






Article

A Secure Key Aggregate Searchable Encryption with Multi Delegation in Cloud Data Sharing Service

JoonYoung Lee ¹, MyeongHyun Kim ¹, JiHyeon Oh ¹, YoungHo Park ^{1,2,*}, KiSung Park ³
and Sungkee Noh ³

¹ School of Electronic and Electrical Engineering, Kyungpook National University, Daegu 41566, Korea; harry250@knu.ac.kr (J.L.); kimmyeong123@knu.ac.kr (M.K.); chldlstr071@knu.ac.kr (J.O.)

² School of Electronics Engineering, Kyungpook National University, Daegu 41566, Korea

³ Blockchain Technology Research Center, Electronics and Telecommunications Research Institute, Daejeon 34129, Korea; ks.park@etri.re.kr (K.P.); sknoh@etri.re.kr (S.N.)

* Correspondence: parkyh@knu.ac.kr; Tel.: +82-53-950-7842

Abstract: As the amount of data generated in various distributed environments is rapidly increasing, cloud servers and computing technologies are attracting considerable attention. However, the cloud server has privacy issues, including personal information and requires the help of a Trusted Third Party (TTP) for data sharing. However, because the amount of data generated and value increases, the data owner who produces data must become the subject of data sharing. In this study, we use key aggregate searchable encryption (KASE) technology, which enables keyword search, to efficiently share data without using TTP. The traditional KASE scheme approach only discusses delegation of authority from the data owner to another user. However, if the delegated entity cannot perform time-critical tasks because the shared data are unavailable, the delegate must further delegate the rights given to other users. Consequently, this paper proposes a new KASE scheme that enables multi-delegation without TTP and includes an authentication technique between the user and the server. After that, we perform informal and formal analysis using BAN logic and AVISPA for security evaluation, and compare the security and performance aspects with existing schemes.

Keywords: KASE; cloud server; data sharing; delegation; BAN logic; AVISPA; MIRACL



Citation: Lee, J.; Kim, M.; Oh, J.; Park, Y.; Park, K.; Noh, S. A Secure Key Aggregate Searchable Encryption with Delegation in Cloud Data Sharing Service. *Appl. Sci.* **2021**, *11*, 8841. <https://doi.org/10.3390/app11198841>

Academic Editor: Charalampos S. Kouzinopoulos

Received: 20 August 2021

Accepted: 22 September 2021

Published: 23 September 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

As a hyper-connected world is realized due to the development of the Internet, data production is increasing in various distributed environments such as medical care, finance, and vehicles. As per a study published by Statista Research Department [1], the total amount of data generated per year is expected to reach 149ZB by 2024 as the amount of data generated worldwide increases exponentially. The generated data can be used as an input for financial, medical, and artificial intelligence development, and cloud storage and computing technologies have been introduced to manage vast amounts of data [2–4]. Cloud computing services provide large-capacity storage and computing resources to resource-constrained computing devices.

However, privacy issues arise because the generated data includes personal information. Research and policies are being developed worldwide to protect the privacy of such data. “Midata” in the UK [5] and “Smart disclosure” in the US are policies for individuals to use and protect personal information as subjects, and have been implemented to date. However, these policies are being implemented with the help of a Trusted Third Party (TTP) because it is difficult to provide services based on personal information. Because the amount and value of the data generated increases, the data owner who produces the data should be the data sharer, and not the TTP.

For the subject of data to manage data without the help of TTP, the following are considered: (i) Key management for data access control must also be performed by the

data owner (DO). (ii) The efficiency of the key for data management should be considered. This is because as the data increases, the key also increases. (iii) The data owner must store the data in an encrypted form in order to maintain the confidentiality and integrity of the data, and the data access policy is required for sharing with data users (DUs).

DO outsources data or computational work to cloud servers. In addition, DOs can optionally share outsourced data with DU groups with the help of cloud computing services through access control. For this purpose, research on cryptosystems such as user-based access control cryptosystem [6], role-based access control cryptosystem [7], and attribute-based access control cryptosystem [8] was studied. However, the computation overhead of encryption and key generation increases with the number of attributes or users in these user-centric data sharing methods. When DO grants a new user access to their data, the DO must either generate a new ciphertext or modify the ciphertext stored in the cloud. Furthermore, because the TTP defines and manages the user's rules and attributes, the DO cannot be the subject of data management. To address these limitations, a data-centric shared encryption scheme called Key Aggregate Encryption (KAE) [9] has been proposed. In KAE, the DO first defines the document set S to which the data that the DO intends to share to users of data belongs, and then aggregates the secret keys of all documents in the set S . Then, DO shares a single key, known as the aggregate key, with the user to grant access to S . Moreover, an extended encryption scheme called Key Aggregate Searchable Encryption (KASE) [10] was proposed, which allows DOs to use aggregation keys to delegate search authority over selected data sets and allow users to retrieve shared data by submitting a single aggregation trapdoor to the cloud. In addition to delegating search rights to data users by data owners, it is also important to consider when the delegated users may need to transfer rights to other users for time-sensitive tasks, processing and creation of various information, and smooth data management.

However, there is no KASE structure given that an authorized user needs to further delegate privileges to other users. Therefore, this paper proposes a KASE cloud data sharing scheme that simultaneously provides user authentication and delegation functions without using TTP. To analyze the safety verification of the proposed scheme, we conduct informal security analysis and formal security analysis using "Burrows-Abadi-Needham (BAN) logic" [11] and "Automated Validation of Internet Security Protocols and Applications (AVISPA)" [12]. We use the "Multiprecision Integer and Rational Arithmetic Cryptographic Library (MIRACL)" [13] to build a test bed and calculate the cost of cryptographic operations. Finally, we compare security and performance with other existing schemes.

1.1. Motivation

Existing KASE schemes only discuss delegating data access rights to other DUs by DO. However, there are cases where the delegated user needs to delegate further the delegated rights to another user because the shared data are unavailable to the delegated user:

- The first is a case in which time-critical work such as immediate life-threatening, bodily harm, and property benefits of the DO is required. For example, in the event of an emergency involving the DO's life and body, the DU must delegate the authority to another DU when the DU who has been authorized to access information such as the DO's health information management is absent.
- In addition, it is necessary to use various information and generate revenue through information sharing. A DU who has received information rights can use the information to make a profit. However, it is difficult to expect visible revenue generation from a single user. In this case, the DU can create new services and revenue by delegating limited access rights to other DUs.
- DUs authorized to provide services by data owners often struggle to manage large amounts of data. In this case, the DU should be able to perform load balancing by assigning limited administrative privileges to some users.

Therefore, we propose a KASE data sharing scheme that can delegate access rights for these cases.

1.2. Contribution

Our proposed scheme is an access control for DOs to share data with DUs without the help of TTP. We also consider cases where DUs may delegate limited data access rights to other users, which is not considered by existing KAEs and KASEs. The detailed contributions of our proposed scheme are as follows:

- *Group data sharing with a keyword search:* In the proposed scheme, DOs can delegate access to and retrieval of data in an encrypted state for data sets requested by DUs. Additionally, each ciphertext in the shared set can be retrieved as a trapdoor of constant size generated using the aggregate key. Furthermore, the proposed system can confirm whether keywords exist in the data set to be searched using a bloom filter [14].
- *Multi-access prevention and privacy preservation:* The authentication of the proposed scheme prevents unauthorized DUs from accessing the trapdoor multiple times. In particular, if an unauthenticated user attempts to intercept and submit a trapdoor, the system prevents it. The identity submitted for authentication is a pseudonym identity which is masked and sent to protect the privacy of the DU. Moreover, keyword ciphertext, the hidden access policy defined by DO, and trapdoor does not disclose information about related keywords.
- *Fine-grained delegation:* In the proposed scheme, the DO provides authentication credentials to delegators and delegates. When a delegator wants to delegate authority, it authenticates with the delegate through the authentication credential. If authentication is valid, delegates can delegate the rights they have received to another users in fine-grained manner.

The rest of the paper is organized as follows: The related works are given in Section 2. In Section 2, we briefly describe studies on KAE and KASE that have been studied. We also provide preliminaries for the proposed scheme. The system model and threat model are defined in Section 3.

2. Related Works

In this section, we review the literature regarding the previously studied of KAE and KASE. This section also provides preliminaries about cryptology concepts that we use throughout the paper.

2.1. Literature Reviews

In 2016, Chu et al. [9] proposed the notion of KAE scheme that can reduce the number of distributed data encryption keys for data sharing system environments. The KAE allows documents or data sets encrypted with different keys to be decrypted with a single aggregate key. In 2018, Guo et al. [15] proposed a scheme for sharing encrypted data with other users through public cloud storage. Their approach involves an authentication process, and they argue that the authentication process can solve the key leak problem of data sharing. However, Alimohmmadi et al. [16] proved that Guo et al.'s scheme does not have security against impersonation and forging authentication key attacks. They demonstrated that the proposed Guo et al.'s scheme could allow anyone to forge an authentication key and access an arbitrary set of files stored in the cloud. Therefore, they proposed a new KASE scheme to solve the problems of Guo et al.

However, since there was no search function for keywords in documents at [9,15,16], Cui et al. [10] proposed a KASE scheme that enables group keyword search in the existing KAE. The scheme of [10], which first proposed the KASE method, provides the searchable group data sharing function, i.e., all users can selectively share selected groups of users and selected groups of files, the latter can perform keyword searches against the former. Unfortunately, Zhou et al. [17] proved that Cui et al.'s scheme is insecure against insider

attack. They demonstrated that the adversary can guess the valid user's key with the insider attacker. Furthermore, Cui et al.'s scheme [10] did not support searching over multi-owner data using a single key of constant size.

To address this problem, Li et al. [18] proposed the scheme for searching over multi-owner's data using a single trapdoor. Their scheme allows verification of search results using an aggregate key. They also offered advance planning in multi-owner settings.

Zhou et al. [17] proposed a KASE scheme of data-centric framework in an Industrial Internet of Things (IIoT) environment. Sensors in IIoT do not support the computational power of pairing operations as their hardware resources are very limited. Therefore, Zhou et al. proposed a KASE scheme that does not use the pairing operation in the encryption phase.

Padhya et al. [19] also proposed a KASE scheme for multi-owner data. Padhya et al.'s scheme is a practical way to generate keyword ciphertext without the use of expensive pairing operations given resource-constrained environments. They also discussed scenarios for federated clouds and proposed methods for delegating search authority when data are stored in federated clouds.

Liu et al. [20] proposed a scheme to validate keyword search results using a single aggregation key. The KASE method of Liu et al. also provides user authentication. In their scheme, the cloud server can verify the legitimacy of a sub-user by verifying that the authorized user's identity set includes the sub-user's identity. However, Li et al.'s protocol is insecure against user impersonation attacks.

In addition to delegating the searchable authority to the user, it is also necessary to consider the case where the delegated user must delegate the authority to another user for time-sensitive tasks, processing and creation of various information, and managing a large amount of data. However, there are many KASE schemes dealing with data sharing between DOs and DUs, but none dealing with cases where DU delegates to other DUs without help of the TTP. Furthermore, no KASE scheme works out the problem of user authentication and fine-grained multi-delegation at the same time. Authentication is one of the basic security services absolutely necessary to provide secure services in various network environments [21–28].

2.2. Preliminaries

In this section, we briefly discuss the cryptographic concepts used in this paper: bilinear map and bloom filter.

2.2.1. Bilinear Map

Pairing is a bilinear map defined for a subgroup of elliptic curves. Assume that G_1 and G_2 are two multiplicative circular elliptic curve subgroups of the same prime order p . A mapping $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear map if it satisfies the following [29]:

1. Bilinearity: For all $u, v \in G_1$, and $x, y \in \mathbb{Z}_q^*$, we have $e(u^x, v^y) = e(u, v^{xy})$.
2. Non-degeneracy: If u and v are generators of G_1 , $e(u, v) \neq 1$.
3. Computability: there is an efficient algorithm to compute $e(u, v)$ for any $u, v \in G_1$.

2.2.2. Bloom Filter

An m -bit bloom filter [14] can be viewed as an array of m bits, all initialized to zero. For verification in bloom filter, k independent hash functions H_1, \dots, H_k with the ranges $\{0, \dots, m-1\}$ is designed. During the generation phase, each element $s \in S = \{s_1, s_2, \dots, s_n\}$ and each $H_j(s)$ -bit in the array is set to 1, where $1 \leq j \leq k$. The value of $H_j(s)$ bit can be determined in the verification phase whether the elements s belongs to S . If the value is 0 then it must be $s \notin S$, otherwise it is highly probable that it is $s \in S$. Assuming the hash function is completely random, the false positive rate is $(1 - (1 - \frac{1}{m})^{kn})^k \approx (1 - e^{-kn/m})^k$. The $k = \frac{(\ln 2)m}{n}$ hash function leads to a minimum false positive rate $\frac{(0.6185)^m}{n}$. Two algorithms are included in the m -bit bloom filter.

1. *BFGen*: *BFGen* hashes the data set S to $\{H_1, \dots, H_k\}$ for producing an m -bit bloom filter.
2. *BFVerify*: *BFVerify* returns 0 if $s \notin S$ and 1 otherwise.

3. System Model and Threat Model

In this section, we describe the system model of our proposed scheme and provide threat model and notations that we use throughout the paper.

3.1. System Model

Our proposed system model is represented in Figure 1 and has three entities:

- **Data Owner (DO)**: *DO* is an entity that independently manages data as an owner of data and information without TTP. When data are requested from *DU*, *DO* encrypts data and related keywords and stores them in the cloud server, delivering a single aggregate key of a fixed size. *DO* encrypts the group identity *GID* for delegation of authority of delegatee and delegator to define delegation of authority.
- **Data User (DU)**: *DU* receives aggregate key when requesting data from the user. *DU* generates a trap door to retrieve data from *CS* using aggregate key and keyword, receives encrypted data through authentication with *CS*, and then decrypts to receive data.
- **Cloud Server (CS)**: Since *CS* is an honest but curious entity, it may legitimately try to learn all the information from a received message. *CS* provides *DO* with storage and computing power. In addition, *CS* searches data through the trapdoor received from *DU* and performs keyword verification.

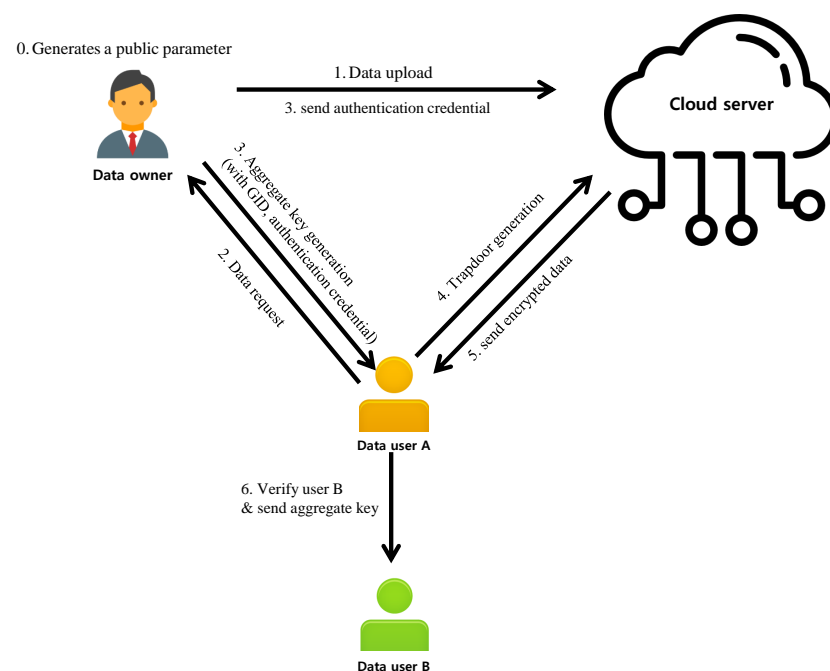


Figure 1. Proposed system model.

DO creates public parameters to be used in the system and publishes them to entities. Then, *DO* creates a bloom filter for keyword verification, encrypts data, and uploads it to *CS*. *DU* sends a data request to *DO*, and *DO* returns a single aggregate key for the data received from *DU*, an authentication credential for authentication with *CS*, and a *GID* for verifying authorization. The authentication credential is delivered it to *CS* at this time. Subsequently, *DU* creates a single trapdoor using aggregate and keywords from *CS* and requests a search query. After *CS* authenticates with *DU*, *CS* searches the data and confirms the keyword using the trap door. After that, *CS* generates a data search result and

proof set for decryption and sends it to *DU*. *DU* uses the bloom filter to decrypt the data after verification. In addition, if a *DU* wants to delegate authority to another *DU*, mutual authentication is performed. If the authentication is valid, *DU* can delegate the aggregate key and some of the keywords he/she has.

3.2. Threat Model

In this paper, we adopt the universally accepted Dolev-Yao (DY) threat model [30] for security analysis of the proposed scheme. In accordance with the DY model, an attacker is able to seize transmitted messages through an open channel, and eavesdrop, delete, inject or modify on the seized messages.

- The attacker has full control over and learns from messages sent over open channels. The attacker can then insert, modify, or remove valid messages.
- Because guessing more than one value at a time is a “computationally infeasible operation”, the attacker can only guess one value in polynomial time.

In addition, this paper additionally adopts the assumptions of the “Canetti and Krawczyk model (CK model)” [31]. It is a more powerful threat model compared to the DY model and is considered the de facto standard for modeling key exchange protocols.

3.3. Notations

Table 1 specifies the symbols used in this paper.

Table 1. Notations used in this paper.

Notations	Meanings
DU_j, ID_j	j th data user and their identity, respectively,
DO	Data owner
CS	Cloud server
HID_j	The hidden identity of j th data user
G_1, G_2	Bilinear groups
PK_{do}	The DO 's public key for encrypt data
DPK_{do}	The DO 's public key for authentication
r_{do}, ρ_{do}	Master secret key and secret key of DO
PK_j, PK_{cs}	The public key of data user and cloud server, respectively
$GID_1, HGID_1$	The group identity defined by data owner and its hidden identity
T_1, T_2, T_3	Current timestamps
ΔT	Maximum transmission delay
R_{du}, R_{cs}, r_A, r_B	Random nonces
W	The keyword
CK	The encrypted keyword
Tr	The trapdoor
$ $	Data concatenation operator
h_1, h_2	The hash function $\{0, 1\}^* \rightarrow Z_q$
h_2	The map-to-point hash function $\{0, 1\}^* \rightarrow G_1$,
\oplus	Bitwise exclusive-or operator

4. Our Proposed Scheme

We propose a key aggregate scheme for multi-delegation and authentication without TTP in this section. The proposed scheme consists of six phases, namely setup phase, data upload phase, aggregation key generation phase, trapdoor generation and retrieve phase, authentication for delegation, and group identity revocation phase.

4.1. Setup Phase

For data sharing and upload data, a data owner *DO* have to generate bilinear map and public system parameters. *DO* also generates hash functions for encrypted information and bloom filter. The detailed steps of the setup phase are summarized in Figure 2 and discussed below.

Step 1: *DO* generates a bilinear map $B = (q, G_1, G_2, e)$, where q is the order of G_1 and $e : G_1 \times G_1 \rightarrow G_2$. G_1 and G_2 are multiplicative elliptic curve groups. Then *DO* picks random generator $g \in G_1$ and random nonce $\alpha \in Z_q$, and computes $g_i = G^{(\alpha_i)}$, where $1 \leq i \leq 2n$.

Step 2: After that, *DO* chooses his/his master secret key $r_{do} \in Z_q^*$, secret key $\rho_{do} \in Z_q^*$. *DO* generates hash functions $h_1 : \{0, 1\}^* \rightarrow Z_q$ and $h_2 : \{0, 1\}^* \rightarrow G_1$ for hashing information. Furthermore, *DO* also generates k independent universal hash functions $\{H'_1, \dots, H'_k\}$ which are used to set up a m -bit bloom filter.

Step 3: Then, *DO* computes public key $PK_{do} = g^{r_{do}}$ for encrypting data and public key $DPK_{do} = g^{\rho_{do}}$ for authentication. At last, *DO* publishes $B, (g, g_1, \dots, g_n), DPK_{do}, PK_{do}, h_1, h_2$ and $\{H'_1, \dots, H'_k\}$.

Setup Phase
Data owner (<i>DO</i>)
Generates bilinear map $B = (q, G_1, G_2, e)$ Pick a random generator $g \in G_2$ and a random nonce $\alpha \in Z_q$ and computes $g_i = g^{(\alpha_i)}, 1 \leq i \leq 2n$ Then chooses master secret key $r_{do} \in Z_q^*$ and chooses secret key $\rho_{do} \in Z_q^*$ Then generate hash functions $h_1 : \{0, 1\}^* \rightarrow Z_q,$ $h_2 : \{0, 1\}^* \rightarrow G_1,$ and k independent universal hash $\{H'_1, \dots, H'_k\}$ Then computes, for encrypt data public key $PK_{do} = g^{r_{do}}$ for authentication public key $DPK_{do} = g^{\rho_{do}}$ Then <i>DO</i> publishes $B, (g, g_1, \dots, g_n), DPK_{do}, PK_{do}, h_1, h_2, \{H'_1, \dots, H'_k\}$

Figure 2. Setup phase.

4.2. Data Upload Phase

In this phase, *DO* encrypts the data and uploads it to the cloud server. At this time, *DO* creates a bloom filter to verify whether the keyword is included in the document set. *DO* encrypts the keyword set CK_i , generates a public auxiliary value ∇_i for index, and sends them to the cloud server. This phase is briefed in Figure 3 and detailed steps are given below.

Step 1: First, *DO* picks a random number $t \in Z_q$ as the actual searchable encryption key and generates a bloom filter for keyword set W_i , where $i \in \{1, \dots, n\}$ is file index. The bloom filter is computed as $BF_i = BFGen(\{H'_1, \dots, H'_k\}, W_i)$.

Step 2: Then, *DO* randomly chooses a $M \in G_2$ and computes a public auxiliary value ∇_i for index i . The ∇_i comprises c_1, c_2, c_3 and c_4 . They are computed as $c_1 = g^t, c_2 = (g_i \cdot PK_{do})^t, c_3 = h_2(M) \oplus BF_i,$ and $c_4 = M \cdot e(g_1, g_n)^t$. Then, *DO* computes $CK_i = \frac{e(g, h_1(w))^t}{e(g_1, g_n)^t}$ for each keyword w in this set's keyword set W_i .

Step 3: At last, *DO* sends ∇_i, CK_i to the cloud server.

Data Upload Phase	
Data owner (<i>DO</i>)	Cloud server (<i>CS</i>)
Pick a random number $t \in Z_q$. Generate a bloom filter for keyword set W_i $BF_i = BFGen(\{H'_1, \dots, H'_k\}, W_i)$ Choose a random $M \in G_2$ Compute $\nabla_i = (c_1, c_2, c_3, c_4)$ for index $i \in \{1, \dots, n\}$ $c_1 = g^t$ $c_2 = (g_i \cdot PK_{do})^t$ $c_3 = h_2(M) \oplus BF_i$ $c_4 = M \cdot e(g_1, g_n)^t$ $CK_i = \frac{e(g, h_1(w))^t}{e(g_1, g_n)^t}$ $\langle \nabla_i, CK_i \rangle$	

Figure 3. Data upload phase.

4.3. Data Request Phase

If DU_j wants to data set S_i , DU_j calculates HID_j and PK_j and requests data from *DO*. *DO* computes the aggregate key k_s corresponding data set. After that, *DO* creates GID_l by defining groups that can delegate or receive authority. *DO* can manage the list of *DUs* belonging to GID_l when a new DU_j is added or an existing DU wants to leave the group. After that, *DO* creates an authentication credential that allows DU_j and *CS* to authenticate each other. *DO* generates TID_j and transmits it with $k_s, HGID_l$ securely to DU_j , and generates and transmits HID_j and A_{cs} securely to *CS*. *CS* uses this value to calculate ACS_j , which is for the authentication credential, and stores it in its own database. Figure 4 summarizes this phase. The detailed steps involved in this phase are given below.

Step 1: DU_j generates a secret key $b_j \in Z_q^*$ and chooses an unique identity ID_j . Then, DU_j computes pseudo identity $HID_j = h_1(ID_j || b_j)$ and public key $PK_j = g^{b_j}$. DU_j sends $\langle HID_j, PK_j, S_i \rangle$ securely to *DO*, where S_i is a document set.

Step 2: After receiving data request from DU_j , *DO* generates an aggregate key $k_s = \prod_{j \in S} g^{r_{do}^{1-j}}$ which is corresponding document set S_i . *DO* then creates GID_l by defining groups to determine which users can delegate or receive privileges from each other. *DO* computes $TID_j = (DPK_{do})^{HID_j \cdot \rho_{do}}$ for authentication credential. Furthermore, *DO* computes $HGID_j = h_2(GID_l || r_{do} || \rho_{do})$ and $A_{cs} = h_2(r_{do} || \rho_{do})$. After that, *DO* sends $\langle k_s, TID_j, HGID_l \rangle$ to securely DU_j and sends $\langle HID_j, A_{cs} \rangle$ to securely *CS*.

Step 3: *CS* computes $ACS_i = h_2(HID_j || A_{cs})$ and public key $PK_{cs} = g^{A_{cs}}$ after receiving messages from *DO*. Then, *CS* stores ACS_i in *CS*'s database.

Data user (DU_j)	Data owner (DO)	Cloud server (CS)
Generate a secret key $b_j \in Z_q^*$ Choose a identity ID_j Compute $HID_j = h_1(ID_j b_j)$ $PK_j = g^{b_j}$ Data request S_i $\langle HID_j, PK_j, S_i \rangle$ $\xrightarrow{\hspace{1cm}}$	Generate a agrreate key k_s $k_s = \prod_{j \in S} g_{n+1-j}^{r_{do}}$ Define GID_j Compute $TID_j = (DPK_{do})^{HID_j \cdot \rho_{do}}$ $HGID_j = h_2(GID_j r_{do} \rho_{do})$ $A_{cs} = h_2(r_{do} \rho_{do})$ $\langle k_s, TID_j, HGID_j \rangle$ $\xleftarrow{\hspace{1cm}}$ $\langle HID_j, A_{cs} \rangle$ $\xrightarrow{\hspace{1cm}}$	Compute $ACS_j = h_1(HID_j A_{cs})$ $PK_{cs} = g^{A_{cs}}$ Store ACS_j in the database

Figure 4. Data request phase.

4.4. Data Retrieve Phase

DU_j generate a trapdoor of keyword w using their aggregate key. DU_j sends the trapdoor to CS for a search query and an authentication credential for mutual authentication. CS authenticates with DU_j , then CS determine whether the encrypted keyword is CK using DU_j 's trapdoor. After verification of keyword, CS generates a result set and proof set. After DU_j receives result set and proof set from CS , DU_j authenticate with CS and conducts the verification proofs that the keyword exists in owner's document set. Figure 5 describes this phase, and the detailed steps are as follows.

Step 1: DU_j generates a single aggregate trapdoor $Tr_j = k_s \cdot h_1(w)$. A trapdoor relates to a set of all documents related to the aggregate key. Then, DU_j generates timestamp T_1 and random nonce R_{du} . Furthermore, DU_j computes $V_j = PK_j^{R_{du}}$, $Verif_j = PK_{cs}^{b_j \cdot R_{du}}$, $M_j = h_1(ID_j || b_j) \oplus Verif_j$, $MA_j = h_1(Verif_j || HID_j || T_1)$, and $HHID_j = TID_j^{MA_j}$. After that, DU_j sends $M_j, V_j, HHID_j, T_1, Tr_j, S_i$ via an insecure channel.

Step 2: After receiving messages from DU_j , CS computes $Verif_j = V_j^{A_{cs}}$ and $HID'_j = Verif_j \oplus M_j$. Furthermore, CS checks if $h_1(HID'_j || A_{cs}) = ACS_j$. If it is valid, CS computes $MA'_j = h_1(Verif_j || HID'_j)$ and checks if $e(HHID_j, PK_{cs}) = e(DPK_{do}^{HID_j \cdot A_{cs}}, DPK_{do}^{MA_j})$. If it is valid, CS computes as follows for index i : $pub_1 = \pi_{z \in S, z \neq i} g_{n+1-z+i}$, $Tr_i = TR_j \cdot pub_1$, $pub_2 = \pi_{z \in S} g_{n+1-z}$, and $p_1 = c_4 \cdot \frac{e(pub_1, c_1)}{e(pub_2, c_2)}$. Then, CS checks $ck = \frac{e(Tr_i, c_1)}{e(pub_2, c_2)}$, where encrypted keyword is $ck \in CK_i$. CS adds the identity of results which is corresponding document to $Result_i$. Furthermore, CS sets $PRF_i = (c_1, p_1, c_3)$. Then, CS generates a random nonce R_{cs} and computes $VA_{cs} = PK_{cs}^{R_{cs}}$, $Verif_{cs} = PK_j^{A_{cs} \cdot R_{cs}}$, $AUTH_{cs} = h_1(MA_j || Verif_j || Verif_{cs})$. CS sends set $Result, PRF, VA_{cs}$ and $AUTH_{cs}$ over an open channel to DU_j .

Step 3: After receiving sets from CS, DU_j computes $Verif_{cs} = VA_{cs}^{b_j}$. Then, DU_j checks if $AUTH'_{cs} = h_1(MA_j || Verif_j || Verif_{cs})$. If it is valid, DU_j computes for each i as follows: $M' = p_1 \cdot e(k_s, c_1)$, $BF'_i = h_1(M') \oplus c_3$, $ACC_i = BFverfiy(\{H'_1, \dots, H'_k\}, BF'_i, W)$. If the keyword w exists in the document, $ACC_i = 1$. Otherwise, $ACC_i = 0$.

Data user (DU_j)	Cloud server (CS)
Generate a trapdoor $Tr_j = k_s \cdot h_1(w)$, where w is a keyword over appreciates document set Generate Timestamp T_1 and random nonce R_{du} Compute $V_j = PK_j^{R_{du}}$ $Verif_j = PK_{cs}^{b_j \cdot R_{du}}$ $M_j = h_1(ID_j b_j) \oplus Verif_j$ $MA_j = h_1(Verif_j HID_j T_1)$ $HHID_j = TID_j^{MA_j}$ $\underline{M_j, V_j, HHID_j, T_1, Tr_j, S_i}$	Compute $Verif_j = V_j^{A_{cs}}$ $HID'_j = Verif_j \oplus M_j$ Check $h_1(HID'_j A_{cs}) = ACS_i?$ If it is valid, then compute $MA'_j = h_1(Verif_j HID'_j T_1)$ Check $e(HHID_j, PK_{cs}) = e(DPK_{do}^{HID_j \cdot A_{cs}}, DPK_{do}^{MA_j})?$ Compute for index i $pub_1 = \prod_{z \in S, z \neq i} \delta_{n+1-z+i}$ $Tr_i = TR_j * pub_1$ $pub_2 = \prod_{z \in S} \delta_{n+1-z}$ $p_1 = c_4 \cdot \frac{e(pub_1, c_1)}{e(pub_2, c_2)}$ Check $ck = \frac{e(Tr_i, c_1)}{e(pub_2, c_2)}$ for encrypted keyword $ck \in CK_i$ Add the identity of the corresponding document to $Result_i$ Set $PRF_i = (c_1, p_1, c_3)$ Generate a random nonce R_{cs} Compute $VA_{cs} = PK_{cs}^{R_{cs}}$ $Verif_{cs} = PK_j^{A_{cs} \cdot R_{cs}}$ $AUTH_{cs} = h_1(MA_j Verif_j Verif_{cs})$ $\underline{Result, PRF, VA_{cs}, AUTH_{cs}}$
Compute $Verif_{cs} = VA_{cs}^{b_j}$ Check $AUTH'_{cs} = h_1(MA_j Verif_j Verif_{cs})?$ Compute for each i $M' = p_1 \cdot e(k_s, c_1)$ $BF'_i = h_1(M') \oplus c_3$ $ACC_i = BFverfiy(\{H'_1, \dots, H'_k\}, BF'_i, W)$	

Figure 5. Data retrieve phase.

4.5. Authentication for Delegation Phase

If DU_A wants to delegate their aggregate key, DU_A and DU_B conduct mutual authentication using $HGID_I$. If they have same $HGID_I$, they compute the same session key SK . After that, DU_A can send their own aggregate key and keyword using SK . In this case, DU_A can delegate limited access rights by sending only some of the keywords which DU_A have. The detailed steps are illustrated in Figure 6 and are as follows.

Step 1: DU_A generates a random nonce r_A and timestamp T_2 . Then, DU_A computes $R_A = PK_A^{r_A}$, $V_A = PK_B^{b_a \cdot r_A}$, and $L_{AB} = h_1(V_A || T_2)$. DU_A sends R_A, L_{AB}, T_2 to DU_B .

Step 2: After receiving messages from DU_A , DU_B computes $V_A = R_A^{b_b}$, and checks if $L_{AB} = h_1(V_A || T_2)$. If it is valid, DU_B generates a random nonce r_B and computes $R_B = PK_B^{r_B}$, $V_B = PK_A^{b_b \cdot r_B}$, $L_{BA} = h_1(V_A || HGID_I || T_3 || V_B)$, $AGID_I = h_1(V_A || HGID_I)$, and session key $SK = h_1(V_A || V_B || AGID_I || T_3)$. After that, DU_B sends R_B, L_{BA}, T_3 to DU_A .

Step 3: DU_A computes $V_B = R_B^{b_a}$ and checks if $L_{BA} = h_1(V_A || HGID_I || T_3 || V_B)$. If it is same value, DU_A computes the session key SK . At the end, DU_A and DU_B authenticate each other and compute the same SK for their secure communication.

Data user (DU_A)	Data user (DU_B)
Generate a random nonce r_A and timestamp T_2 Compute $R_A = PK_A^{r_A}$ $V_A = PK_B^{b_a \cdot r_A}$ $L_{AB} = h_1(V_A T_2)$ $\underline{R_A, L_{AB}, T_2}$	$V_A = R_A^{b_b}$ Check $L_{AB} = h_1(V_A T_2)?$ If valid, Generate a random nonce r_B Then, compute $R_B = PK_B^{r_B}$ $V_B = PK_A^{b_b \cdot r_B}$ $L_{BA} = h_1(V_A HGID_I T_3 V_B)$ $AGID_I = h_1(V_A HGID_I)$ $SK = h_1(V_A V_B AGID_I T_3)$ $\underline{R_B, L_{BA}, T_3}$
Compute $V_B = R_B^{b_a}$ Check $L_{BA} = h_1(V_A HGID_I T_3 V_B)?$ $SK = h_1(V_A V_B AGID_I T_3)$	

Figure 6. Authentication for delegation phase.

4.6. Group Identity Revocation Phase

When DU_j wants to leave the group, DO updates the group ID list to which DU_j belongs. DO updates GID with GID^{new} and issues new $HGID^{new}$ calculated as GID^{new} to data users corresponding to the existing GID list to send in bulk.

5. Security Analysis

In this phase, we present the non-mathematical (informal) security analysis and formal security analysis. We use broadly accepted “BAN logic” to show that the proposed scheme

can provide the mutual authentication and use “Automated Validation of Internet Security Protocols and Applications (AVISPA) simulation tool” for proving of security protocols from man-in-the-middle and replay attacks.

5.1. Informal Analysis

We conduct the informal analysis to analyze security capabilities and the security against various attacks.

5.1.1. Correctness

The DU_j should obtain the bloom filter by decrypting the corresponding ciphertext of the i -th document with the aggregation key. For correctness, the M can be obtained by:

$$\begin{aligned}
 p_1 \cdot e(k_s, c_1) &= c_4 \cdot \frac{e(pub_1, c_1)}{e(pub_2, c_2)} \cdot e(k_s, c_1) \\
 &= c_4 \cdot \frac{e(k_s \cdot pub_1, c_1)}{e(pub_2, c_2)} \\
 &= c_4 \cdot \frac{e(k_s \cdot \prod_{z \in s, z \neq i} g_{n+1-z+i}, g^t)}{e(\prod_{z \in s} g_{n+1-z}, (g_i \cdot PK_{do})^t)} \\
 &= \frac{c_4 \cdot e(k_s, g^t) \cdot e(\prod_{z \in s, z \neq i} g_{n+1-z+i}, g^t)}{e(\prod_{z \in s} g_{n+1-z}, g^{r_{do} \cdot t}) \cdot e(\prod_{z \in s} g_{n+1-z}, g_i^t)} \\
 &= \frac{c_4 \cdot e(k_s, g^t)}{e(\prod_{z \in s} g_{n+1-z}, g^{r_{do} \cdot t}) \cdot e(g_{n+1}, g^t)} \\
 &= \frac{M \cdot e(g_1, g_n)^t \cdot e(\prod_{z \in s} g_{n+1-z}^{r_{do}}, g^t)}{e(\prod_{z \in s} g_{n+1-z}, g^{r_{do} \cdot t}) \cdot e(g_1, g_n^t)} \\
 &= M
 \end{aligned}$$

5.1.2. Impersonation Attacks

If an adversary attempts to impersonate a legitimate DU , the adversary must be able to compute the legitimate message $M_j, V_j, HHID_j, T_1, Tr_j, S_i$. However, the attacker cannot compute $HHID_j$ because TID_j is computed using secret identity HID_j . Furthermore, CS checks $e(HHID_j, PK_{cs}) = e(DPK_{do}^{HID_j \cdot A_{cs}}, DPK_{do}^{MA_j})$. If it is not valid, then the impersonation attack is aborted by CS . Therefore, our proposed scheme can protect data user impersonation attacks.

5.1.3. Data User Anonymity

The real identity ID_j of DU_j is calculated as the pseudo identity HID_j , which depends on the random secret key b_j . In addition, the data user is provided with TID_j to be used for authentication in the data retrieve phase from DO . Since TID_j is dependent on the DO 's secret key rho_{do} , the attacker cannot know ID_j , which is the real identity of the data user. Therefore, we can say that we guarantee the anonymity of data users.

5.1.4. Perfect Forward Secrecy

In the data retrieve phase, suppose that an adversary obtains secret key A_{cs} of the cloud server. Then, the adversary is able to compute $Verif_j$ and HID_j . However, the adversary cannot compute VA_{cs} and $AUTH_{cs}$ since the adversary cannot know a random nonce R_{cs} . Thus, the data retrieve phase provides perfect forward secrecy. In the authentication for delegation phase, suppose that an adversary obtains secret key b_a or b_b of DU_A or DU_B . The adversary cannot compute session key SK because the adversary cannot compute V_A or V_B , which is dependent on random nonces r_A and r_B . Therefore, our proposed scheme ensures perfect forward secrecy.

5.1.5. Privileged-Insider Attacks

If an adversary is a privileged insider, the adversary is able to obtain HID_j and A_{cs} during the data request phase. Then, the attacker can compute $Verif_j$ and MA_j . However, CS generates a random nonce R_{cs} in the data retrieve session, and the adversary cannot compute $Verif_{cs} = PK_j^{A_{cs} \cdot R_{cs}}$ without R_{cs} . Therefore, the proposed scheme is secure against the privileged-insider attacks.

5.1.6. Replay and Man-In-The-Middle Attacks

An adversary can learn about transmitted messages over open wireless channels according to Section 3.2. However, in our proposed scheme, the adversary cannot conduct replay and man-in-the-middle attacks because every transmitted message contains timestamp or random nonce. Timestamps or random nonces T_1, T_2, T_3, R_{cs} , and r_A are generated by DU_j or CS and included in the message $MA_j = h_1(Verif_j || HID_j || T_1)$, $AUTH_{cs} = h_1(MA_j || Verif_j || R_{cs})$, $R_A = PK_a^{r_A}$, $L_{AB} = h_1(V_A || T_2)$, and $L_{BA} = h_1(V_A || HGID_I || T_3)$. Therefore, the proposed scheme can successfully prevent against replay and man-in-the-middle attacks.

5.1.7. Known Session-Specific Temporary Information Attacks

If an adversary obtains random numbers r_A and r_B according to CK-threat model mentioned in Section 3.2 in authentication for delegation phase, then the attacker can compute R_A or R_B . However, the adversary cannot compute V_A or V_B without obtaining data user's secret key b_a or b_b . Therefore, the attacker cannot compute $SK = h_1(V_A || V_B || AGID_I || T_3)$. Thus, we can say that our proposed scheme can prevent against the known session-specific temporary information attacks.

5.1.8. Ephemeral Secret Leakage (ESL) Attacks

In the authentication for delegation phase, DU_A and DU_B establish the same session key $SK = h_1(V_A || V_B || AGID_I || T_3)$. Based on the CK-threat model Section 3.2, the short term ephemeral secrets r_A, r_B can be leaked. However, the adversary still cannot compute SK because the adversary does not have b_a and b_b . Furthermore, assuming that the long-term secret keys b_a and b_b have been leaked, the adversary cannot calculate the session key because r_A and r_B cannot be known. SK can be computed only when both short term and long term are leaked, and since this is a computationally infeasible problem, our scheme can resist ESL attacks.

5.1.9. Session Key Disclosure Attacks

An adversary tries to obtain sensitive information by calculating a legitimate session key SK . However, as discussed in Sections 5.1.4, 5.1.7 and 5.1.8, the adversary cannot compute SK because of the computationally infeasible problem. Therefore, our proposed scheme is safe against session key disclosure attacks.

5.1.10. Mutual Authentication

After receiving the message from DU_j in the data retrieve phase, CS checks

$$\begin{aligned} e(DPK_{do}^{HID_j \cdot A_{cs}}, DPK_{do}^{MA_j}) &= e(DPK_{do}^{HID_j \cdot A_{cs}}, DPK_{do}^{MA_j}) \\ &= e(g^{\rho_{do} \cdot HID_j \cdot A_{cs}}, g^{\rho_{do} \cdot MA_j}) \\ &= e(g, g)^{\rho_{do} \cdot HID_j \cdot A_{cs} \cdot \rho_{do} \cdot MA_j} \\ &= e(g^{\rho_{do} \cdot HID_j \cdot \rho_{do} \cdot MA_j}, g^{A_{cs}}) \\ &= e(HHID_j, PK_{cs}) \end{aligned}$$

According to Sections 5.1.2 and 5.1.3, an adversary cannot impersonate legitimate DU_j . Moreover, DU_j also checks $AUTH'_{cs} = h_1(MA_j || Verif_j || Verif_{cs})$.

In addition, in the authentication for delegation phase, DU_A and DU_B check $L_{AB} = h_1(V_A || T_2)$ and $L_{BA} = h_1(V_A || HGID_I || T_3 || V_B)$. Therefore, our scheme provides mutual authentication.

5.2. BAN Logic Analysis

This section uses BAN logic [11] to prove that the proposed scheme provides mutual authentication in the data retrieve phase and authentication for delegation phase. Table 2 provides a description of the notation of BAN logic and we also describe the rules, goals, assumptions and ideal form of ban logic [32,33].

Table 2. The basic BAN logic notations.

Notations	Meaning
SK	The used session key in current authentication session
$\#ST$	The statement ST is fresh
$\omega \triangleleft ST$	ω sees the statement ST
$\omega \equiv ST$	ω believes the statement ST
$\omega \sim ST$	ω once said ST
$\langle ST \rangle_{For}$	Formula SR is united with formula For
$\{ST\}_{Key}$	Encrypt the formula ST encrypted the key Key
$\omega \stackrel{Key}{\leftrightarrow} \sigma$	ω and σ uses Key as shared key for communicating
$\omega \Rightarrow ST$	ω controls the statement ST

5.2.1. Logical Rules of BAN Logic

The Logical rules of the BANlogic are:

1. Jurisdiction rule :

$$\frac{\omega | \equiv \sigma | \implies S, \quad \omega | \equiv \sigma | \equiv ST}{\omega | \equiv S}$$

2. Nonce verification rule :

$$\frac{\omega | \equiv \#(ST), \quad \omega | \equiv \sigma | \sim ST}{\omega | \equiv \sigma | \equiv ST}$$

3. Message meaning rule :

$$\frac{\omega | \equiv \omega \stackrel{K}{\leftrightarrow} \sigma, \quad \sigma \triangleleft \{S\}_K}{\omega | \equiv B | \sim ST}$$

4. Belief rule :

$$\frac{\omega | \equiv (ST, For)}{\omega | \equiv ST}$$

5. Freshness rule :

$$\frac{\omega | \equiv \#(ST)}{\omega | \equiv \#(ST, For)}$$

5.2.2. Goals for Data Retrieve Phase

The following goals are presented to demonstrate that the proposed scheme achieves a mutual authentication :

Goal 1: $CS| \equiv (R_{du}),$

Goal 2: $CS| \equiv DU_j| \equiv (R_{du}),$

Goal 3: $DU_j| \equiv (R_{cs}),$

Goal 4: $DU_j| \equiv CS| \equiv (R_{cs}),$

5.2.3. Idealized Forms for Data Retrieve Phase

The idealized forms are as following :

$M_1 : DU_j \rightarrow CS : (HID_j, T_1, R_{du}, DPK_{do})_{g^{b_j \cdot A_{cs}}}$

$M_2 : CS \rightarrow DU_j : (HID_j, T_1, R_{cs})_{g^{b_j \cdot A_{cs}}}$

5.2.4. Assumptions for Data Retrieve Phase

The following assumptions are the initial state of the proposed scheme to achieve BAN logic proof.

$A_1 : CS| \equiv (CS \xrightarrow{g^{b_j \cdot A_{cs}}} DU_j)$

$A_2 : DU_j| \equiv (DU_j \xrightarrow{g^{b_j \cdot A_{cs}}} CS)$

$A_3 : CS| \equiv \#(R_{du})$

$A_4 : DU_j| \equiv \#(R_{cs})$

$A_5 : CS| \equiv DU_j \Rightarrow (R_{du})$

$A_6 : DU_j| \equiv CS \Rightarrow (R_{cs})$

5.2.5. Proof Using BAN Logic for Data Retrieve Phase

Main proofs using rules and assumptions of the BAN logic are as the following steps :

Step 1: S_1 can be obtained from M_1

$$S_1 : CS \triangleleft (HID_j, T_1, R_{du}, DPK_{do})_{g^{b_j \cdot A_{cs}}}.$$

Step 2: For obtaining S_2 , we apply the message meaning rule with A_1

$$S_2 : CS| \equiv DU_j| \sim (HID_j, T_1, R_{du}, DPK_{do}).$$

Step 3: For obtaining S_3 , we apply the freshness rule with A_3

$$S_3 : CS| \equiv \#(HID_j, T_1, R_{du}, DPK_{do}).$$

Step 4: For obtaining S_4 , we apply the nonce verification rule with S_2 and S_3

$$S_4 : CS| \equiv DU_j| \equiv (HID_j, T_1, R_{du}, DPK_{do}).$$

Step 5: For obtaining S_5 , we apply the belief rule

$$S_5 : CS | \equiv DU_j | \equiv (R_{du}). \text{ (Goal 2)}$$

Step 6: For obtaining S_6 , we apply the jurisdiction rule with A_5

$$S_6 : CS | \equiv R_{du}. \text{ (Goal 1)}$$

Step 7: S_7 can be obtained from M_2

$$S_7 : DU_j \triangleleft (HID_j, T_1, R_{cs})_g^{b_j, A_{cs}}.$$

Step 8: For obtaining S_8 , we apply the message meaning rule with A_2

$$S_8 : DU_j | \equiv CS | \sim (HID_j, T_1, R_{cs}).$$

Step 9: For obtaining S_9 , we apply the freshness rule with A_4

$$S_9 : DU_j | \equiv \#(HID_j, T_1, R_{cs}).$$

Step 10: For obtaining S_4 , we apply the nonce verification rule with S_8 and S_9

$$S_{10} : DU_j | \equiv CS | \equiv (HID_j, T_1, R_{cs}).$$

Step 11: For obtaining S_{11} , we apply the belief rule

$$S_{11} : DU_j | \equiv CS | \equiv (R_{cs}). \text{ (Goal 4)}$$

Step 6: For obtaining S_{12} , we apply the jurisdiction rule with A_6

$$S_{12} : DU_j | \equiv R_{cs}. \text{ (Goal 3)}$$

Thus, our scheme has completed the proof that it provides mutual authentication for data retrieve phase. BAN logic proof of authentication for delegation phase is similar to the above proof. Therefore, our scheme can provide secure mutual authentication.

5.3. AVISPA Simulation Analysis

We adopt the “Automated Validation of Internet Security Protocols and Applications (AVISPA) Simulation Tools” [12] to perform validation of security protocols against replay and man-in-the-middle attacks. AVISPA includes four backends: “Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP)”, “SAT-based Model Checker (SATMC)”, “Constraint-logic-based Attack Searcher (CL-AtSe)”, and “On-the-fly mode-checker (OFMC)”. Neither the SATMC nor the TA4SP backends currently support “bitwise exclusive OR (XOR)” operations. Therefore, official security validation-based simulations rely on two backends: CL-AtSe and OFMC.

We use “High-Level Protocol Specification Language (HLPSL)” to implement the proposed scheme for the primary roles of data owner DO, data user DU, and cloud server CS, and also mandatory “Sessions and Goals and Environments”. It is worth noting that AVISPA uses the DY threat model for validation. Figure 7 provides simulation results of OFMC and CL-ATse backends in the data retrieve phase and authentication for delegation phase, and clearly shows that the proposed protocol is safe from “replay and man-in-the-middle attacks” [34,35].

<pre>% OFMC % Version of 2006/02/13 SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS TYPED_MODEL PROTOCOL /home/span/span/testsuite/results/aggkey_case1_2.if GOAL as_specified BACKEND OFMC COMMENTS STATISTICS parseTime: 0.00s searchTime: 5.42s visitedNodes: 1808 nodes depth: 9 plies</pre>	<pre>SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS TYPED_MODEL PROTOCOL /home/span/span/testsuite/results/aggkey_case1_2.if GOAL As Specified BACKEND CL-AtSe STATISTICS Analysed : 0 states Reachable : 0 states Translation: 0.06 seconds Computation: 0.00 seconds</pre>	<pre>% OFMC % Version of 2006/02/13 SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS TYPED_MODEL PROTOCOL /home/span/span/testsuite/results/aggkey_case2.if GOAL as_specified BACKEND OFMC COMMENTS STATISTICS parseTime: 0.00s searchTime: 8.71s visitedNodes: 4096 nodes depth: 12 plies</pre>	<pre>SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS TYPED_MODEL PROTOCOL /home/span/span/testsuite/results/aggkey_case2.if GOAL As Specified BACKEND CL-AtSe STATISTICS Analysed : 0 states Reachable : 0 states Translation: 0.08 seconds Computation: 0.00 seconds</pre>
--	---	---	---

Figure 7. (left) Results of data retrieve phase. (right) Results of authentication for delegation phase.

6. Security and Efficiency Features Comparison

We compare the proposed scheme with the existing competing schemes in the domain of KASE such as Cui et al. [10] and Liu et al. [20], in terms of security functions, computational and communication overhead.

6.1. Functionality and Security Features Comparison

We compare the proposed scheme with the existing competing scheme in terms of various security features, such as replay, man-in-the-middle, impersonation, privileged-insider, session key disclosure attacks. Moreover, we compare various functional aspects such as user anonymity, mutual authentication, multi-access and delegation. Table 3 shows that existing schemes do not meet all security requirements. Moreover, unlike existing schemes, our proposed scheme additionally provides multi-access and delegation functions, and it is worth noting that DO or DU can perform various functions without TTP assistance.

Table 3. Security Properties.

Security Properties	Cui et al. [10]	Liu et al. [20]	Ours
Man-in-the-middle attack	o	o	o
Replay attack	o	o	o
Impersonation attack	x	x	o
User anonymity	o	o	o
Privileged-insider attack	x	o	o
Session key disclosure attack	x	o	o
Mutual authentication	x	x	o
Multi-access	-	-	o
Multi-delegation	-	-	o

x: Insecure. o: Secure. -: Not concerned.

6.2. Comparison of Computation Costs

This section performs a testbed experiment on cryptographic computation of the data retrieve phase using the popular “Multiprecision Integer and Rational Arithmetic Cryptographic Library (MIRACL)” [13] on two platforms:

- Platform 1:** The platform 1 is general personal computer environment, and the detailed performance of the personal computer is as follows: “Ubuntu 18.04.4 LTS with memory 8 GiB, processor: Intel Core i7-4790 @ 3.60GHz × 4, CPU Architecture: 64-bit.” The experiments are executed for “one-way-hash-function (T_h)”, “Bilinear pairing operation (T_b)”, “Scalar point multiplication (T_{spm})”, and “Exponentiation operation (T_e)” for 100 runs. After that the average run-time in milliseconds are recorded for

these operations or functions from 100runs, which are 0.003 ms, 6.575 ms, 2.373 ms, and 0.819 ms, respectively.

- Platform 2:** The platform 2 is Raspberry Pi environment for considering mobile device, and the detailed performance of the Raspberry Pi is as follows: “Model: Raspberry PI 3 B, with CPU 64-bit, Processor: 1.2 GHz Quad-core, Memory: 1 GiB, and OS: Ubuntu 20.04.2 LTS 64-bit.” Figure 8 shows the setting of Raspberry Pi environment. The experiments are executed for “one-way-hash-function (T_h)”, “Bilinear pairing operation (T_b)”, “Scalar point multiplication (T_{spm})”, and “Exponentiation operation (T_e)” for 100 runs. After that the average run-time in milliseconds are recorded for these operations or functions from 100runs, which are 0.020 ms, 21.348 ms, 5.686 ms, and 2.973 ms, respectively.



Figure 8. Raspberry Pi Platform.

Table 4 reveals the message computation costs of data user and cloud server entities in the data retrieval phase. As a result of comparing Cui et al., Liu et al., and ours, respectively, it can be seen that our scheme has a higher total cost compared to the existing schemes. However, the proposed scheme has the strength of showing that it is safe against various attacks.

Table 4. Comparison of computation costs.

Protocol	Computation Cost			Total Cost
	Data User	Cloud Server	Personal Computer	Raspberry Pi
Cui et al. [10]	$1T_h + 1T_{spm}$	$(2n)T_{spm} + 2T_b$	$(4.746n + 15.526)$ ms	$(11.372n + 48.402)$ ms
Li et al. [20]	$2T_h + 2T_{spm} + 1T_b$	$(2n + 1)T_{spm} + 4T_b$	$(4.746n + 40)$ ms	$(11.372n + 123.838)$ ms
Ours	$5T_h + 3T_{spm} + 4T_e + 1T_b$	$3T_h + (2n + 2)T_{spm} + 3T_e + 6T_b$	$(4.746n + 63.647)$ ms	$(11.372n + 198.837)$ ms

6.3. Comparison of Computation and Communication Complexity

The number of keywords in the ciphertext and the number of keywords in the search query set affect computation and communication costs. In our scheme, the pairing operation between the user and the cloud server is additionally calculated compared to other schemes, but authentication is performed only once regardless of the number of keyword value. Therefore, according to (O) asymptotic notation, our scheme has the same computational and communication costs as other existing KASE schemes. A comparative analysis in Table 5 shows that the complexity of computational and communication costs for the different features of the proposed scheme are comparable to those of the other schemes.

Table 5. Comparison of complexity.

Protocol	Computation Cost			Communication Cost		
	Encryption	Trapdoor	Retrieve of CS	Aggregate Key	Trapdoor	Ciphertext
Cui et al. [10]	$O(KW P)$	$(O Q M)$	$O(Q P)$	$O(1)$	$O(Q)$	$O(KW)$
Li et al. [20]	$O(KW P)$	$(O Q M)$	$O(Q P)$	$O(1)$	$O(Q)$	$O(KW)$
Ours	$O(KW P)$	$(O Q M)$	$O(Q P)$	$O(1)$	$O(Q)$	$O(KW)$

$|KW|$: number of keywords with the ciphertext, $|Q|$: number of keywords in the query set, P : pairing.

6.4. Discussion of Comparison

We can see from a comparative analysis that the computation costs demonstrate that the proposed protocol is expensive compared to other schemes. As per the asymptotic notation, the proposed scheme's calculation complexity and communication consumption cost are the same as those of other schemes such as Cui et al. [10] and Liu et al. [20]. Furthermore, as shown in Table 3, our scheme outperforms other schemes in terms of security and features.

7. Conclusions and Future Works

In this paper, we designed a novel KASE scheme for data sharing without assistance of TTP, considering multi-delegation. The proposed scheme provides mutual authentication to secure data sharing. Moreover, our protocol can provide keyword verification through a bloom filter technique, and can resist various security attacks such as impersonation, privileged-insider and session key disclosure attacks. Moreover, our proposed scheme satisfies user anonymity property. We performed BAN logic to prove that the scheme can provide mutual authentication, and we also applied AVISPA simulation tool to demonstrate that the proposed scheme is secure from man-in-the-middle and replay attacks. Our scheme has higher computation cost compared to existing schemes, but the complexity according to the number of keywords and data sets is the same as existing schemes, proving that it is more secure than existing schemes. In the future, we will build a test-bed that simulates the real environment for efficient data sharing in real cloud services environment. After that, we will apply our scheme to the test-bed and improve it to a more efficient scheme.

Author Contributions: Conceptualization, J.L., K.P. and Y.P.; software, J.L.; validation, M.K. and J.O.; formal analysis, J.L., M.K. and J.O.; investigation, M.K. and J.O.; writing—original draft preparation, J.L.; writing—review and editing, K.P., S.N. and Y.P.; supervision, Y.P.; project administration, Y.P.; funding acquisition, K.P. and S.N. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Electronics and Telecommunications Research Institute(ETRI) grant funded by the Korean government. [21ZR1330, Core Technology Research on Trust Data Connectome].

Conflicts of Interest: The authors declare no conflict of interest.

References

- Holst, A.; Statista. Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2024. 2020. Available online: <https://www.statista.com/statistics/871513/worldwide-data-created/> (accessed on 30 January 2021).
- Kamara, S.; Lauter, K. Cryptographic Cloud Storage. In Proceedings of the International Conference on Financial Cryptography and Data Security, Tenerife, Spain, 25–28 January 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 136–149.
- Osanaiye, O.; Choo, K.K.R.; Dlodlo, M. Distributed denial of service (DDoS) resilience in cloud: Review and conceptual cloud DDoS mitigation framework. *J. Netw. Comput. Appl.* **2016**, *67*, 147–165. [CrossRef]
- Juliadotter, N.V.; Choo, K.K.R. Cloud attack and risk assessment taxonomy. *IEEE Cloud Comput.* **2015**, *2*, 14–20. [CrossRef]
- MiData. The Midata Project. Available online: <https://www.midata.coop> (accessed on 9 August 2021).
- Fiat, A.; Naor, M. Broadcast encryption. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 22–26 August 1993; pp. 480–491.

7. Ferrailol, D.F.; Kuhn, D.R. Role based access control national computer security conference. In Proceedings of the 15th National Computer Security Conference (NCSC), Baltimore, MD, USA, 13–16 October 1992; pp. 554–563.
8. Sahai, A.; Waters, B. Fuzzy identity-based encryption. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, 22–26 May 2005; Volume 3494, pp. 457–473.
9. Chu, C.K.; Chow, S.S.; Tzeng, W.G.; Zhou, J.; Deng, R.H. Key-aggregate cryptosystem for scalable data sharing in cloud storage. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 468–477.
10. Cui, B.; Liu, Z.; Wang, L. Key-aggregate searchable encryption (KASE) for group data sharing via cloud storage. *IEEE Trans. Comput.* **2016**, *65*, 2374–2385. [[CrossRef](#)]
11. Burrows, M.; Abadi, M.; Needham, R. A logic of authentication. *ACM Trans. Comput. Syst.* **1990**, *8*, 18–36. [[CrossRef](#)]
12. AVISPA. Automated Validation of Internet Security Protocols and Applications. Available online: <http://www.avispa-project.org/> (accessed on 9 August 2021).
13. MIRACL Cryptographic SDK: Multiprecision Integer and Rational Arithmetic Cryptographic Library. Available online: <https://github.com/miracl/MIRACL> (accessed on 9 August 2021).
14. Zheng, Q.; Xu, S.; Atenise, G. VABKS: Verifiable attribute-based keyword search over outsourced encrypted data. In Proceedings of the IEEE INFOCOM 2014—IEEE Conference on Computer Communications, Toronto, ON, Canada, 27 April–2 May 2014; pp. 522–530.
15. Guo, C.; Luo, N.; Bhuiyan, M.Z.A.; Jie, Y.; Chen, Y.; Feng, B.; Alam, M. Key-aggregate authentication cryptosystem for data sharing in dynamic cloud storage. *Future Gener. Comput. Syst.* **2018**, *84*, 190–199. [[CrossRef](#)]
16. Alimohammadi, K.; Bayat, M.; Javadi, H.H. A secure key-aggregate authentication cryptosystem for data sharing in dynamic cloud storage. *Multimedia Tools Appl.* **2020**, *79*, 2855–2872. [[CrossRef](#)]
17. Zhou, R.; Zhang, X.; Du, X. File-centric multikey aggregate keyword searchable encryption for industrial internet of things. *IEEE Trans. Ind. Inf.* **2018**, *14*, 3648–3658. [[CrossRef](#)]
18. Li, T.; Liu, Z.; Li, P. Verifiable searchable encryption with aggregate keys for data sharing in outsourcing storage. In Proceedings of the Australasian Conference on Information Security and Privacy, Melbourne, VIC, Australia, 4–6 July 2016; pp. 153–169.
19. Padhya, M.; Jinwala, D.C. MULKASE: A novel approach for key-aggregate searchable encryption for multi-owner data. *Front. Inf. Technol. Electron. Eng.* **2019**, *20*, 1717–1748. [[CrossRef](#)]
20. Liu, Z.; Li, T.; Li, P. Verifiable searchable encryption with aggregate keys for data sharing system. *Future Gener. Comput. Syst.* **2018**, *78*, 778–788. [[CrossRef](#)]
21. Yu, S.; Lee, J.; Lee, K.; Park, K.; Park, Y. Secure authentication protocol for wireless sensor networks in vehicular communications. *Sensors* **2018**, *18*, 3191. [[CrossRef](#)] [[PubMed](#)]
22. Yu, S.; Lee, J.; Park, K.; Das, A.K.; Park, Y. IoV-SMAP: Secure and efficient message authentication protocol for IoV in smart city environment. *IEEE Access* **2020**, *8*, 167875–167886. [[CrossRef](#)]
23. Kwon, D.; Yu, S.; Lee, J.; Son, S.; Park, Y. WSN-SLAP: Secure and lightweight mutual authentication protocol for wireless sensor networks. *Sensors* **2021**, *21*, 936. [[CrossRef](#)] [[PubMed](#)]
24. Oh, J.; Yu, S.; Lee, J.; Son, S.; Kim, M.; Park, Y. A secure and lightweight authentication protocol for IoT-based smart homes. *Sensors* **2021**, *21*, 1488. [[CrossRef](#)]
25. Lee, J.; Yu, S.; Park, K.; Park, Y.; Park, Y. Secure three-factor authentication protocol for multi-gateway IoT environments. *Sensors* **2019**, *19*, 2358. [[CrossRef](#)]
26. Park, K.; Noh, S.; Lee, H.; Das, A.K.; Kim, M.; Park, Y.; Wazid, M. LAKS-NVT: Provably secure and lightweight authentication and key agreement scheme without verification table in medical internet of things. *IEEE Access* **2020**, *8*, 119387–119404. [[CrossRef](#)]
27. Lee, J.; Kim, G.; Das, A.K.; Park, Y. Secure and Efficient Honey List-Based Authentication Protocol for Vehicular Ad Hoc Networks. *IEEE Trans. Netw. Sci. Eng.* **2021**, *8*, 2412–2425.
28. Wazid, M.; Bagga, P.; Das, A.K.; Shetty, S.; Rodrigues, J.J.; Park, Y. AKM-IoV: Authenticated key management protocol in fog computing-based Internet of vehicles deployment. *IEEE Internet Things J.* **2019**, *6*, 8804–8817. [[CrossRef](#)]
29. Boneh, D.; Franklin, M. Identity-based encryption from the Weil pairing. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2001; Springer: Berlin/Heidelberg, Germany, 2001; pp. 213–229.
30. Dolev, D.; Yao, A. On the security of public key protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198–208. [[CrossRef](#)]
31. Canetti, R.; Krawczyk, H. Universally composable notions of key exchange and secure channels. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, 17 April 2002; Springer: Berlin/Heidelberg, Germany, 2002; pp. 337–351.
32. Son, S.; Lee, J.; Kim, M.; Yu, S.; Das, A.K.; Park, Y. Design of secure authentication protocol for cloud-assisted telecare medical information system using blockchain. *IEEE Access* **2020**, *8*, 192177–192191. [[CrossRef](#)]
33. Park, K.; Park, Y.; Das, A.K.; Yu, S.; Lee, J.; Park, Y. A dynamic privacy-preserving key management protocol for V2G in social internet of things. *IEEE Access* **2019**, *7*, 76812–76832. [[CrossRef](#)]
34. Lee, J.; Yu, S.; Kim, M.; Park, Y.; Lee, S.; Chung, B. Secure key agreement and authentication protocol for message confirmation in vehicular cloud computing. *Appl. Sci.* **2020**, *10*, 6268. [[CrossRef](#)]
35. Kim, M.; Lee, J.; Park, K.; Park, Y.; Park, K.H.; Park, Y. Design of Secure Decentralized Car-Sharing System Using Blockchain. *IEEE Access* **2021**, *9*, 54796–54810. [[CrossRef](#)]