Liu Z, Li T, Li P, Jia C, Li J (2018) Verifiable searchable encryption with aggregate keys for data sharing system. Future Gen Comput Syst 78:778–788
21. Liu Z, Liu Y (2018) Verifiable and authenticated searchable encryption scheme with aggregate key in cloud storage. In: 2018 14th international conference on computational intelligence and security (CIS), IEEE, pp 421–425
22. MuktiPadhya DCJ (2018) Mulkase—a novel approach for key aggregate searchable encryption for multi-owner data. Front Inf Technol Electron Eng. https://doi.org/10.1631/FITEE.1800192
23. Padhya M, Jinwala DC (2019) BTG-RKASE: privacy preserving revocable key aggregate searchable encryption with fine-grained multi-delegation & break-the-glass access control. In: Proceedings of the 16th international joint conference on e-business and telecommunications, ICETE 2019, Volume 2: SECRYPT, Prague, Czech Republic, July 26–28, 2019, pp 109–124. https://doi.org/10.5220/0007919901090124
24. Padhya M, Jinwala DC (2020) CRSQ-KASE: key aggregate searchable encryption supporting conjunctive range and sort query on multi-owner encrypted data. Arab J Sci Eng. https://doi.org/10.1007/s13369-019-04302-x
25. Rouselakis Y, Waters B (2015) Efficient statically-secure large-universe multi-authority attribute-based encryption. In: International conference on financial cryptography and data security, Springer, pp 315–332
26. Shim KA (2012) Cpas: an efficient conditional privacy-preserving authentication scheme for vehicular sensor networks. IEEE Trans Veh Technol 61(4):1874–1883
27. Shim KA, Lee YR, Park CM (2013) Eibas: an efficient identity-based broadcast authentication scheme in wireless sensor networks. Ad Hoc Netw 11(1):182–189

28. Song DX, Wagner D, Perrig A (2000) Practical techniques for searches on encrypted data. In: Proceedings. 2000 IEEE symposium on security and privacy, 2000. S&P 2000, IEEE, pp 44–55
29. Sun W, Yu S, Lou W, Hou YT, Li H (2014) Protecting your right: attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud. In: 2014 Proceedings IEEE on INFOCOM, IEEE, pp 226–234
30. Wang C, Li W, Li Y, Xu X (2013) A ciphertext-policy attribute-based encryption scheme supporting keyword search function. In: Cyberspace safety and security, Springer, pp 377–386
31. Wang X, Cheng X (2019) Efficient verifiable key-aggregate keyword searchable encryption for data sharing in outsourcing storage. IEEE Access
32. Yao Y, Zhai Z, Liu J, Li Z (2019) Lattice-based key-aggregate (searchable) encryption in cloud storage. IEEE Access 7:164544–164555
33. Zhang L, Ahn GJ, Chu BT (2003) A rule-based framework for role-based delegation and revocation. ACM Trans Inf Syst Secur (TISSEC) 6(3):404–441
34. Zhou R, Zhang X, Du X, Wang X, Yang G, Guizani M (2018) File-centric multi-key aggregate keyword searchable encryption for industrial internet of things. IEEE Trans Ind Inform 14:3648–3658
35. Zhou R, Zhang X, Wang X, Yang G, Li W (2018) Keyword searchable encryption with fine-grained forward secrecy for internet of thing data. In: International conference on algorithms and architectures for parallel processing, Springer, pp 288–302
36. Zhou Y, Xu G, Wang Y, Wang X (2016) Chaotic map-based time-aware multi-keyword search scheme with designated server. Wirel Commun Mobile Comput 16(13):1851–1858
37. Zhou Z, Huang D (2012) Efficient and secure data storage operations for mobile cloud computing. In: Proceedings of the 8th international conference on network and service management, International Federation for Information Processing, pp 37–45