International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6964
***Article Received**: 28 March 2023 **Revised**: 18 May 2023 **Accepted**: 01 June 2023*

Optimal and Efficient Searchable Encryption with Single Trapdoor for Multi-Owner Data Sharing in Federated Cloud Computing

[1]Vadlamani Veerabhadram, [2]Dr. Gregory Arul Dalton
[1]Assistant Professor, CVR College of Engineering,
Research Scholar, Jawaharlal Nehru Technological university Hyderabad, Telangana,india
drbhadram@gmail.com
[2]Professor of CSE, Princeton Institute of Engineering and Technology for Women Hyderabad, Telangana, India
aruldalton@rediffmail.com

Abstract-Cloud computing, an Internet based computing model, has changed the way of data owners store and manage data. In such environment, data sharing is very important with more efficient data access control. Issuing an aggregate key to users on data enables and authorizes them to search for data of select encrypted files using trapdoor or encrypted keyword. The existing schemes defined for this purpose do have certain limitations. For instance, Cui et al. scheme is elegant but lacks in flexibility in access control in presence of multiple data owners sharing data to users. Its single trapdoor approach needs transformation into individual trapdoors to access data of specific data owner. Moreover, the existing schemes including that of Cui et al. does not support federated cloud. In this paper we proposed an efficient key aggregate searchable encryption scheme which enables multiple featuressuch as support for truly single aggregate key to access data of many data owners, federated cloud support,query privacy, controlled search process and security against cross-pairing attack. It has algorithms for setup, keygen, encrypt, extract, aggregate, trapdoor, test and federator. In multi-user setting it is designed to serve data owners and users with secure data sharing through key aggregate searchable encryption The proposed scheme supports federated cloud. Experimental results revealed that the proposed scheme is provably secure withrelatively less computational overhead and time complexity when compared with the state of the art.

Keywords – Security, Key Aggregate Searchable Encryption, Federated Cloud, Singe trapdoor search, Query Privacy.

I. INTRODUCTION

Cloud computing is widely used for data storage and computing resources. The services rendered by cloud in terms of data storage, retrieval and sharing of data are provided through Internet. Therefore, the service is linked to untrusted channels and vulnerable to attacks and data leakage. Data owners need to encrypt data before outsourcing to cloud to avoid leakage. However traditional encryption is not flexible for data sharing and searching on the encrypted data. It is important to have mechanisms to share data to multiple users with better access control mechanisms. Sharing data and security keys to intended users is important. There are many searchable encryption schemes such as [1], [6], [8], [9] to mention few. These schemes support keyword search on encrypted contents. It is found that searchable encryption is very much suitable for data sharing over public cloud. Such schemes also prevent adversaries from lunching attacks on data.

In this context, the aim of the research presented in this paper is to propose a security scheme that fulfils many security requirements. They include searchability, compactness, keyword privacy, and aggregate key unforgeability. From the literature we found that the work in [6], [40] and [41] is close to our work in this paper. Cui et al. [6] proposed a scheme for key aggregate searchable encryption. In this scheme data owner provides aggregate keys to users to enable them to access only authorized data. It is done by users with generation of trapdoors that facilitate searching over encrypted data. This scheme is found efficient as its size of aggregate keys and cipher texts does not rely on number of users and number of documents. However, it has some limitations in terms of keyword privacy and aggregate key unforgeability. Aggregate searchable encryption schemes are also found in [40] and [41]. Authors of [40] proposed verifiable searchable encryption where only one aggregate key is sent by data owner to user which is meant for search and also verification. Wang et al. [41] proposed a searchable encryption process which is authorized through secure key-aggregation. The three schemes aforementioned are able to support one of the security requirements. Compactness is supported by [6] and [40] while [41] supports keyword privacy. The scheme in [42] also does not support multi-owner environment also. In addition to the aforementioned drawbacks, the existing schemes do not support single

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6964
***Article Received**: 28 March 2023 **Revised**: 18 May 2023 **Accepted**: 01 June 2023*

trapdoor based access to data of multiple data owners and also lack support or federated cloud. We designed a scheme to overcome the drawbacks and support single trapdoor and federated cloud besides common security requirements. Our contributions are as follows.

We proposed a key aggregate searchable encryption scheme that leverages security and flexibility to secure data and share it among cloud users with controlled access to authorized documents. Particularly, our scheme enables users to gain access to data of multiple owners with a single trapdoor leading to computational and communication efficiency. The proposed scheme is designed work with federated cloud as well which brings more flexibility in cloud data sharing. We proposed several algorithms that constitute the proposed scheme for more optimal and efficient key aggregate searchable encryption. We made security analysis of the proposed and existing schemes such as [40] and [42] against many security features such as query privacy, controlled search process and security against cross-pairing attack. Our study also throws light on the

1. We implemented the proposed scheme to evaluate the algorithms and overall scheme in multi-user cloud environment and the observations are compared against state of the art schemes.

The remainder of the paper is structured as follows. Section 2 reviews literature related to security schemes particularly on key aggregate searchable encryption methods suitable for cloud environments. Section 3 presents the proposed scheme with underling algorithms. Section 4 presents security analysis of the proposed scheme. Section 5 presents comparison of proposed and existing schemes. Section 6 presents results of experiments in terms of time complexity and space complexity. Section 7 concludes the paper and gives directions for future scope of the research.

II. RELATED WORK

This section reviews literature related to security schemes particularly on key aggregate searchable encryption methods suitable for cloud environments. Dong et al. [1] proposed a searchable encryption scheme with specific keys to users of the system. It is implemented to have secure data sharing in untrusted servers' environment. Li et al. [2] proposed a methodology for convergent management scheme for security and data deduplication in distributed environments. Centry and Peikart [3] considered worst case security scenarios and proposed trapdoor construction methods. Lu et al. [4] proposed an aggregate signature scheme which is based on lattice and unordered in nature. It solves the problem of aggregate signature using intersection method. Reddy and Gopal [5] proposed a scheme for data security and sharing of data with aggregate signature scheme using identity-based keys. Cui et al. [6] proposed a key aggregate searchable encryption scheme for cloud computing. It enables data owner to share data across users with secure access control. Chen et al. [7] investigated on attribute based signatures with a novel approach suitable for outsourced security mechanism. Li et al. [8] considered hybrid cloud for building an access control mechanism that supports keyword search on the encrypted data.

Curtmola et al. [9] made efficient constructions for searchable symmetric cryptography. Chu et al. [10] defined key-aggregate encryption mechanisms with data sharing support with required scalability. It is designed to be feasible for cloud storage. Gu et al. [11] focused on lattice based approach for public key encryption which supports keyword search. The search mechanism works on the encrypted data. Boneh et al. [12] proposed schemes for broadcast encryption. These schemes are with private keys and short cipher texts besides being collision resistant. Regev [13] focused on reducing worst case lattice problems by analysing concepts such as learning with errors, cryptography and random linear codes. Li and Kim [14] proposed a novel approach known as hidden lattice based signatures but does not consider anonymity revocation in their approach. Centry et al. [15] proposed a method for sequential aggregate signature based on trapdoor-permutation concept towards a unified security framework.

Boneh et al. [16] proposed a scheme for identity based encryption which is space efficient and does not consider pairings. It is based on the theory of ternary quadratic forms. Alwen and Piekert [17] focused on the problem of generating random lattice which is hard. Their idea of generation of hard random lattices is based on shorter bases generation. Du et al. [18] proposed an aggregate encryption scheme without certificates. It was designed to be useful for data security in healthcare industry. Shor [19] focused on discrete logarithms and prime factorization which is used in polynomial-time algorithms required for quantum computing scenarios. Liu et al. [20] proposed a coarse-grained access control approach useful for hybrid cloud with searchable encryption. This scheme is designed to work in multi-user environment. From the literature, it is ascertained that there are many security schemes suitable for cloud environment to support data sharing and access control in secure fashion. However, the research in [6], [40], [41] and [42] is closely related to the work in this paper. These schemes do not support single trapdoor for accessing data of multiple owners and federated cloud. To overcome this drawback, we proposed a scheme that outperforms state of the art.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6964
Article Received: 28 March 2023 Revised: 18 May 2023 Accepted: 01 June 2023

III. PROPOSED SCHEME

We proposed a KASE scheme that is designed to leverage performance and efficiency of both data owners and data users in multi-owner, multi-user federated cloud environment.

3.1 Problem Definition

Before defining the problem, it is useful to know the breakthrough achieved by existing KASE schemes explored in [29], [40] and [42]. KASE enables secure sharing of data in multi-user cloud environment where data owners encrypt data and generate aggregate keys for the same. These aggregate keys are shared to data users to have authorized search over encrypted data. Towards this end, data users can generate trapdoors to search the encrypted data to obtain required data directly. Each data owner might have number of data users. The breakthrough achieved by KASE is that the aggregate key size and cipher text size remains constant for any number of data users. Thus KASE has potential to leverage operational efficiency in multi-user setting of cloud storage. Though KASE is designed to solve problems in group data sharing which is indispensable in cloud environments, it has significant limitations.

1. The existing KASE solutions need a trapdoor to be generated by a data user to gain access to all documents shared by a single data owner. It does mean that data users must have multiple trapdoors to access data in multi-owner setting. A data user who desires to query electronic documents authorized to him by multiple owners needs to send multiple trapdoors to cloud server to get access to those documents. As of now, there is no provision to have a single trapdoor generation by a data user to access data of multiple users. This is the potential limitation of existing KASE schemes.
2. Federated cloud is the seamless integration of different cloud platforms for achieving diversified business needs, scalability and availability. It combines all kinds of cloud deployments with a common standard. It offers vendor-neutral computing services on demand to be highly robust and scale up the resources beyond any boundaries. The existing KASE solutions were not designed to run in federated cloud environments. They are designed to run a single cloud multi-owner and multi-user setting. Designing a KASE scheme to work in federated cloud is another challenging problem.

3.2 Our System Model

We designed a system model for proposing a novel KASE scheme to address problems mentioned in Section 3.1. It reflects a federated cloud environment that is not explored in existing KASE research found in [29], [40] and [42]. Let the federated cloud denoted as F= $\{C_1, C_2,\ldots, C_n\}$. There are multiple owners who share electronic documents to authorized users in a meaningful cloud based application. Multiple data owners are denoted as O=$\{o_1, o_2, \ldots, o_n\}$. Each data owner may share number of electronic documents, denoted as D=$\{d_1, d_2, \ldots, d_n\}$, to authorized users. There are multiple data users who are authorized to access electronic documents of multiple owners denoted as U=$\{u_1, u_2, \ldots, u_n\}$. Our system model is illustrated in Figure 1. Each data owner encrypts documents and associated keywords using our proposed scheme described in Section 3.3. The encryption results in cipher text for both electronic document and its associated keywords. Then the encrypted content is saved to federated cloud. The storage in federated cloud does mean that each encrypted document may be stored in one of the constituent clouds. Data storage and retrieval are managed by a program known as federator. Documents {d1, d2, d3, d4} shared by data owner o1 may be saved in constituent clouds C1, C2, C3 and C4 in F. Then for set of documents of o1, an aggregate key is generated and shared to one or

more data owners. In the same fashion, documents {d1, d2, d3, d4} shared by data owner o2 may be saved in constituent clouds C1, C2, C3 and C4 in F. Then for set of documents of o2, an aggregate key is generated and shared to one or more data owners.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6964
***Article Received**: 28 March 2023 **Revised**: 18 May 2023 **Accepted**: 01 June 2023*

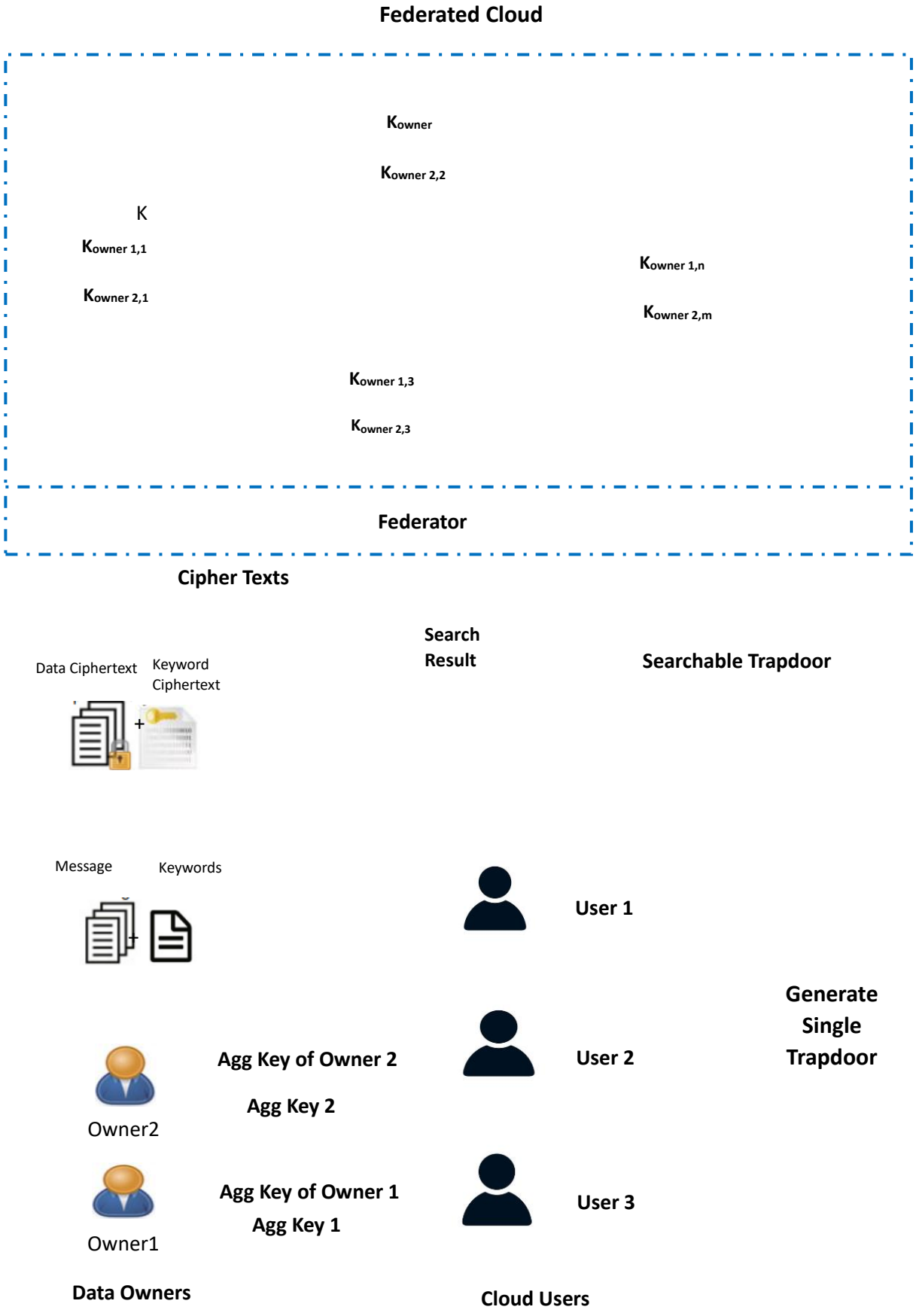

Figure 1: Proposed system model for efficient key aggregate searchable encryption with single trapdoor for accessing files of multiple users in federated cloud environment

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6964
***Article Received**: 28 March 2023 **Revised**: 18 May 2023 **Accepted**: 01 June 2023*

The novelty of our method is that when aggregate keys $\{k_{o1,agg}, k_{o2,agg}, k_{on,agg}\}$ of multiple owners are shared to data users an overall aggregate key is generated by data user. Then data user can generate single searchable trapdoor for accessing all encrypted documents of multiple owners. Another novelty of our scheme is to work in federated cloud environment which is indispensable in the future of cloud usage. Then the user submits the trapdoor to federated cloud where federator uses the trapdoor to search for documents across the constituent clouds and returns search results to data user. In the proposed system model data user is not generating a separate trapdoor for accessing each data owner's shared documents unlike existing KASE schemes discussed in [29], [40] and [42]. The provision to exploit federated cloud improves flexibility of our scheme as it is suitable for future of the cloud deployments in the real world. Another important aspect of our scheme is to allow data user to have single trapdoor for all shared documents by multiple data owners which makes it further efficient computationally and ease flexibility in operations.

There are many use cases of our scheme in real world applications. For instance, in healthcare domain, a doctor (data user) can view different medical documents of many patients with a single trapdoor. In fact, doctor need not be an expert in cryptography as the scheme is built in an application in a user-friendly fashion. Moreover, the data user (doctor) does not know the internal operations pertaining to security as they are transparent to end users. Another use case suitable for using our scheme is that a project manager (data user) can gain access to all work or documents of team members (data owners) in a software development company. This way thousands of use cases are possible in cloud-assisted real world applications.

3.3 Our Scheme

Our scheme for KASE with single trapdoor for multi-owner data sharing in federated cloud computing is realized by defining a set of algorithms as presented in Table 1. The parties involved in our scheme include Cloud Service Provider (CSP) or cloud server, data owner who has original documents to be shared to a group of users and data user who is authorized to access documents shared by data owners.

Table 1: Brief description of our algorithms

Algorithm	Signature	Description	Who runs it?
Setup	Setup(1^{λ},n)	It produces system parameters required for security by taking inputs such as security parameter (λ)and total number of documents (n) of data owner.	CSP
Keygen	Keygen	It generates a key pair public key (pk)-secret key (msk)	Data owner
Encrypt	Encrypt(pk, i, m, KW)	The i^{th} document is encrypted using inputs such as public key (pk), cipher text class (i), a specific keyword (KW) and message (m)	Data owner
Extract	Extract(msk, S)	Generates ASE key (kagg). It is used to give access on set of documents of owner to set of users. It takes inputs as secret key of data owner (msk) and his owned set of documents (S).	Data owner
Aggregate	Aggregate(kagg1, kagg2,...,kaggk, S1, S2,..., Sk)	Takes aggregate keys of multiple owners and generates a single aggregate key (kagg).	Runs at user side
Trapdoor	Trapdoor(kagg, w)	By taking aggregate key (kagg) and keyword (w), it generates a single trapdoor (Tr) with which user can access documents shared by multiple owners.	Data user
Test	Test(Tr, S, i)	Performs actual search operation over encrypted documents (C) using trapdoor (Tr), set of shard documents (S) and document index i.	CSP
Decrypt	Decrypt(kagg, S, i, Ci)	It takes cipher text class (Ci), index I, set of permitted documents S and aggregate key (kagg) as input and	Data user

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6964
Article Received: 28 March 2023 Revised: 18 May 2023 Accepted: 01 June 2023

		results in decrypted document (m)	
Federator	Federator()	This will be used only to delegate execution of Setup and Test algorithms to constituent cloud service providers in federated cloud.	This algorithm is automatically invoked by the scheme while performing Setup and Test functionalities in federated cloud.

These algorithms are designed to achieve multiple objectives such as compactness, correctness, proper delegation of rights, preserving query privacy, strict authorization and resistance to collision attacks. Compactness denotes that the size of aggregate key and is independent of number of files and number of cipher text classes. Another requirement of compactness objective is constant size of ciphertext. Correctness refers to the fact that a document (m) search by user using a keyword (w) should return true only when that exists in ciphertext documents (C) with given trapdoor (Tr). Proper delegation of rights indicates that search and decryption rights from data owner on collection of documents are delegated to users and users can perform search and decryption operations accordingly. Table 2 shows different notations used in the proposed scheme.

Table 2: Notations used in our scheme

Notation	Description
k_{agg}	Denotes aggregate key given by data owner
δ_i	Denotes public information
C_i	Denotes ciphertext
D	Denotes large amounts of data
I	Denotes index
k_i	Denotes an encryption key
KW	Denotes keyword in a document
M	Denotes a message
Msk	Denotes a master secret key
N	Denotes number of documents of a data owner
Pk	Denotes public key
S	Denotes a set of indices
S1, S2,...,Sk	Denotes different sets of document indices
SP	Denotes generated system parameters
Tr	Denotes a trapdoor
Tr_i	Denotes a trapdoor output
W	Denotes a query keyword
B	Denotes a secret key
λ	Denotes security parameter

Preserving query privacy does mean that the trapdoor (Tr) given by user to cloud server should not disclose its contents to any third party including cloud server. Strict authorization refers to the fact that only authorized users should be able to access data as per access rights and any third party is never allowed to do so. Resistance to collision attacks refers to the fact that users having multiple aggregate keys should not be able to access information of other aggregate key by combining two aggregate keys. For instance, kagg1 (to access set of documents denoted as S1) and kagg2 (to access set of documents denoted as S2) cannot be combined to access all documents from S1 and S2. Trying to access like this is referred to as collision attack.

3.3.1 Setup Algorithm

This algorithm is executed by cloud server to generate system parameter, denoted as SP=(B, PubK, T $*$), that are used later for secure communications. B refers to a bilinear map group system denoted as B=(p, G, GT , e(·)) G should have an order denoted by p such that $2\lambda \leq p \leq 2\lambda+1$. The second system parameter is a public key (PubK) which is expressed as PubK=(g, g1, ..., gn) $\in$ Gn+1. In the public key

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6964
***Article Received**: 28 March 2023 **Revised**: 18 May 2023 **Accepted**: 01 June 2023*

expression number of documents is referred to as n which is associated with set of documents D= {doc1, doc2..., docn}. These documents belong to data owner. The value of n is incremented as the data owner uploads more documents to cloud. It results in adding another public parameter, denoted as gn+2, is incorporated into PubK. This is carried out without causing any modifications to other parameters. Thus it is known that (g, g1, ..., gn, gn+1) is the pattern in which PubK gets updated from time to time. There is possibility of increase in number of classes of ciphertext. For instance, ciphertext classes increase to n+m can affect PubK leading to (g1, g2, ..., gn, gn+2, gn+3, ..., gn+m) to be reflected in PubK. This actually occurs when data owner saves documents that are beyond pre-defined number of ciphertext classes to cloud. Therefore, our scheme is scalable to number of ciphertext classes which is expressed as gi = gα ∈G for i = {1, 2,..., n} where α is a random number such that α ∈ Zp. This third parameter in SP is T ∗. It denotes a map of keywords and corresponding unique values. Provided keywords of m categories in the system, the keyword space is denoted as ks={w1, w2, ..., wm} and, for 1 < i < m, it is associated with value space KVi={KV1, KV2, ..., KVm}. Provided this knowhow, a hash function denoted as H(KVi) shows the underlying server operation for each m. Given wi and KVi, as key-value pair, cloud server checks whether it has been part of previous queries. If not, it computes Ti∗ = gkvi where kvi is the random value such that kvi ∈RZp. Afterwards, the server updates table (Tkw) with a tuple denoted as <wi, KVi, kvi, Ti∗> and returns Ti∗.

3.3.2 Keygen Algorithm

This algorithm generates a key pair namely public key (pk)-secret key (msk). It is executed by data owner to have pk-msk pair such that:

pk=v=gγ, msk=γ, where γ ∈ Zp

The generated key pair plays crucial role in other algorithms of our scheme.

3.3.3 Encrypt Algorithm

In the proposed scheme Encrypt plays important role in converting plain data into cipher text in such a way that it can be shared to number of users. Data owner performs encryption process. The underling signature of the method is as follows.

Encrypt(pk, i, m, KW)

The i^{th} document is encrypted by user using inputs such as public key (pk), cipher text class (i), a specific keyword (KW) and message (m). The algorithm generates cipher text for data and cipher text for keyword. This algorithm's functionality is split into number of steps as given below.

Step 1: Searchable encryption key ki is set to t as follows. ki= t, and t ∈ Zp

Step 2: Public information is computed as follows. δi = (C1, C2) for ki

C1 = gt

C2=e(v, gi)=e(gγ, gi)

Step 3: Cipher text generation for given keyword.

C3=CKW=(gkvi)t

Step 4: Cipher text generation for given document. m=doci

C4 = Cm=m · e(g1, gn)t

In the public parameters of PubK it is observed that gn+1 = gαn+1 to restrict an attacker from gaining information about message m. After generating cipher texts for keyword and document, the data owner sends them to cloud server.

Ci = (δi, CKW, Cm) = (C1, C2, C3, C4)

3.3.4 Extract Algorithm

Extract algorithm is executed by data owner to generate ASE key (kagg). It is used to give access on set of documents of owner to set of users. It takes inputs as secret key of data owner (msk) and his owned set of documents (S). This algorithm helps in rendering data access and keyword search to a group of users on set of documents. The result of this algorithm is an aggregate key provided by a data owner. In this fashion, in our scheme, multiple data owners can generate kagg and give it to data users. This algorithm has following steps.

Step 1: Algorithm takes secret key of data owner (msk) and his owned set of documents (S) as inputs.

S ⊆ {1, 2,...,n}

Step 2: Algorithm computes kagg.

kagg = πj∈S gγ j

Step 3: Data owner sends kagg and S to set of users to give them access rights and delegate keyword search.

3.3.5 Aggregate Algorithm

This algorithm takes aggregate keys of multiple owners and generates a single aggregate key (kagg). This plays crucial role in our scheme where it is intended to reduce number of trapdoors to only one for user to access shared documents of multiple owners. It has the following steps.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6964
Article Received: 28 March 2023 Revised: 18 May 2023 Accepted: 01 June 2023

Step 1: It takes aggregate keys from many data owners on sets of documents kagg1, kagg2,...,kaggk on S1, S2,...,Sk

Step 2: It generates single aggregate key

$$kagg = kagg1 \cdot kagg2 \cdot ... = \pi j \in S1\ g\gamma 1\ j \cdot \pi j \in S2\ g\gamma 2\ j \cdot ... = \pi j \in S, i \in \{1,2,...,k\} g\gamma i\ j = kagg$$

This algorithm provides access rights to users on all the documents shared by multiple users by enabling single trapdoor generation.

$$S = S1 \cup S2 \cup ... \cup Sk$$

An authorized user has kagg1 associated with set of documentsof owner 1 denoted as S1. In the same fashion, another user has kagg2 associated with set of documentsof owner 2. This way multiple data owners send aggregate keys to authorized data users. Aggregate(·) algorithm helps in generation of an aggregate key kagg for all aggregate keys of different data owners. This kagg enables authorized users to generate a single trapdoor to gain access to shared documents of multiple data owners.

3.3.6 Trapdoor Algorithm

By taking aggregate key (kagg) and keyword (w), it generates a single trapdoor (Tr) with which user can access documents shared by multiple owners. This algorithm is executed by data users. Instead of generating a trapdoor for shared documents of each data owner, this algorithm generates a single trapdoor Tr. This is an important contribution of our scheme. It has the following steps.

Step 1: It takes aggregate key (kagg) and keyword (w) as inputs

Step 2: For given keyword w, this algorithm computes single trapdoor Tr

$$Tr = (Tr0, Tr1) = (kagg \cdot (gkvi)b, gb), \text{ where } b \in Zp$$

After generating Tr, user needs to send Tr and S to server to enable it to perform desired keyword search. User can perform search on any document that belongs to S as far as the set of documents come under the purview of Tr.

3.3.7 Test Algorithm

This algorithm is executed by cloud server to perform keyword search expected by end users. In fact, it performs actual search operation over encrypted documents (C) using trapdoor (Tr), set of shard documents (S) and document index i. It has the following steps.

Step 1: It takes Tr, S and i as inputs.

Step 2: If i ∈ S Then, it returns true indicating that keyword associated with cipher text is same as that of keyword in the query. If not, it returns false.

While running this algorithm, the cloud server receives trapdoor:

$$Tr = (Tr0, Tr1)$$

Then the server runs the algorithm to perform keyword search on i^{th} document and returns NULL if the document does not belong to S. Otherwise the algorithm returns result as:

$$R \in \{0, 1\}$$

R holds 1 if the i^{th} document is associated with given keyword w and R holds 0 if the i^{th} document has no such keyword. In order to know whether ith document has keyword, the server computes

$$C = \pi j \in SC2,j = \pi j \in S\ e(v, gj)$$

In order to search over data of multiple owners, the server generates C using:

$$C2,j \mid \forall j \in S$$

Where S holds all shared documents of k data owners. The judgement of server to return true or false as desired is as follows.

$$0 \text{ or } 1 = ? = e(C, C1) \cdot e(C3, Tr1)/e(Tr0, C1)$$

3.3.8 Decrypt

This algorithm takes cipher text class (Ci), index I, set of permitted documents S and aggregate key (kagg) as input and results in decrypted document (m). This algorithm is executed by an authorized user to gain access to the encrypted documents retrieved using kagg given by a data owner. It has the following steps.

Step 1: It takes cipher text class (Ci), index I, set of permitted documents S and aggregate key (kagg) as input and results in decrypted document (m) as input.

Step 2: If i ∈ S then it returns the message m, else it returns NULL.

$$m = C_4 e(k_{agg} \pi_{j \in S, j=n} g_{j+1}, C_1)/e(pub \cdot C', C_1)$$

3.3.9 Federator Algorithm

In the proposed architecture, federator is the component that facilitates cloud server side functionalities since there are multiple cloud platforms combined. This component has an associated algorithm known as **Federator** which has provision to deal with constituent cloud platforms appropriately. This algorithm is required only in case of

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6964
***Article Received**: 28 March 2023 **Revised**: 18 May 2023 **Accepted**: 01 June 2023*

Setup() and Test() functions as they are to be carried in the cloud server.

IV. SECURITY ANALYSIS

This section analyses security in terms of query privacy, controlled search process and security against attack such as cross-pairing.

4.1 Query Privacy

Query privacy refers to the fact that an attacker cannot learn a keyword associated with a query that is submitted by user in the form of trapdoor. and cannot learn a keyword linked to stored keyword-cyphertext. This feature of the proposed scheme is divided into two propositions.

First One:An attacker cannot know a keyword which is part of a user query associated with a trapdoor

Here is the proof of the proposition. When an attacker, denoted as A, wanted to know a keyword from query of trapdoor submitted, A needs to have aggregate key k_{agg}which is not possible. The reason behind this is that A knows only public parameters SP = $(B,\ PubK, Tr^{*})$ and with them, adversary cannot obtain aggregate key from T_r which is expressed as follows.

$T_r = \ (Tr_0\ , Tr_1) \ = \ \left(k_{agg} \cdot \left(g^{kvi}\right)^{b} . g^{b}\right)$ where $b \ \in \ Z_p$

For each document in S, attacker need to compute g_i^{γ} where secret key of data owner γ is involved. Therefore, it is not possible for the adversary to learn any keyword from the given query.

Second One: An attacker cannot know a keyword using public information and outsourced keyword-ciphertext

Here is the proof of the proposition.A cloud server has access to public information and outsourced keyword-ciphertext. Assume that cloud server is an attacker and tries to obtain a keyword from the encrypted data. Public information such as δi = (C1, C2) and public key PubK = (g, g1,...,gn) ∈ Gn+1 cannot help the attacker to know a keyword *t* as it involves discrete logarithm problem. Thus attacker fails in obtaining actual value of *t*. The two propositions show that the proposed scheme ensures query privacy.

3.4.2 Controlled Search Process

The proposed scheme ensures controlled search process. It does mean that an authorized user can only perform search on set of documents associated with given aggregate keys and the user cannot new aggregate keys to gain access to unauthorized set of documents. This is reflected when four propositions are combined.

First One: An attacker cannot know a keyword which is part of a user query associated with a trapdoor

Here is the proof of the proposition. It also proves that the proposed scheme is correct. On obtaining Tr, Test is invoked at the server side in order to perform keyword search without the need for transforming the single Tr into owner-dependent Tr. Since the Tr consists of kagg for search over documents shared by multiple data owners, adversaries trying to know keyword associated with user query will fail. On receiving Tr from user, the server computes $C^{'} = \pi_{j\epsilon s} = \pi_{j\epsilon s}$, i∈{1,2,...,m} e($g^{\gamma i}$, g_j) and performs keyword search $e\left(C^{'}, C_1\right) e(C_3, Tr_1) \Big/ e(Tr_0, C_1)$ which is computed further as follows.

$$= \frac{e\left(\pi_{j\epsilon s},\ \mathrm{i} \in \{1,2,\ldots,\mathrm{m}\}\ \mathrm{e}(\ g^{\gamma i}, g_j), g^{t}\right) . e\left(\left(g^{kv_i}\right)^{t}, g^{b}\right)}{e\left(\left(k_{agg} \cdot \left(g^{kv_i}\right)^{b}, g^{t}\right)\right)}$$

$$= \frac{e\left(\pi_{j\epsilon s},\ \mathrm{i} \in \{1,2,\ldots,\mathrm{m}\}\ \mathrm{e}(\ g^{\gamma i}, g_j), g^{t}\right) . e\left(\left(g^{kv_i}\right)^{t}, g^{b}\right)}{e\left(k_{agg}, g^{t}\right) . e\left(\left(g^{kv_i}\right)^{b}, g^{t}\right)}$$

=1.

Thus, it proves that only the user who has valid Kagg can perform keyword search successfully.

Second One: An attacker cannot know a keyword using public information and outsourced keyword-ciphertext

Here is the proof of the proposition. A cloud server has access to public information and outsourced keyword-ciphertext. Assume that cloud server is an attacker and tries to obtain a keyword from the encrypted data. Public information such as δi = (C1, C2) and public key PubK = (g, g1,...,gn) ∈ Gn+1 cannot help the attacker to know a keyword *t* as it involves discrete logarithm problem. Thus attacker fails in obtaining actual value of *t*. The two propositions show that the proposed scheme ensures query privacy.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6964
***Article Received**: 28 March 2023 **Revised**: 18 May 2023 **Accepted**: 01 June 2023*

Third One: When an authorized user becomes malicious and collides with server, he cannot perform keyword search for any document that is not in the scope of his aggregate key.

Here is the proof of the proposition. Since each data owner performs key aggregation and has his secret key kept private, this kind of collision does not work. Adversary will fail to get secret key or keyword from the trapdoor given to him. Even by combining many aggregate keys, user will not be able to infer required information. Thus users cannot perform collision attacks to search or decrypt the documents which are not associated with the given trapdoor.

Fourth One: For documents shared by data owners, an attacker cannot produce a new aggregate key based on the known aggregate key.

Here is the proof of the proposition. An authorized user turned adversary, denoted as A, has his aggregate key denoted as k_{agg} which is meant for accessing set of documents S. Computing new aggregate key to access documents not in the range of S is not possible. It is because computation of new aggregate key needs data owner's secret key. Therefore, adversary cannot generate aggregate key for other documents. With the four propositions, it is proved that controlled searching is made possible in the proposed scheme.

3.4.2 Security Against Cross-Pairing Attack

The proposed scheme does not allow cross-pairing attack. An attacker (malicious user or cloud) may analyse stored ciphertext for which he is not authorized. The knowledge of PubK = (g, g_1,...,g_n) ∈Gn+1 and other information such as δ_i= (C_1, C_2). It is also linked to $g_1 = g^t$, C_2 = e(v, g_i) = e(g^γ , g_i) and adversary tries to get m through computation of $e(g_1, g_n)^t$. As the term denoted as $g_{n+1} = g^{\alpha^{n+1}}$ is not part of PubK, it is not possible for an adversary to know $e(g_1, g_n)^t$. In other words, adversary cannot make cross-pairing attack based on reuse of PubK information and δ_i. Moreover, as per the discrete logarithm problem, A cannot get the value associated with t. In addition to this, using PubK information and keyword-ciphertext also cross-pairing attack is not possible as the proposed scheme protects contents of Tr with b $\epsilon_R Z_p$. The cross-pairing therefore generates incompatible values and thus attack fails.

V. PERFORMANCE COMPARISON

This section provides analysis and comparison of the proposed and existing KASE schemes in terms of their features, complexity and overhead.

5.1 Feature Comparison

KASE schemes presented in [40] and [42] are compared with the proposed scheme. As presented in Table 3, important features are compared among all the schemes. The features include the search process support in multi-owner data, support for single aggregate key, whether transformation is required from single trapdoor to multiple individual trapdoors and support for federated cloud.

Table 3: Comparison of schemes in terms of important features (* indicates need to have multiple trapdoors to be obtained before submitting to server)

Scheme	Search of Multi-Owner Data	Support for Single Aggregate Key	Transformation Required to Have Individual Trapdoors	Federated Cloud Search	Security Against Cross-Pairing Attack
[42]	No	No	Yes	No	No
[40]	Yes *	Yes	Yes	No	No
Proposed Scheme	Yes	Yes	No	Yes	Yes

The proposed scheme supports users to search data of multiple owners using a single aggregate key which reduces in overhead and complexity. Besides the proposed scheme supports federated cloud which is essential due to emerging trends in the cloud infrastructure provisioning. The proposed scheme directly allows users to use single trapdoor without the need for transforming that into number of individual trapdoors unlike [40] and [42] that do not. The existing KASE schemes [40] and [42] do not support federated cloud search.

5.2 Communication Complexity

Communication overhead of the proposed and existing ([40], [42]) KASE schemes is analysed. It is based on a

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6964
Article Received: 28 March 2023 Revised: 18 May 2023 Accepted: 01 June 2023

search request sent to the server in a context of multi-owner sharing data to users. Since the proposed scheme needs only single trapdoor to be sent to federated cloud its communication cost is constant such as O(1). Thus it is more efficient when compared with existing ones [40] and [42]. Table 4 shows the communication overhead comparison among the KASE schemes.

Table 4: Communication overhead comparison for multi-owner data search

Scheme	Communication Cost (multi-owner data search)
[42]	T*U
[40]	U+T*A
Proposed Scheme	T

Since the proposed scheme needs only single trapdoor to search in the context of multi-owner data sharing. The scheme in [42] needs communication cost of number of trapdoors (T) multiplied by number of data owners (U). The scheme in [40] has the overhead of T multiplied by required number of auxiliary values for the process of searching plus U. Thus the proposed scheme outperforms existing ones in terms of communication cost.

5.3 Storage Overhead

KASE schemes need storage for PubK, k_{agg}, Tr and encrypted data (C). The storage overhead of the proposed scheme and the existing ones is provided in Table 5.

Table 5: Storage overhead of different KASE schemes

Scheme	Tr (multi-owner)	PubK	Ciphertext
[42]	U|G|	(2n + 1)|G|	2G+ G_T
[40]	|G| + U * A	(2n + 1)|G	3G+G_T
Proposed Scheme	2|G	(n + 1)|G	2G+ G_T

The public key overhead is based on associated number of documents (n) of each data owner. The number of documents (n) and bilinear group G are affecting the PubK storage overhead of the schemes [40] and [42] and proposed scheme. However, the proposed scheme has less overhead when compared with existing ones. Tr storage overhead is depending on the G and U in case of [40] and [42] but the proposed scheme depends on G only. With regard to ciphertext storage overhead, all the schemes' overhead is based on G and G_T (two bilinear groups). However, the ciphertext storage overhead of the proposed scheme is less than that of [42] but equal to that of [40].

5.4 Computational Cost

The computational overhead of the proposed and existing schemes is provided in Table 6 in terms of overhead required by trapdoor generation, generation of aggregate key, encryption and keyword search in multi-owner setting.

Table 6: Computational overhead of all schemes for different operations

Scheme	Tr Generation	Kagg Generation	Encryption	Keyword Search
[42]	U ·M	-	2E+3P +2M	(|S|·M)+ U(|S|·M+2P)
[40]	(M+E)+ U(M+E)	U ·M	2E+3P +2M	U(|S|·M)+ U(|S|+2M)+2P
Proposed Scheme	2E+M	U ·M	2E+2P +M	3P +|S|·M

In the overhead values E denotes exponentiation, U denotes number of data owners, P denotes the process of pairing, M denotes scalar multiplication, S denotes a set while |S| denotes size of S. Trapdoor generation computation cost of the proposed scheme is constant and its does not rely on U while the existing methods' computational overhead increases when number of data owners increase. While performing keyword search also, proposed scheme needs constant number of pairing operations. Moreover, the existing schemes cause more overhead as they need to use trapdoor transformation process. On the other hand, proposed scheme allows direct search using single trapdoor without generating trapdoors for individual users. They keyword search and query performance of the proposed scheme is also better than existing methods. Kagg generation of the proposed scheme needs same computational overhead as that of [40] while encryption process of the proposed scheme outperforms existing ones.

VI. EXPERIMENTAL RESULTS

We built a prototype to evaluate our scheme and also that of [40] and [42] with simulation study. Around 40 experiments are conducted with each operation of the scheme involved. Average of values for parameters considered are used for comparison. Experiments are done using a computer with Windows 10 64-bit OS, Intel Core i5-4210U CPU @ 1.70 GHz, 4 GB RAM and Intel HD graphics family. Simulations are made to observe performance of schemes in terms of space and time complexity against number of data owners, number of keywords, size of trapdoor and size of system parameters.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6964
***Article Received**: 28 March 2023 **Revised**: 18 May 2023 **Accepted**: 01 June 2023*

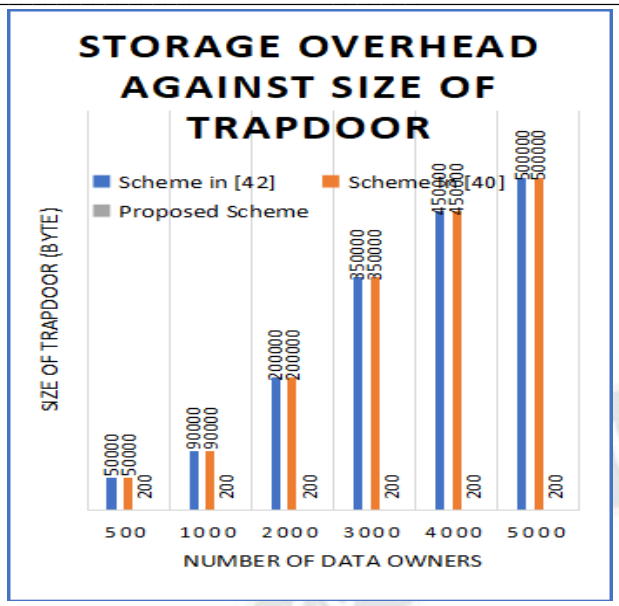

Figure 2: Storage overhead in terms of size of trapdoor against number of data owners

The observations provided in Figure 2 reveal size of trapdoor against number of data owners. It is observed that there is linear increase in the size of trapdoor in case of existing schemes in [40] and [42]. However, in the proposed method, the size of trapdoor is constant even if the number of data owners increased. The scheme in [40] does not support single trapdoor. Therefore, its size depends on the number of data owners giving access to their data to users. In the same fashion, the scheme in [42] exploits single trapdoor concept but it has to transform it into multiple individual trapdoors leading to linear increase in the size of trapdoor.

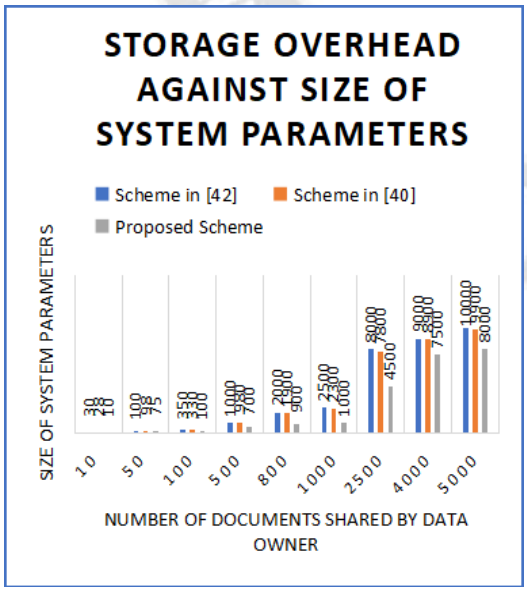

Figure 3: Storage overhead in terms of size of system parameters against number of documents shared by data owner

With respect to size of system parameters, the proposed scheme needs n+1 (where number of documents is denoted by n). Whereas the existing schemes need 2n+1 in terms of size of system parameters. This will also affect computation time required for different operations.

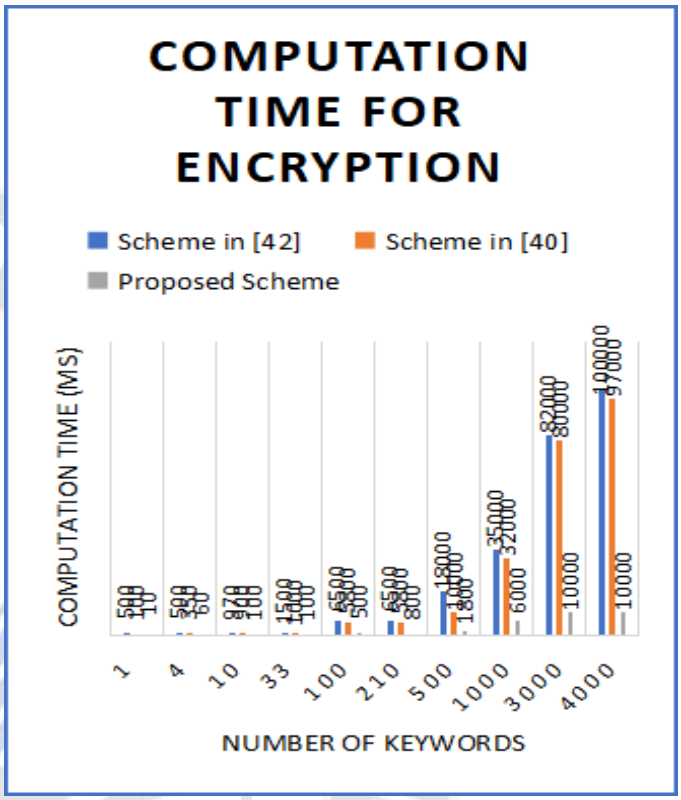

Figure 4:Computation time for encryption against number of keywords

As presented in Figure 4, the computation time required by encryption algorithm is observed against number of keywords. As the number of keywords is increased, it causes overhead in a linear fashion. It is evident with all the schemes. However, the computation time of the proposed scheme is significantly less than existing schemes. The rationale behind this is that the encryption algorithm involves less number of multiplication and pairing operations.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6964
***Article Received**: 28 March 2023 **Revised**: 18 May 2023 **Accepted**: 01 June 2023*

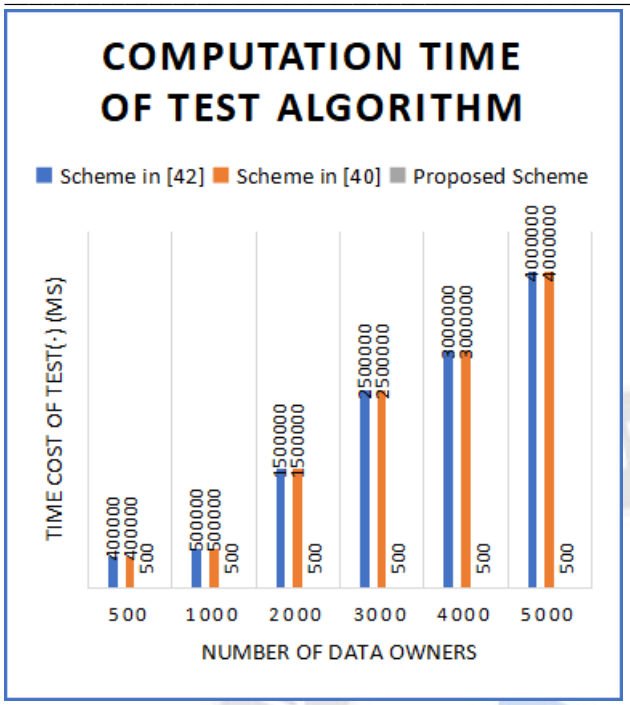

Figure 5: Computation time for test algorithm against number of data owners

The Test(.) operation of the proposed scheme is evaluated in term of execution time against number of data owners. As shown in Figure 5, for any number of data owners, the test operation carried out by CSP shows same time cost in case of the proposed scheme. However, the existing schemes need linear increase in the time cost as the number of data owners is increased. The reason behind this is that search operation in cloud server is done using single trapdoor in the proposed scheme while existing schemes need a trapdoor for each user. This they cause more time complexity as the number of data owners increase.

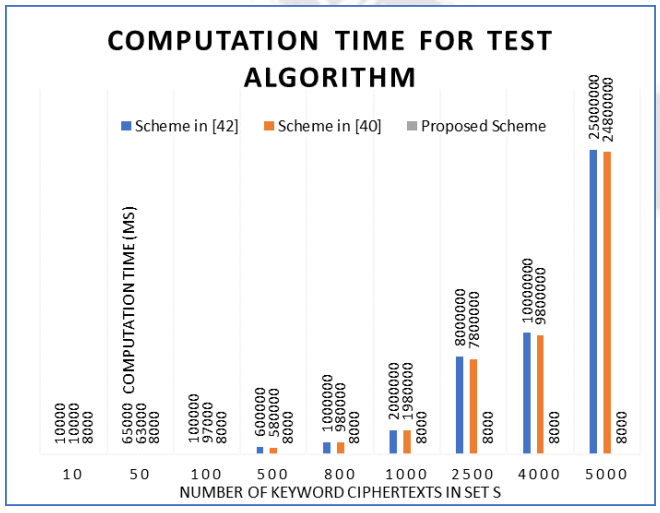

Figure 6: Computation time for test algorithm against number of keyword ciphertexts in set S

The Test(.) operation of the proposed scheme is evaluated in term of execution time against number of keyword cipher texts associated with a set of shared documents denoted as S. As shown in Figure 5, for any number of keyword cipher texts, the test operation carried out by CSP shows same time cost in case of the proposed scheme. However, the existing schemes need linear increase in the time cost as the number of data owners is increased. The reason behind this is that search operation in cloud server is done using single trapdoor in the proposed scheme while existing schemes need a trapdoor for each user. Thus they cause more time complexity as the number of keyword cipher texts increase.

VII. CONCLUSION AND FUTURE WORK

In this paper we proposed an efficient key aggregate searchable encryption scheme which enables multiple features such as support for truly single aggregate key to access data of many data owners, federated cloud support, query privacy, controlled search process and security against cross-pairing attack. This scheme is designed to be more flexible and efficient in serving user groups in cloud-assisted application scenarios. It has algorithms for setup, keygen, encrypt, extract, aggregate, trapdoor, test and federator. In multi-user setting it is designed to serve data owners and users with secure data sharing through key aggregate searchable encryption. The proposed scheme supports federated cloud. Experimental results revealed that the scheme is provably secure with relatively less computational overhead and time complexity when compared with the state of the art. In future, we consider an Internet of Things (IoT) use case and investigate on improving the proposed scheme to adapt to such environment.

REFERENCES

[1] Changyu Dong a, Giovanni Russello and Naranker Dulay. (2011). Shared and Searchable Encrypted Data for Untrusted Servers. Journal of Computer Security. 19(3), pp.367-397.

[2] J. Li, X. Chen, M. Li, J. Li, P. P. C. Lee, and W. Lou. (2014). Secure Deduplication with Efficient and Reliable Convergent Key Management. IEEE Trans. Parallel Distrib. Syst.. 25(6), pp.1615-1625.

[3] C. Gentry, C. Peikert, and V. Vaikuntanathan. (2008). Trapdoors for Hard Lattices and New Cryptographic Constructions. STOC, pp.1-41.

[4] X. Lu, W. Yin, Q. Wen, Z. Jin, and W. Li. (2018). A Lattice-Based Unordered Aggregate Signature Scheme Based on the Intersection Method. IEEE Access,. 6, p.33986–33994.

[5] P. V. Reddy and P. Gopal. (2015). Identity-based key-insulated aggregate signature scheme. J. King Saud Univ.-Comput. Inf. Sci.. 29, pp.1-8. Online].

[6] Baojiang Cui, Zheli Liu and Lingyu Wang. (2014). Key-Aggregate Searchable Encryption (KASE) for Group Data

Sharing via Cloud Storage. IEEE TRANSACTIONS ON COMPUTERS. 6(1), pp.1-13. Online].

[7] Xiaofeng Chen, Jin Li, Xinyi Huang, Jingwei Li, Yang Xiang, and Duncan S. Wong. (2013). Secure Outsourced Attribute-based Signatures. IEEE Transactions on Parallel and Distributed Systems. 65(8), pp.1-11.

[8] . Li, J. Li, X. Chen, C. Jia, and Z. Liu,. (2012). Efficient Keyword Search over Encrypted Data with Fine-Grained Access Control in Hybrid Cloud. 6th Int. Conf. Netw, pp.490-502.

[9] Mr. Dharmesh Dhabliya. (2012). Intelligent Banal type INS based Wassily chair (INSW). International Journal of New Practices in Management and Engineering, 1(01), 01 - 08. Retrieved from http://ijnpme.org/index.php/IJNPME/article/view/2

[10] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky,. (2011). Searchable symmetric encryption: Improved definitions and efficient constructions. Department of Computer Science, New Jersey Institute of Technology, pp.895-934.

[11] C.-K. Chu, S. S. M. Chow, W.-G. Tzeng, J. Zhou, and R. H. Deng. (2014). Key-Aggregate Cryptosystem for Scalable Data Sharing in Cloud Storage. IEEE Trans. Parallel Distrib.. 25(2), pp.468-477.

[12] C. Gu, Y. Guang, Y. Zhu, and Y. Zheng. (2013). Public Key Encryption with Keyword Search from Lattices. Int. J. Inf. Technol. 19(1), pp.1-10.

[13] Sharma, M. K. (2021). An Automated Ensemble-Based Classification Model for The Early Diagnosis of The Cancer Using a Machine Learning Approach. Machine Learning Applications in Engineering Education and Management, 1(1), 01–06. Retrieved from http://yashikajournals.com/index.php/mlaeem/article/view/1

[14] D. Boneh, C. Gentry, and B. Waters,. (2005). Collusion Resistant Broadcast Encryption With Short Ciphertexts and Private Keys. CRYPTO. 3621, pp.258-275.

[15] O. Regev. (2005). On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. J. ACM,. 56(6), pp.84-93.

[16] J. Li and K. Kim. (2010). Hidden attribute-based signatures without anonymity revocation. Department of Computer Science, Korea Advanced Institute of Science and Technology. 180(9), pp.1681-1689.

[17] C. Gentry, A. O'Neill, and L. Reyzin. (2018). A Unified Framework for Trapdoor-Permutation-Based Sequential Aggregate Signatures. PKC PTII, pp.1-17.

[18] D. Boneh, C. Gentry, and M. Hamburg. (2007). Space-Efficient Identity Based Encryption Without Pairings. FOCS. , p.647–657.

[19] J. Alwen and C. Peikert. (2010). Generating Shorter Bases for Hard Random Lattices. Theory Comput. Syst. 48(.), pp.535-553.

[20] H. Du, Q. Wen, and S. Zhang. (2018). An Efficient Certificateless Aggregate Signature Scheme without Pairings for Healthcare Wireless Sensor Network. IEEE Access. 7, pp.1-14.

[21] Peter W. Shor. (1999). Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer∗. AMS subject classifications. 41(2), pp.303-332.

[22] Z. Liu, Z. Wang, X. Cheng, C. Jia, and K. Yuan,. (2013). "Multi-user searchable encryption with coarser-grained access control in hybrid cloud,". 4th Int. Conf. Emerg. Intell. Data Web Technol, pp.249-255.

[23] M. Noroozi and Z. Eslami,. (2019). Public-key encryption with keyword search: a generic construction secure against online and offline keyword guessing a. J. Ambient Intell. Humanized Comput, pp.1-12.

[24] X. Zhang, Y. Tang, H. Wang, C. Xu, Y. Miao, and H. Cheng. (2019). Lattice-based proxy-oriented identity-based encryption with keyword search for cloud storage. Inf. Sci. 494, pp.193-207.

[25] Dhabliya, P. D. . (2020). Multispectral Image Analysis Using Feature Extraction with Classification for Agricultural Crop Cultivation Based On 4G Wireless IOT Networks. Research Journal of Computer Systems and Engineering, 1(1), 01–05. Retrieved from https://technicaljournals.org/RJCSE/index.php/journal/article/view/10

[26] D. Wu, X. Wang, and Q. Gan. (2016). Public Key Encryption with Keyword Search from Lattices in Multiuser Environments. Math. Problems Eng, pp.1-8.

[27] R. A. Popa and N. Zeldovich. (2019). Multi-Key Searchable Encryption. Cryptol. ePrint Arch., Tech, pp.1-18.

[28] Xiaojun Zhanga,b,c, Huaxiong Wangb and Chunxiang Xu. (2018). Identity-based key-exposure resilient cloud storage public auditing scheme from lattices. Information Sciences, pp.1-29.

[29] Smith, J., Jones, D., Martinez, J., Perez, A., & Silva, D. Enhancing Engineering Education through Machine Learning: A Case Study. Kuwait Journal of Machine Learning, 1(1). Retrieved from http://kuwaitjournals.com/index.php/kjml/article/view/86

[30] X. Zhang, C. Xu, H. Wang, Y. Zhang, and S. Wang. (2021). "FS-PEKS: Lattice based forward secure public-key encryption with keyword search for cloud-assisted industrial Inte. IEEE Trans. Dependable Secure Comput., to be published, pp.1-16.

[31] F. Zhao, T. Nishide, and K. Sakurai. (2012). Multi-User Keyword Search Scheme for Secure Data Sharing with Fine-Grained Access Control. Int. Conf. Inf. Secur. Cryptol, pp.406-418. Online].

[32] Muhammad Khan, Machine Learning for Predictive Maintenance in Manufacturing: A Case Study , Machine Learning Applications Conference Proceedings, Vol 1 2021.

[33] S. Agrawal, D. Boneh, and X. Boyen,. (2010). Lattice Basis Delegation in Fixed Dimension and Shorter-Ciphertext Hierarchical IBE. CRYPTO, pp.98-115.

[34] YANQING YAO, ZHENGDE ZHAI , JIANWEI LIU , AND ZHOUJUN L. (2019). Lattice-Based Key-Aggregate (Searchable) Encryption in Cloud Storage. J. Softw. 7, pp.164544-164555.

[35] YANQING YAO, Zhengde Zhai , Jianwei Liu , Zhoujun Li. (2016). Lattice-based Key-Aggregate (Searchable) Encryption in Cloud Storage. IEEE Access. 4, pp.1-13.

[36] S.Vinothkumar, J.Amutharaj and Jeyabalan. (2021). A COMPREHENSIVE STUDY OF CRYPTOGRAPHY AND

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6964
Article Received: 28 March 2023 Revised: 18 May 2023 Accepted: 01 June 2023

KEY MANAGEMENT BASED SECURITY IN CLOUD COMPUTING. 1Department of CSE, ACS College of Engineering, Bengaluru,. 27(3), pp.1909-1923.

[37] Mahnaz Noroozi , Ziba Eslami. (2019). Public-key encryption with keyword search. Ambient Intelligence and Humanized Computing , pp.1-12. Online].

[38] Rui Araújo and António Pinto. (2021). Secure Remote Storage of Logs with Search Capabilities. J. Cybersecur. Priv, pp.340-364. Online].

[39] YANQING YAO, Zhengde Zhai, Jianwei Liu and , Zhoujun Li 3. (2016). Lattice-Based Key-Aggregate (Searchable) Encryption in Cloud Storage. IEEE Access. 4, pp.1-13.

[40] Rajat Chaudhary, Anish Jindal, Gagangeet Singh Aujla, Neeraj Kumar, Ashok Kumar. (2018). LSCSH: Lattice-Based Secure Cryptosystem for Smart Healthcare in Smart Cities Environment. IEEE Communications Magazine , pp.24-32.

[41] Shanthi D.L, Sahana .P, Abishek .P, Shilpa A.S, Juhi Kumari . (2016). A Study on the Security Issues , Algorithms and Schemes towards a more secure Cloud Storage. International Journal of Scientific & Engineering Research. 7(3), pp.729-735. Online].

[42] Chiara Marcolla, Victor Sucasas and Marc Manzano, Riccardo Bassoli. (2022). Survey on Fully Homomorphic Encryption, Theory and Applications. IEEE, pp.1-40.

[43] Jelizaveta Vakarjuk, Nikita Snetkov and Jan Willemson. (2021). DiLizium: A Two-Party Lattice-Based Signature Scheme. Entropy, pp.1-30. Online].

[44] A. Boorghanya S. Bayat-Sarmadib , and R. Jalili. (2018). Practical provably-secure authenticated encryption schemes using lattice-based pseudorandom function SPRING. Scientia Iranica. 25(6), pp.3442-3460.

[45] Tong Li, Zheli Liu, Ping Li, Chunfu Jia, Zoe L Jiang, and Jin Li. Verifiable searchable encryption with aggregate keys for data sharing in outsourcing storage. In Australasian Conference on Information Security and Privacy, pages 153–169. Springer, 2016.

[46] Haijiang Wang, Xiaolei Dong, Zhenfu Cao, Dongmei Li, and Nanyuan Cao. Secure key-aggregation authorized searchable encryption. Science China Information Sciences, 62(3):39111, 2019.

[47] Cui BJ, Liu ZL, Wang LY, 2016. Key-aggregate searchable encryption (KASE) for group data sharing via cloud storage. IEEE Trans Comput, 65(8):2374-2385. https://doi.org/10.1109/TC.2015.2389959