Journal of Organizational and End User Computing
Volume 32 • Issue 4 • October-December 2020

Secure Fine-Grained Keyword Search With Efficient User Revocation and Traitor Tracing in the Cloud

Mamta, National Institute of Technology, Kurukshetra, India

Brij B. Gupta, National Institute of Technology, Kurukshetra, India & Asia University, Taiwan & Macquarie University, Australia

ABSTRACT

Fine-grained searching is an important feature in multi-user cloud environment and a combination of attribute-based encryption (ABE) and searchable encryption (SE) is used to facilitate it. This combination provides a powerful tool where multiple data owners can share their data with multiple data users in an independent and differential manner. In this article, the authors have used key-policy design framework of attribute-based encryption to construct the multi-keyword search scheme where access rights assigned to a data user are associated with his/her secret key. This leads to a situation where a data user can abuse his secret key to distribute it illegally to the unauthorized users to perform search over the shared data which is not intended for him/her. Therefore, to track such kind of key abusers the authors have embedded an extra functionality of tracing the traitors. For this purpose, each user is assigned a unique identity in the form of binary string where each bit represents an attribute related to his identity. In addition to the normal attributes, the access structure of a user also possesses identity-related attributes which are hidden from the user along with some normal attributes. Hence, the proposed scheme supports partial anonymity. Further, in the event of user revocation the proposed scheme efficiently handles the system update process by delegating the computationally intensive tasks to the cloud server. Finally, the proposed scheme is proved secure under Decisional Bilinear Diffie-Hellman (DBDH) assumption and decision linear assumption in the selective security model.

KEYWORDS

Attribute-Based Encryption, Cloud Computing, Efficient User Revocation, Key Abuse, Multi-Keyword Search, Multi-User

INTRODUCTION

Cloud computing is one of the most promising technologies of the recent times as it has fundamentally changed the way we store and access our data. In cloud, the storage and management of data is delegated to a remote cloud server. This unburdens the user from the overhead of local storage and management of data and moreover, this stored data could be accessed anywhere anytime and on any device. Owing to these advantages more and more users are shifting towards cloud-based storage. But apart from these benefits there are some privacy concerns associated with the data stored over the cloud because the data is stored over a remote server which could not be fully trusted. One simple solution to this issue could be to store the data in an encrypted form. This definitely solve the issue of data privacy but will beget another problem (Gupta, 2016; Gupta, 2018). Searching operation is one

DOI: 10.4018/JOEUC.2020100106

of the most basic and essential operations and encryption of data will severely debilitate this basic operation. Hence, there arises a need for a technique which should be conducive for search operation and at the same time ensures the privacy of data. Secure searchable encryption is the answer for this need (Yu, 2018; Yu, 2018; Gupta, 2017; Subramaniyaswamy, 2017).

Searchable encryption (SE) scheme enables the cloud server to perform keyword search over encrypted data without disclosing any information about the keyword being searched (San Nicolas-Rocca, T., 2013). SE scheme can be developed using either symmetric key or asymmetric/public key cryptographic primitive. Between these two, public key setting is a more preferable choice as it solves the issue of complicated key sharing in symmetric key setting when there are multiple users in the system. Further, there are several choices available in public key setting like Identity Based Encryption (IBE), Attribute Based Encryption (ABE), Functional Encryption (FE), etc. In this paper, we have used ABE scheme and particularly the key-policy (KP) design framework to develop SE scheme as it provides fine-grained searching capability in multi-user setting. In KP-ABE, the access policy is embedded in the secret key of the user. Any authorized user can misuse his/her access rights by sharing his secret key with other users who are not supposed to have access to the information. Consider a database which contains digital media in an encrypted form and a user is provided access depending upon the subscription and the amount he paid. There is no way of tracing if the user who has got the subscription is not sharing his secret credentials with other users, which usually happens. To prevent such unauthorized searching and retrieval of information, we have added an extra functionality of tracing given by Yu et al. (2010). There are several key-policy attribute based keyword search schemes in the literature given by Zheng et al. (2014), Li et al. (2017), Ameri et al. (2018) and Mamta and Gupta (2019) but none of them has incorporated the feature of tracing the key abusers, which is the main contribution of this paper. Following are the key highlights of the proposed scheme:

- It provides protection against key abusers by incorporating extra ciphertext components which are used for tracing the identity of the traitors. The ciphertext used in the normal operation and in the tracing operation is indistinguishable under decisional linear assumption.
- It efficiently handles the event of user revocation by delegating the task of updating the secret key of remaining users to the cloud.
- The proposed scheme takes multi-valued attributes and also partially hides the access structure associated with the user.
- The proposed scheme performs multi-keyword search and supports monotonic predicate which consists of AND, OR and threshold gates. It uses the top-down approach for distributing the secret values to an access structure.
- The proposed scheme is proved secure against chosen keyword attack in selective security model under decisional Diffie-Hellman assumption.

Application Example

The proposed scheme suits well in the banking system where the data is stored over the remote cloud server in an encrypted form to maintain the privacy of customer's financial and personal information. In order to access the information every employee is assigned an access privilege as per their role. For example: To evaluate the customer's potential to return the loan amount, the loan officer needs to access the current financial status of the customer while a customer service representative's role is to assist the new customers in completing their paperwork and answering any queries related to bank's product and services. So, depending upon the role played by each employee their access rights differ and different employees can have different view of the information stored over the cloud server. As in the example mentioned above the loan officer can have the information about the current financial condition of a customer while a customer service representative do not require any such information about that customer. In such scenario, if any employee misuses his/her access privilege and shares his/her secret key with some other employee. This may result in unauthorized access to sensitive data. To prevent such kind of abuse there should be a way through which these abusers can be tracked. The proposed scheme aims to provide this facility by associating a unique identity with each employee and generating the tracing ciphertext for some keywords encrypted using identity related attributes of the suspected user. Now, the user is tricked to find the keyword contained in tracing ciphertext which has the identity of the suspicious user. If a user is able to search that keyword and if a mismatch is found between his identity and the suspected user's identity, then it means this user is using the secret key of other user to access the information which is not intended for him. Further, the proposed scheme supports the situation if any employee who resigns from the bank can no longer be able to have access to any information with his secret key, this is achieved by defining efficient procedure for user revocation using proxy re-encryption and lazy re-encryption techniques.

Apart from the introduction rest of the paper is organized as: Section 2 discusses the related work. In section 3, essential background needed to develop a searchable encryption scheme is discussed along with system definition, framework and security model of the proposed scheme. Section 4 gives the basic design and detailed construction of the proposed scheme with the proof of correctness of the proposed scheme. Next section analysis the proposed scheme in terms of security and compares the storage and computational cost with existing schemes. Finally, section 6 gives the concluding remarks and provides the future directions.

RELATED WORK

This section provides an overview of the existing attribute-based keyword search schemes and gives an analysis of the existing schemes in terms of the key features. Attribute based encryption (ABE) is a technique which enables fine-grained search in multi-user environment when it is used as an underlying technique to develop a searchable encryption scheme (Zheng et al., 2014; Sun et al., 2016). The concept of ABE was first introduced by Sahai and Waters (2005) in the form of fuzzy identity-based encryption where identity is viewed as a set of descriptive attributes. After that two variants of ABE were proposed, namely key-policy attribute-based encryption (KP-ABE) (Goyal et al., 2006) and ciphertext-policy attribute-based encryption (CP-ABE) (Bethencourt et al., 2007). In KP-ABE, the access policy is embedded in the secret key of the user and the attributes are associated with the ciphertext while in CP-ABE, the access policy is embedded in the ciphertext and the attributes are associated with the secret key of the user. These variants can be used to develop an attribute based searchable encryption (Zheng et al., 2014; Sun et al., 2016; Li et al., 2017; Hu et al., 2017; Wang et al., 2017; Qiu et al., 2017; Cui et al., 2018; Ameri et al., 2018; Chen et al., 2018; Chaudhari & Das, 2019; Yin et al., 2019; Cui et al., 2019; Mamta & Gupta, 2019) where the access policy is used to determine who can perform search like it was used to determine the decryption capabilities in an encryption scheme. Out of the several existing schemes in the literature Zheng et al., Li et al. and Ameri et al. have used the key policy design framework to develop attribute-based keyword search scheme. Zheng et al. (2014) proposed the first attribute-based keyword search based on tree access structure where they have used both variants of ABE. The main feature of their technique is verifiability of the search result returned by the cloud server and hence broke the assumption that the cloud is honest. Later, Li et al. (2017) proposed an attribute based searchable encryption scheme based on the key policy design framework where they tried to reduce the computational burden by outsourcing heavy computational tasks to the cloud server, although it reduces the computational cost but results in an increased communication cost between different parties. A new attribute-based keyword search scheme based on key policy design framework was proposed Ameri et al. (2018) where a new concept of temporary keyword search is introduced. The main focus of this paper was to improve the security by associating a time period with the search token and with this search token the cloud server can perform search for only those keyword's ciphertexts which were encrypted in that time interval. It was also based on key-policy design framework of ABE. Recently, Mamta & Gupta

(2019a) proposed a key policy attribute-based keyword search scheme where the main focus was to improve the efficiency of the scheme by making the size of secret key and the trapdoor independent of the number of attributes. Further, Mamta and Gupta (2019b) proposed a dynamic policy ABE scheme with constant size secret key using KP design framework and then transformed it to a multi-keyword search scheme which inherits all the features of the proposed dynamic policy ABE scheme. So, from time to time several KP attribute-based keyword search schemes have been proposed where the focus was on either improving the efficiency or the security. In this paper, the focus is to add more functionality in terms of user's secret key accountability where the identity of the secret key abusers can be disclosed. A detailed analysis of the existing schemes is given below in Table 1.

Table 1 compares the basic features of the proposed scheme with other schemes. As shown above in Table 1, the proposed scheme uses key-policy (KP) design framework of underlying ABE scheme and uses tree-based data structure which consists of AND, OR and threshold gate. In the proposed scheme some of the attributes assigned to the user are hidden, thus ensures partial anonymity. The proposed scheme is secure in the selective security model under decision linear and DBDH assumption. The proposed scheme supports user revocation and moreover, the system update task is delegated to the cloud server when a user is revoked from the system. In addition, the proposed scheme supports tracing of secret key abusers thus ensures secret key accountability.

PRELIMINARIES

In this section, first an introduction about the general mathematical entities involved in the proposed scheme is given and then it provides the basic definitions and models of the proposed scheme.

General Background

This section gives the necessary information about bilinear map, hardness assumptions on which the proposed scheme relies and the structure of access policy used in the proposed scheme.

Bilinear Map

Let G, G_T be the source and target cyclic groups of prime order p and g be the generator of the source group G. Let e be the symmetric bilinear map between G and $G_T, e : G \times G \rightarrow G_T$, which satisfies the following properties:

- $\forall l = g^x \in G, m = g^y \in G : e\left(g^x, g^y\right) = e\left(g, g\right)^{xy}$, where $x, y \in Z_p$.
- $e\left(g, g\right)$ is the generator of the target group, G_T, if g is the generator of source groups G.
- $\forall l, m \in G; e\left(l, m\right)$ is efficiently computable.

Security Assumptions

Decisional Bilinear Diffie-Hellman (DBDH) assumption: It is hard to distinguish the tuples of the form $\left(g^a, g^b, g^c, e\left(g, g\right)^{abc}\right)$ from the tuple of the form $\left(g^a, g^b, g^c, e\left(g, g\right)^{z}\right)$ where $a, b, c, z \in Z_p$ and $g \in G$.

Decision linear (DL) assumption: It is hard to distinguish the tuple of the form $\left(g, g^x, g^c, g^\alpha, g^{cx}, g^{\alpha y}, g^{x+y}\right)$ from the tuple of the form $\left(g, g^x, g^c, g^\alpha, g^{cx}, g^{\alpha y}Q\right)$, where $g, Q \in G$ and $x, y \in Z_p$.

Table 1. Analysis of the related works

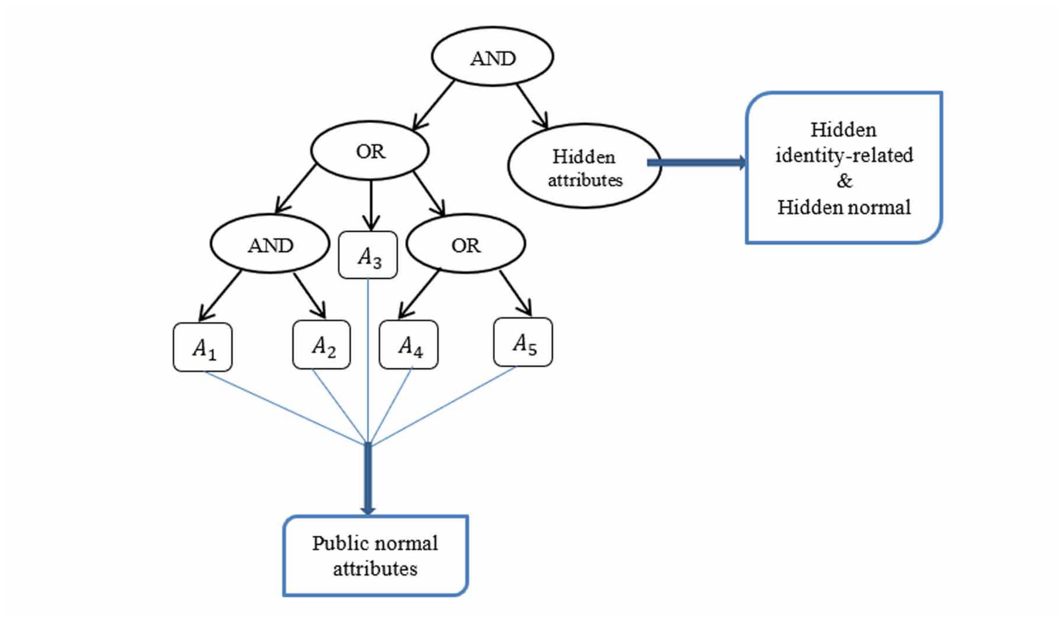
Scheme	Underlying Technique	Access Structure	Hidden Policy	Security Model & Assumption	Support for user revocation	Secret Key Accountability
Zheng et al. (2014)	KP-ABE CP-ABE	Tree	X	SS, Decision Linear	X	X
Sun et al. (2016)	CP-ABE	AND	X	SS, DBDH	√	X
Li et al. (2017)	KP-ABE	Tree	X	SS, DBDH	X	X
Hu et al. (2017)	CP-ABE	Tree	X	SS, Decision Linear	X	X
Wang et al. (2017)	CP-ABE	AND	Full	FS, generic group model	√	X
Qiu et al. (2017)	CP-ABE	AND	Full	SS, generic group model	X	X
Cui et al. (2018)	CP-ABE	Tree	X	DBDH	√	X
Chen et al. (2018)	CP-ABE	Tree	X	SS, DBDH	X	X
Ameri et al. (2018)	KP-ABE	Tree	X	SS, MDDH	X	X
Chaudhari and Das (2019)	CP-ABE	AND	Full	SS, DBDH	X	X
Yin et al. (2019)	CP-ABE	Tree	X	SS, DBDH	X	X
Cui et al. (2019)	CP-ABE KP-ABE	LSSS	X	-	X	X
Mamta & Gupta (2019a)	KP-ABE	Tree	X	SS, Decision Linear	√	X
Mamta & Gupta (2019b)	KP-ABE	Tree	X	SS, DBDH	X	X
Proposed Scheme	KP-ABE	Tree	Partial	SS, DBDH, Decision Linear	√	√

Notations used in Table 1:
CP-ABE: Ciphertext-Policy Attribute Based Encryption FS: Full Security SS: Selective Security
KP-ABE: Key-Policy Attribute Based Encryption
DDH: Decisional Diffie-Hellman
DBDH: Decisional Bilinear Diffie-Hellman Assumption
MDDH: Modified Decisional Diffie-Hellman
LSSS: Linear Secret Sharing Scheme

Access Policy

In this paper, the access policy is represented by the tree data structure as shown in Figure 1. Here, each leaf node represents an attribute and each non-leaf node represents a threshold gate. Let T_y be an access tree corresponding to a given access policy γ. Here, the type of the threshold gate is determined by the number of children, l_i, of a node i. A threshold value k_i is defined for each node i, such that $1 \leq k_i \leq l_i$. For an internal node, if $k_i = 1$, then it represents an OR gate and if $k_i = l_i$, then it represents an AND gate and for a leaf node, $k_i = 1$. It is assumed that access tree represents a subset of attribute universe *Att*. In the access tree, all the children of node are numbered in an ordered manner and let index(i) be a function which returns the number linked with node i. The type of attributes present at the leaf nodes can be the public normal attributes or the hidden attributes where each attribute can take multiple values. It is assumed that each attribute, A_j can have v_j number of possible values, $A_j = \left\{A_{j,1}, A_{j,2}, \ldots, A_{j,v_j}\right\}$. For the purpose of key management on the event of user revocation the root node of the access tree is always assumed to be an AND gate. The identity-related hidden attributes (HID) and the normal hidden attributes (HN) are assumed to be present at depth 1, while the public normal attributes (PN) are present in a subtree at depth 1. The structure of the access policy is shown below in Figure 1.

Figure 1. Structure of the access policy



Technique Preliminaries

This section provides the definition of the proposed scheme with the detailed framework and then defines the security model used for proving the security of the scheme.

System Definition

The system is composed of the following polynomial time algorithms:

1. $\left[pp, MSK\right] \leftarrow Setup\left(\lambda, Att, \mathcal{W}, n\right)$: This algorithm takes security parameter, attribute universe, keyword universe and a parameter *n* as input where *n* denotes that the first *n* elements in *Att* corresponds to the identity related attributes and outputs public parameters and master secret key.
2. $SK \leftarrow KeyGen\left(pp, MSK, \gamma\right)$: This algorithm assigns the secret key to the user by taking the public parameters, master secret key and access policy as the input. The access policy assigned to a user contains the normal attributes as well as the identity related attributes.
3. $C \leftarrow GenIndex\left(pp, \xi \subseteq Att, W \in \mathcal{W}\right)$: This algorithm encrypts the set of keywords, *W*, under a set of attributes $\xi = \xi_{PN} \cup \xi_{HN} \cup \xi_{HID}$ using the public parameters.
4. $TK \leftarrow GenTrap\left(pp, SK, W'\right)$: A legitimate user with secret key *SK* runs this algorithm to generate the trapdoor for a set of keywords, W'.

5. $\left[0/1\right] \leftarrow Search\left(pp, C, TK\right)$: The cloud server uses this algorithm to find whether the ciphertext and the trapdoor corresponds to same set of keywords or not. If so, it returns 1 otherwise it returns 0.

Correctness

The CP-ABSE scheme is correct if the following condition holds:

$$Search\left(PP, C, TK\right) = 1 \quad if\, W' = W \tag{1}$$

For the given $\left[pp, MSK\right] \leftarrow Setup\left(\lambda, Att, n\right)$, $C \leftarrow GenIndex\left(pp, \xi, W\right)$, $SK \leftarrow Keygen\left(pp, MSK, \gamma\right)$ and $TK \leftarrow GenTrap\left(pp, SK, W'\right)$.

System Model

The system is composed of the following entities as shown in Figure 2:

1. Trusted Third Party (TTP): is an entity that issues secret key credentials to the cloud users. It uses KeyGen algorithm and corresponding to the access right assigned to a user it generates the secret key for that user.
2. Data Owner: is an entity who wants to share his data with other users and for this purpose, the data owner encrypts his data and associate it with index of keywords which is also encrypted but this index can be searched by the authorized user. To generate this encrypted index data owner uses GenIndex algorithm.
3. Data User: is an entity who wants to retrieve the data stored by data owner and to do so the data user generates the trapdoor using GenTrap algorithm through which the data user can allow the cloud server to find the data which contains the keyword specified in the trapdoor.
4. Cloud server: is an entity that provides the storage facility to the data owner and performs the actual search operation on the behalf of data user based on the trapdoor provided by the data user.

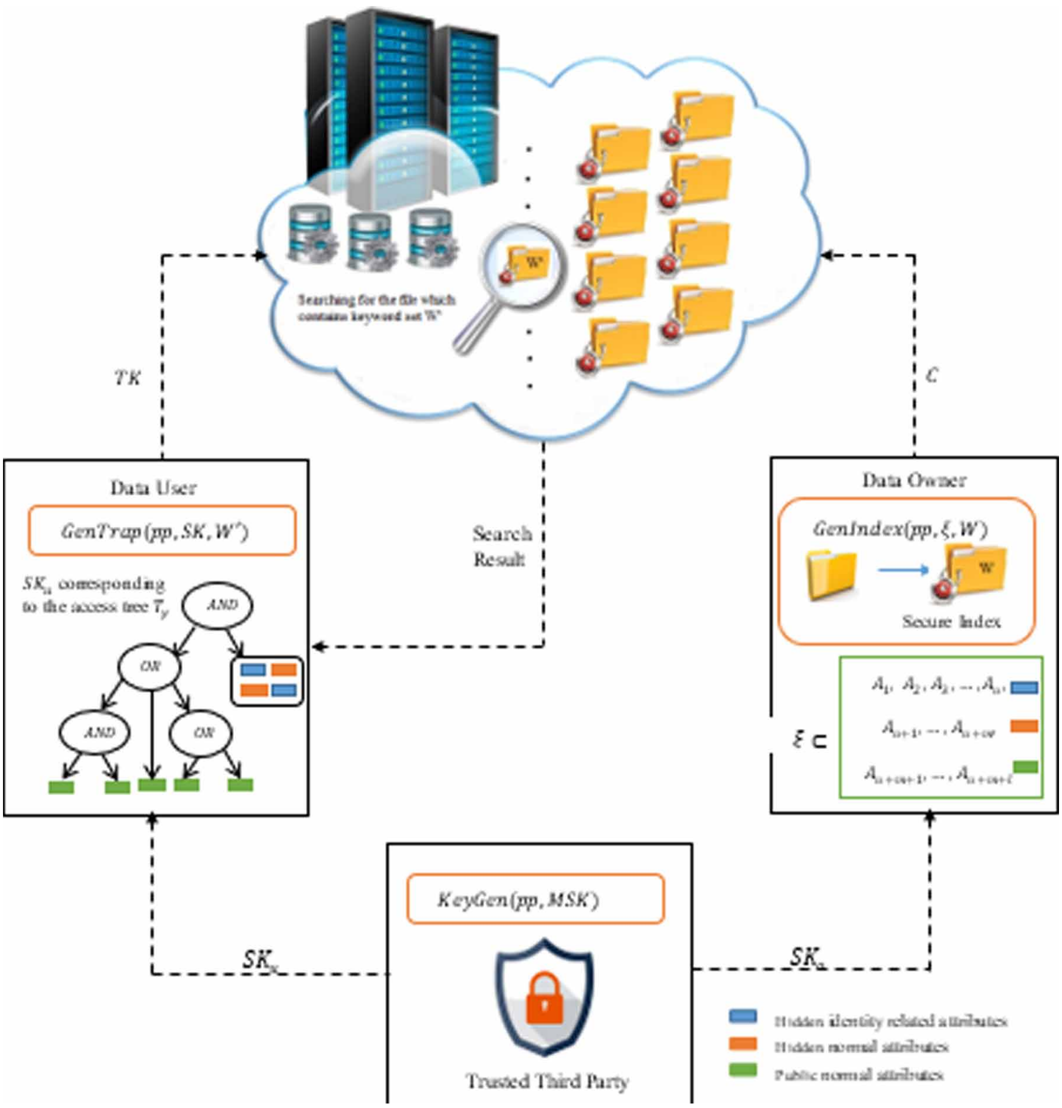
Security Model

The security of the proposed scheme is defined by the following two security games:

Game 1: This game defines the security against chosen keyword attack in selective security model which states that the adversary $\mathcal{A}$ cannot distinguish between encryption of two challenge keywords of its choice in the selective security model (Canetti et al. (2003)). It consists of the following steps:

1. Initialization: The adversary $\mathcal{A}$ commits the challenge attribute set ξ^*, a version number ver^* and a set of public normal attributes $\left\{S^{(\rho)}\right\}_{1 \le \rho \le ver^* - 1}$ on which she wish to be challenged upon.
2. Setup: The challenger $\mathcal{C}$ runs the Setup algorithm of the proposed scheme to generate the public parameters and master secret key and gives the public parameters to $\mathcal{A}$.
3. Phase 1: $\mathcal{A}$ can ask the secret key corresponding to any policy, γ, provided $\gamma\left(\xi^*\right) \neq 1$ and adversary can repeat this step polynomial number of times. By generating the secret key the challenger $\mathcal{C}$ can answer trapdoor query for any keyword, *w*, and maintains a list of keywords, L_k which is initially empty and adds *w* to L_k if $w \notin L_k$.

Figure 2. System model


Searching for the file which contains keyword set W'
TK
C
Data User
$GenTrap(pp, SK, W')$
SK_u corresponding to the access tree T_y
AND
OR
AND
OR
Search Result
Data Owner
$GenIndex(pp, \xi, W)$
W
Secure Index
$\xi \subset$
$A_1, A_2, A_3, \cdots, A_n$
$A_{n+1}, \cdots, A_{n+m}$
$A_{n+m+1}, \cdots, A_{n+m+l}$
$KeyGen(pp, MSK)$
Trusted Third Party
SK_u
SK_o
Hidden identity related attributes
Hidden normal attributes
Public normal attributes

4. Challenge Phase: $\mathcal{A}$ selects two challenge keywords w_0 and w_1 of equal length such that $w_0, w_1 \notin L_k$. The challenger flips a random coin b and encrypts w_b with parameters selected by $\mathcal{A}$ during initialization phase. The resulting challenge ciphertext is sent to $\mathcal{A}$.
5. Phase 2: $\mathcal{A}$ continues to query like in phase 1 with the additional restriction that $\mathcal{A}$ cannot ask trapdoor for w_0 and w_1.
6. Guess Phase: $\mathcal{A}$ outputs a guess bit b' of b and wins the game if $b' = b$.

Let $Adv_{\mathcal{A}}^{sCKA} = \left| Pr\left[b' = b\right] - \frac{1}{2} \right|$ be the advantage of adversary winning the above defined game and the proposed scheme is sCKA secure if the advantage of any adversary winning the sCKA security game is negligible.

Game 2: This game defines the indistinguishability of the tracing and normal ciphertext in the selective security model. If one can distinguish between these two ciphertexts then we can get information about which set of attributes are used for generating them. As we know the attribute set used for encryption consists of three disjunctive subsets of public normal (PN) attributes, hidden normal (HN) attributes and hidden identity related (HID) attributes. In normal operation HID attributes are set as don't care while during trace HID attributes are set to represent the identity of a suspicious user. This discloses an ongoing tracing activity. It consists of the following steps:

1. Initialization: The adversary $\mathcal{A}$ commits two challenge attribute set $\xi_0 = \xi_{PN} \cup \xi_{HN} \cup \xi_{HID}$ and $\xi_1 = \xi_{PN} \cup \xi_{HN} \cup \xi_{HID}^*$ that $\mathcal{A}$ wishes to be challenged upon where ξ_{HID} represents the identity of a suspicious user and ξ_{HID}^* represents the identity where each bit is set as don't care and submits them to the challenger $\mathcal{C}$.
2. Setup: The challenger $\mathcal{C}$ runs the Setup algorithm of the proposed scheme to generate the public parameters and gives them to $\mathcal{A}$.
3. Phase 1: $\mathcal{A}$ can asks the secret key corresponding to any access structure T_γ and $\mathcal{C}$ answers the query only if the following condition holds: $\left(\left(\xi_0 \vDash T_\gamma\right) \wedge \left(\xi_1 \vDash T_\gamma\right)\right)$ or $\left(\left(\xi_0 \nvDash T_\gamma\right) \wedge \left(\xi_1 \nvDash T_\gamma\right)\right)$.
4. Challenge Phase: $\mathcal{A}$ submits two challenge keywords w_0 and w_1 to $\mathcal{C}$. If $\mathcal{A}$ obtained the secret key such that $\left(\left(\xi_0 \vDash T_\gamma\right) \wedge \left(\xi_1 \vDash T_\gamma\right)\right)$ then w_0 and w_1 should be of equal length. The challenger flips a random coin *b* and encrypts w_b using attribute set ξ_b and gives the resulting challenge ciphertext to $\mathcal{A}$.
5. Phase 2: $\mathcal{A}$ continues to ask the secret key like in phase 1 but if $w_0 \neq w_1$, $\mathcal{A}$ cannot submit the secret query for the access structure T_γ for which $\left(\left(\xi_0 \vDash T_\gamma\right) \wedge \left(\xi_1 \vDash T_\gamma\right)\right)$.
6. Guess Phase: $\mathcal{A}$ outputs a guess bit b' of *b* and wins the game if $b' = b$.

Let $Adv_{\mathcal{A}}^{IND} = \left| Pr\left[b' = b\right] - \frac{1}{2} \right|$ be the advantage of adversary winning the above defined game and in the proposed scheme the tracing and the normal ciphertexts are indistinguishable if the advantage of an adversary winning the above defined security game is negligible.

PROPOSED SCHEME

This section discusses the approach used by authors for tracing the key abusers which is embedded in the scheme in the construction part given in the subsequent subsection. Finally, this section discusses the event when a user leaves the system and explains the necessary system update operations.

Basic Design

To achieve the security against key abusers we have used the approach proposed by Yu et al. (2009) where each user is assigned a unique identity, $ID = I_1 I_2 \cdots I_n$ which is a *n* - bit binary string and each bit corresponds to an attribute. If $I_j = 0$, then the corresponding attribute, A_j does not belong to the user having identity *ID*, otherwise $A_j \in ID$. So, to incorporate the tracing feature each user's access structure will have an extra *n* (at most) identity-related attributes in addition to the normal attributes corresponding to the access rights assigned to the user. When the data owner generates the ciphertext, these identity-related attributes are also used along with the normal attributes. There are different ways in which data owner can add the identity-related attributes: i) for tracing activity, data owner set these attributes to represent suspicious identity; ii) for normal operation, data owner sets these attributes as "don't care". During tracing, the data user is tricked to find some keyword which is included in the traceable ciphertext and he can generate a valid trapdoor and perform search if his identity is same as the one represented by suspicious identity. Now, the data owner can compare it with the identity used in traceable ciphertext and if a mismatch is found then data owner can find the original holder who has misused his secret key. For making the traceable ciphertext indistinguishable from the normal ciphertext, the identity-related attributes are hidden and also some of the normal attributes are hidden to avoid detection of ongoing tracing activity upon a failed search. Thus, the data user cannot find whether the failed search is the result of his access rights or the mismatched identity.

To enable the tracing of secret key abusers, the attributes are broadly divided into two types: i) Normal attributes: These attributes are used to assign the access rights to a user, ii) Hidden attributes: It contains some of the normal attributes which are fixed at the system setup and all the identity-related attributes. The normal attributes which are not hidden are called public normal attributes. The universe of attributes is represented as *Att* which consists of universe of public normal attributes, Att_{PN}, universe of hidden normal attributes, Att_{HN} and hidden identity-related attributes, Att_{HID} and $Att_{HN} \cup Att_{HID}$ represents the universe of hidden attributes, Att_H. Since, some attributes are hidden in the access structure of a user therefore our scheme supports partial anonymity. It is assumed that the attributes from Att_H are fixed at the system setup phase and are never updated.

Further, to handle the user revocation efficiently we have used the proxy re-encryption (PRE) (Yu et al., 2010) and lazy re-encryption techniques as used in (Yu et al., 2010; Sun et al., 2016) to delegate the computational intensive tasks to the cloud server. PRE scheme enables the semi-trusted proxy to convert a ciphertext from one form to another given the proxy re-encryption key i.e. using the proxy re-encryption key one can convert the ciphertext encrypted under public parameters, pp_a to another ciphertext encrypted under public parameters, pp_b and vice versa. Using lazy re-encryption technique one can aggregate several system operations together to save both computational effort and time. When a user is revoked, only the attributes from Att_{PN} needs to be updated because the hidden attributes are used only for the purpose of tracing and key management.

Detailed Construction

Before introducing the construction, Table 2 describes the various notations used in the proposed scheme as follows:

The details of the algorithms involved in the proposed scheme are given below:

1. $Setup\left(\lambda, Att, n\right)$: Using the attribute universe, *Att*, define the universe of hidden attributes, $Att_H = \left\{A_1, A_2, \cdots, A_n, A_{n+1}, \cdots, A_{n+m}\right\}$. Here, first *n* elements corresponds to hidden identity-related attributes, Att_{HID} and next *m* elements corresponds to hidden normal attributes, Att_{HN}. Att_{PN} represents the universe of public normal attributes and

Table 2. Description of the notations used in the proposed scheme

Notation	Description
λ	Security parameter
ID	Unique identity (n-bit binary string) assigned to a user, $ID = I_1, I_2, \cdots, I_n$
Att_{PN}, Att_{HN}, Att_{HID}	Universe of public normal attributes, hidden normal attributes and hidden identity related attributes
Att, $\mathcal{W}$	Attribute universe, $Att_{PN} \cup Att_{HN} \cup Att_{HID}$ and Keyword space, $\mathcal{W} = \{w_1, w_2, \cdots, w_{m'}\}$ where $w_i = \{0,1\}^*$
Z_p, *H*	Group of integers of prime order *p* and hash function, *H*, which maps any binary string to Z_p
G, G_T, *g*	*G*, G_T - Cyclic group of prime order *p* with generator *g*
MSK, *SK*, *PSK*	Master secret key of the trusted third party (TTP), secret key of the user and partial secret key sent to the cloud server respectively
ver	Version number of the system initialized to 1
pp	Public parameters
C_u	Ciphertext component corresponding to a user, *u*, sent to the cloud server
C, *TK*	Ciphertext and trapdoor for a set of keywords

$Att_{PN} = \{A_{n+m+1}, A_{n+m+2}, \cdots, A_{n+m+l}\} : |Att_{PN}| = l$, $|Att| = n + m + l$. For each attribute, $i \in Att_{PN}$, select random numbers $\{t_{i,k}\}_{1 \le k \le v_i} \in Z_p$ and compute $T_{i,k} = \{g^{t_{i,k}}\}_{1 \le k \le v_i}$ and for each hidden attribute, $j \in Att_H$, choose random numbers, $\{a_{j,k}, b_{j,k}\}_{1 \le k \le v_j} \in Z_p$ and random points, $\{B_{j,k}\}_{1 \le k \le v_j} \in G$ and compute $\left\{\left\{\left(B_{j,k}\right)^{a_{j,k}}, \left(B_{j,k}\right)^{b_{j,k}}\right\}_{1 \le k \le v_j}\right\}$. Finally, select a random number, $\alpha \in Z_p$ and compute $Y = e(g,g)^{\alpha}$. Select collision resistant one-way hash function, $H : \{0,1\}^* \to Z_p$. Output the public parameters,

$pp = \left(e, g, p, H, Y, \left\{\{T_{i,k}\}_{1 \le k \le v_i}\right\}_{i \in Att_{PN}}, \left\{\left\{\left(B_{j,k}\right)^{a_{j,k}}, \left(B_{j,k}\right)^{b_{j,k}}\right\}_{1 \le k \le v_j}\right\}_{j \in Att_H}\right)$ and master secret key, $MSK = \left(\alpha, \left\{\{t_{i,k}\}_{1 \le k \le v_i}\right\}_{i \in Att_{PN}}, \left\{\{a_{j,k}, b_{j,k}\}_{1 \le k \le v_j}\right\}_{j \in Att_H}\right)$. TTP publishes (ver, pp) and keep (ver, MSK), where *ver* is the version number and initially *ver* = 1.

2. New user Registration ($KeyGen(pp, MSK, \gamma)$): Whenever a new user wants to enroll, he/she sends a registration request to TTP. After receiving the registration request, TTP assigns the access privilege to that user as follows:

a. First of all, TTP selects a random number, $f_u \leftarrow Z_p$. and adds it to the MSK components and computes the corresponding public key component, $Y_u = Y^{f_u}$ for that user and publish it with other *pp*.
b. TTP constructs an access tree, T_γ corresponding to the access policy, γ assigned to a user. The root node of T_γ is an AND gate and all the hidden attributes assigned to a user appears at depth 1 of T_γ while the public normal attributes are present in the subtree, T_r rooted at depth 1. Let $Att_H^{'} \subseteq Att_H$ be the set of hidden attributes present in γ.
c. Now, TTP generates the secret key components for the subtree T_r (public normal attributes) as follows: [5]
   i. TTP defines a random polynomial, $q_i\left(x\right)$ for each node, *i* in T_r in top-down manner starting from the root node *r* of T_r.
   ii. For each node, *i*, set the degree, d_i of $q_i\left(x\right)$ to be one less than the threshold value, K_i of that node.
   iii. For root node *r*, randomly select a number, $v \in Z_p$ and set $q_r\left(0\right) = v$ and then randomly select $K_r - 1$ points to define q_r completely.
   iv. For each non-root node, $j \in T_r$ set $q_j\left(0\right) = q_{parent\left(j\right)}\left(index\left(j\right)\right)$ and other $K_j - 1$ points are chosen randomly.

The above process is repeated till the leaf nodes are reached. Let L_r denotes the set of leaf nodes of T_r where each leaf node is associated with a public normal attribute taken from Att_{PN}. Now, TTP outputs secret key component for each leaf node, $i \in L_r$ as: $D_i = g^{\frac{q_i(0)}{t_{i,k}}}$ where $t_{i,k}$ is the value taken by the attribute, A_i.

d. The secret key components corresponding to the hidden identity-related attributes are computed as follows:
   i. Let $ID = I_1, I_2, \cdots, I_n$ be the unique identity assigned to the user.
   ii. If $I_j = 1$, then for the corresponding $A_j \in Att_{HID}$ TTP selects random numbers $x_j, y_j \in Z_p$ and computes: $\tilde{D}_j = g^{x_j}\left(B_{j,k}\right)^{a_{j,k}\, b_{j,k}\, y_j}$, $\hat{D}_j = g^{a_{j,k}\, y_j}$, $\overset{\text{Ç}}{D}_j = g^{b_{j,k}\, y_j}$
e. In the similar manner, the secret key components corresponding to the hidden normal attributes are generated.
f. Now, TTP calculates $x = \sum_{j \in Att_H^{'}} x_j$ and gets another secret key component, $D_0 = g^{\alpha - v - x}$.

Finally, TTP outputs the user secret key, *SK* as follows:

$$SK = \left(ver, f_u, D_0, \left\{D_i\right\}_{\forall i \in L_r}, \left\{\tilde{D}_j, \hat{D}_j, \overset{\text{Ç}}{D}_j\right\}_{\forall j \in Att_H^{'}}\right)$$

Now, TTP sends the partial secret key, $PSK = \left(ver, \left\{D_i\right\}_{\forall i \in L_r}\right)$ to the cloud server along with the identity of the user. PSK contains the secret key components corresponding to the public normal attributes. It allows the cloud server to update these components during user revocation. Note that the cloud server cannot generate a valid trapdoor using PSK because of the undisclosed components corresponding to hidden attributes.

3. $GenIndex\left(pp, \xi \subseteq Att, W \in \mathcal{W}\right)$: Before outsourcing a data file to the cloud server, the data owner generates the encrypted index for this file using this algorithm. First of all data owner defines a set of attributes, $\xi \subseteq Att$ and the set of keywords, *W*, which will be used for searching this data file and then encrypt *W* with ξ to generate the index. The set of attributes, ξ is the combination of attributes taken from Att_{PN}, Att_{HN} and Att_{HID}. So, $\xi = \xi_{PN} \cup \xi_{HN} \cup \xi_{HID}$, where $\xi_{PN} \subseteq Att_{PN}$, $\xi_{HN} \subseteq Att_{HN}$, and $\xi_{HID} \subseteq Att_{HID}$. Let an identity corresponding to a ξ_{HID} be $I_1, I_2, \cdots, I_n$ where $I_j = 0 / 1 / *$, $\forall j : 1 \leq j \leq n$. Data owner selects a random number $s \leftarrow Z_p$ and compute $C_0 = g^s$, $C_1 = Y^s$. The remaining components are computed as follows:

   a. For the ciphertext components corresponding to ξ_{PN}, the data owner computes $C_{i,k} = \left\{ T_{i,k}^{\ s} \right\}_{i \in \xi_{PN}}$ and for some fixed attribute, $i' \in \xi_{PN}$ (position is publicly visible), set $C_{i',k} = T_{i',k}^{\ s \prod_{\forall w_j \in W} H\left(w_j\right)}$.

      i. For the ciphertext components corresponding to ξ_{HID}, the data owner selects random numbers $\left\{r_{j,k}\right\}_{1 \leq k \leq v_j} \leftarrow Z_p; 1 \leq j \leq n$ and computes $\left\{\hat{C}_{j,k}, \overset{\text{Ç}}{C}_{j,k}\right\}_{1 \leq j \leq n, 1 \leq k \leq v_j}$ as follows:

      ii. If $I_j = 0$, data owner sets $\left\{\hat{C}_{j,k}, \overset{\text{Ç}}{C}_{j,k}\right\}$ as random (mal-formed components) and if $I_i = 1$ then data owner sets $\left\{\hat{C}_{j,k}, \overset{\text{Ç}}{C}_{j,k}\right\} = \left\{\left(B_{j,k}^{a_{j,k}}\right)^{r_{j,k}}, \left(B_{j,k}^{b_{j,k}}\right)^{s - r_{j,k}}\right\}$ as well-formed components.

   b. If $I_j = *$, then data owner sets $\left\{\hat{C}_{j,k}, \overset{\text{Ç}}{C}_{j,k}\right\} = \left\{\left(B_{j,k}^{a_{j,k}}\right)^{r_{j,k}}, \left(B_{j,k}^{b_{j,k}}\right)^{s - r_{j,k}}\right\}$ as well-formed components.

   c. The ciphertext components corresponding to ξ_{HN} are computed in the same manner as stated in step b.

In addition to these components, data owner computes a ciphertext component, C_u for each newly registered user, *u*, with whom he wants to share his data by allowing him to search his encrypted data files as, $C_u = Y_u^{-s}$ and send it to the cloud server along with the identity of the user. The cloud server stores this information in a list, LU of legitimate users. Finally, output ciphertext,

$$C = \left(ver, C_0, C_1, \left\{C_{i,k}\right\}_{\forall i \in \xi_{PN}, 1 \leq k \leq v_i}, \left\{\hat{C}_{j,k}, \overset{\text{Ç}}{C}_{j,k}\right\}_{\forall j \in \xi_{HN}, 1 \leq k \leq v_j}\right)$$

4. $GenTrap\left(pp, SK, W'\right)$: When a legitimate user wants to find a document which contains a set of keywords, W', he generates a trapdoor by selecting a random number, $u \leftarrow Z_p$ and computing;

$Q_0 = D_0^u,\ Q_1 = u + f_u,\ \left\{ \tilde{Q}_j = \left(\tilde{D}_j\right)^u, \hat{Q}_j = \left(\hat{D}_j\right)^u, \overset{Ç}{Q}_j = \left(\overset{Ç}{D}_j\right)^u \right\}_{\forall j \in Att_H^{'}}$, for the same i' as in $GenIndex$, get $Q_{i'} = \left\{ D_{i'}^{\frac{u}{\prod_{\forall w_j \in W'} H\left(w_j\right)}} \right\}_{i \in L_r}$ and $Q_{i,k} = \left\{ \left(D_i\right)^u \right\}_{\forall i \in L_r \setminus i'}$.

Finally, output $TK = \left(ver, Q_0, Q_1, \left\{ \tilde{Q}_j, \hat{Q}_j, \overset{Ç}{Q}_j \right\}_{\forall j \in Att_H^{'}}, \left\{ Q_i \right\}_{\forall i \in L_r} \right)$.

5. $Search\left(pp, C_u, C, TK\right)$: Cloud server will execute this algorithm and output:
   a. The secret key components corresponding to the public normal attributes are combined with the corresponding ξ_{PN} components of the ciphertext as follows:
      i. For each attribute, $A_i \in \xi_{PN}$, let $C_i^{'} = C_{i,k}$ where value taken by A_i is $A_{i,k}$. For each leaf node, $x \in L_r$ (assuming $att\left(x\right)$ represents the attribute, A_i associated with node x) and for the ciphertext component corresponding to each $A_i \in \xi_{PN}$, compute:

$$F_x = \begin{cases} e\left(Q_i, C_i^{'}\right) = e\left(g, g\right)^{suq_x\left(0\right)}, & if\, att\left(x\right) \in \xi_{PN}\, and W = W' \\ \perp, & otherwise \end{cases}$$

For the attribute, $i' \in \xi_{PN}$, $e\left(Q_{i',k}, C_{i',k}\right)$ is also equal to $e\left(g, g\right)^{suq_i\left(0\right)}$.

      ii. Now, for each non-leaf node, y of T_r, recursively execute the following step in bottom-up manner: For each child x of y, construct a k_y sized set S_y (initially empty) which contains child of y such that $F_x \neq \perp$ and compute, F_y as follows:

$$F_y = \prod_{x \in S_y} F_x^{\Delta_{x,S_y}\left(0\right)} = \prod_{x \in S_y} e\left(g, g\right)^{suq_x\left(0\right)\Delta_{x,S_y}\left(0\right)}$$

By construction,

$$F_y = \prod_{x \in S_y} e\left(g, g\right)^{suq_{parent\left(x\right)}\left(index\left(x\right)\right)\Delta_{x,S_y}\left(0\right)}$$

By Lagrange Interpolation method,

$$F_y = e\left(g, g\right)^{suq_y\left(0\right)}$$

This recursion is repeated till the root node, r is reached:

$$F_r = e\left(g, g\right)^{suq_r\left(0\right)} = e\left(g, g\right)^{suv}$$

b. For each attribute, $A_j \in \xi_H$, let $\left[\hat{C}'_j, \overset{\text{ç}'}{C}_j\right] = \left[\hat{C}_{j,k}, \overset{\text{ç}}{C}_{j,k}\right]$ where value taken by A_j is $A_{j,k}$. The secret key components corresponding to the hidden attributes are combined with the corresponding ξ_H components of the ciphertext as follows:

$$F_H = \prod_{j \in Att_H} \frac{e\left(C_0, \tilde{Q}_j\right)}{e\left(\hat{C}'_j, \overset{\text{ç}}{Q}_j\right) e\left(\overset{\text{ç}'}{C}_j, \hat{Q}_j\right)} = e\left(g, g\right)^{sux}$$

The following equation holds if $\gamma\left(\xi\right) = 1$ and $W = \mathrm{W}'$:

$$C_1^{Q_1} \cdot C_u = e\left(C_0, Q_0\right) \cdot F_r \cdot F_H$$

6. Trace: For each id, choose a set of attributes, $\xi = \xi_{PN} \cup \xi_{HN} \cup \xi_{HID}$, such that ξ satisfies the access structure of id. Randomly choose a keyword, $w \in \mathcal{W}$ and get the index. Check if the suspected user is able to search *w*. If it does, stop and return id. Otherwise, continue.

Correctness

$$C_1^{Q_1} \cdot C_u = \left(Y^s\right)^{u+f_u} Y_u^{-s} = e\left(g, g\right)^{\alpha su}$$

$$e\left(C_0, Q_0\right) \cdot F_r \cdot F_H = e\left(g^s, g^{u\left(\alpha - v - x\right)}\right) \cdot e\left(g, g\right)^{suv} \cdot e\left(g, g\right)^{sux} = e\left(g, g\right)^{\alpha su}$$

User Revocation

When a user is revoked from the system then he/she should not be able to generate a valid trapdoor. For this purpose, TTP first determines the minimal set of attributes without which the revoked user's access structure will never be satisfied and he will never be able to generate a valid trapdoor. So, TTP should update these attributes by defining new *MSK* and *pp* corresponding to all the updated attributes and should update secret key for all the remaining users and further the index also needs to be re-encrypted with the modified *pp*. This whole process incurs a heavy computational burden on TTP and also need TTP to remain online. To avoid this overhead of heavy computations we have used proxy re-encryption and lazy re-encryption techniques where these computations are delegated to the cloud server. Whenever a user is revoked from the system the version number is incremented by 1. The process of user revocation is divided into the following two stages:

1. In the first stage, TTP determines the minimal set, $S \subseteq Att_{PN}$ of attributes that needs to be updated for the revoked user having identity, *id*. For each attribute, $i \in S$, TTP selects a new *MSK* component, t'_i, compute *pp* component, $T'_i = g^{t'_i}$ and generates the corresponding re-encryption key, $rk_i = \frac{t'_i}{t_i}$ and for the remaining attributes set $rk_i = 1$. TTP sends

$rk = id, S, ver, rk_i,\ T_i^{'}$ to the cloud server and go offline. On receiving this information, the cloud server removes the revoked users from *LU* and compares the *ver* in *rk* with the current *ver* of the system. If it is less than the current *ver* then the cloud server discards *rk*, otherwise, stores the re-encryption key.

2. The second stage of user revocation starts when the cloud server receives a trapdoor for a particular keyword. On receiving the trapdoor request cloud server first determines if the requesting user is a legitimate by checking his entry in *L*. If the user belongs to *L* list, the cloud server gets the $PSK = \left(ver, \left\{D_i\right\}_{\forall i \in L_r}\right)$ tuple stored at the cloud server at the time of key generation. If *ver* is already the latest version then the cloud server will not update the index or the secret key components else calls ReGenIndex and ReKeyGen algorithm, only when it receives a trapdoor for a particular keyword. Thus, reduces a lot of computational burden from the cloud server by aggregating multiple re-encryption keys. This technique of re-encryption is called lazy re-encryption where the index is re-encrypted once with aggregated re-encryption keys, $RK = \prod_{\rho=2}^{q} rk_i^{\rho} = \frac{t_i^{(q)}}{t_i}$ where q is the latest version. Now, compute $C_i^{(q)} = \left(C_i\right)^{RK}$ and $D_i^{(q)} = \left(D_i\right)^{\frac{1}{RK}}$. The cloud server replaces the old C_i components with the updated $C_i^{(q)}$ components and sends the updated secret key components, $D_i^{(q)}$ to the user and finally the cloud server can remove the id of the revoked user and the associated data from *LU*. On receiving the updated $D_i^{(q)}$ components from the cloud server, the user can verify whether the cloud server has correctly computed them by checking $e\left(T_i, D_i\right) = e\left(T_i^{(q)}, D_i^{(q)}\right)$.

ANALYSIS OF THE PROPOSED SCHEME

This section gives the detailed security analysis of the proposed scheme and also gives a comparative analysis of the proposed scheme with similar existing scheme in terms of storage and computational cost.

Security Analysis

(IND-sCKA) Indistinguishability of keywords against chosen keyword attack in selective security model. The following theorem guarantee the semantic security of keywords in selective security model:

Theorem 5.1: The proposed scheme is IND-sCKA secure under DBDH assumption if $\forall \mathcal{A}$ (probabilistic polynomial time adversary), the advantage, ε of winning the security game is negligible.

$$Adv_{\mathcal{A}}^{IND-sCKA} = Pr\left[b' = b\right] - \frac{1}{2} = \varepsilon$$

Proof: Let there exists an adversary $\mathcal{A}$ who can win the IND-sCKA game with advantage ε then there exists a challenger $\mathcal{C}$. who can break the DBDH assumption with advantage $\frac{\varepsilon}{2}$, given an instance $\left(g^{z_1}, g^{z_2}, g^{z_3}, Z\right)$ of DBDH assumption. $\mathcal{C}$ simulates the IND-sCKA game defined in security model as follows:

1. Initialization: $\mathcal{A}$ selects the challenge attribute set $\xi^* = \xi^*_{PN} \cup \xi^*_{HN} \cup \xi^*_{HID}$, a version number ver^* and a set of public normal attributes $\left\{S^{(\rho)}\right\}_{1 \le \rho \le ver^*-1}$ and gives them to the challenger $\mathcal{C}$.
2. Setup: $\mathcal{C}$ sets $Y = e\left(g^{z_1}, g^{z_2}\right) = e(g,g)^{z_1 z_2}$ and implicitly defines $\alpha = z_1 z_2$. Randomly choose $\theta \leftarrow Z_p$ and set $Y_u = Y^{\theta}$. Now, for each value of an attribute $A_i \in Att_{PN}$, choose a random number $r_{i,k} \leftarrow Z_p$ and if $A_i \in \xi^*_{PN}$, set $T_{i,k} = g^{r_{i,k}}$; otherwise, set $T_{i,k} = g^{z_2 r_{i,k}}$. For each value of an attribute $A_j \in Att_{HID}$, choose a random number $h_{j,k} \leftarrow Z_p$, and if $A_j \in \xi^*_{HID}$ set $B_{j,k} = g^{h_{j,k}}$ else set $B_{j,k} = g^{z_2 h_{j,k}}$. Now publish the *pp* by choosing $a_{j,k}, b_{j,k} \leftarrow Z_p$ for version 1. For each attribute set $\left\{S^{(\rho)}\right\}_{2 \le \rho \le ver^*-1}$, $\mathcal{C}$ generates the re-encryption key $rk^{(\rho)}$ and the corresponding *pp* for that version. For each attribute, $i \in S^{(\rho)}$, choose a random number $rk^{(\rho)}_{i,k} \leftarrow Z_p$ and set $T^{(\rho+1)}_{i,k} = \left(T^{(\rho)}_{i,k}\right)^{rk^{(\rho)}_{i,k}}$ and for the remaining attributes $i \notin S^{(\rho)}$, set $rk^{(\rho)}_{i,k} = 1$ and sends the $rk^{(\rho)}_{i,k}$ to $\mathcal{A}$. The remaining *pp* for version $\rho + 1$ are same as version ρ.
3. Phase 1: In this phase $\mathcal{A}$ can ask secret key for any access policy γ represented by T_γ, such that $\xi^* \not\models T_\gamma$. Depending upon the structure of T_γ, $\mathcal{C}$ defines the following two cases and answers the secret key queries accordingly:
   a. *Case 1-* $\xi^*_{PN} \not\models T_r$: Here, $\mathcal{C}$ generates the secret key components corresponding to the hidden attributes in the same manner as in the original scheme and to generate the secret key components corresponding to T_R, $\mathcal{C}$ uses the $PolySat\left(\mathcal{T}_x, \xi^*_{PN}, \lambda_x\right)$ and $PolyUnSat\left(\mathcal{T}_x, \xi^*_{PN}, g^{\lambda_x}\right)$ procedures defined in [Goyal] to assign a polynomial Q_x for every node in T_r. $\mathcal{C}$ calls $PolyUnSat\left(T_r, \xi^*_{PN}, g^{z_1 + r'}\right)$ by choosing a random number $r' \leftarrow Z_p$ and implicitly sets $q_r(0) = z_1 + r'$ and then recursively calls $PolySat\left(\mathcal{T}_x, \xi^*_{PN}, q_r\left(index(x)\right)\right)$ for each satisfied child, x of node r and $PolyUnSat\left(\mathcal{T}_x, \xi^*_{PN}, g^{q_r\left(index(x)\right)}\right)$ for each unsatisfied child, x of node r to get q_x. Now, $\mathcal{C}$ defines the final polynomial $Q_x(\cdot) = z_2 q_x(\cdot)$ for each node x of T_r and defines the secret key components corresponding to each leaf node of T_r as follows, where each leaf node x corresponds to an attribute A_i:

$$D_x = \begin{cases} g^{\frac{Q_x(0)}{t_{i,k}}} = g^{\frac{z_2 q_x(0)}{r_{i,k}}} = \left(g^{z_2}\right)^{\frac{q_x(0)}{r_{i,k}}}, & if\, att(x) \in \xi^*_{PN} \\ g^{\frac{Q_x(0)}{t_{i,k}}} = g^{\frac{z_2 q_x(0)}{z_2 r_{i,k}}} = g^{\frac{q_x(0)}{r_{i,k}}}, & Otherwise \end{cases}$$

Finally set $D_0 = g^{\alpha - v - x} = g^{z_1 z_2 - z_2\left(z_1 + r'\right) - x} = g^{-z_2 r' - x}$, where x is generated while computing secret key components corresponding to hidden attributes. Now, for $2 \le \rho \le ver^*$ if an attribute $A_i \notin S^{(\rho)}$, set $rk_{i,k}^{(\rho)} = 1$. Otherwise, for each $A_i \in S^{(\rho)}$, $\mathcal{C}$ computes $R_i^\rho = \prod_{\omega=2}^{\rho-1} rk_{i,k}^{\omega}$ and set $D_x = \left(D_x\right)^{\frac{1}{R_i^\rho}} = \left(g^{z_2}\right)^{\frac{q_x(0)}{r_{i,k}R_i^\rho}}$ if $att(x) \in \xi_{PN}^*$ else set $D_x = (g)^{\frac{q_x(0)}{r_{i,k}R_i^\rho}}$.

b. *Case 2-* $\xi_{PN}^* \vDash T_r$: Here, $\mathcal{C}$ generates the secret key components corresponding to the *PN* attributes in the same manner as in the original scheme. Since $\xi_{PN}^* \vDash T_r$, therefore the hidden attributes in T_γ does not match ξ_H^*. Let A_j be a hidden attribute which is not meant by ξ_{HID}^*. For the hidden attributes, randomly choose $x_i^{'}\Big|_{\forall i=1}^{m+n} \leftarrow Z_p$ and set $x_j = z_1 z_2 + x_j^{'}$ and $x_i = x_i^{'}$ for every $i \neq j$. Now, set $x = \sum_{i=1}^{m+n} x_i = z_1 z_2 + \sum_{i=1}^{m+n} x_i^{'}$ and compute $\tilde{D}_j = g^{x_j}\left(B_{j,k}\right)^{a_{j,k}b_{j,k}y_j} = g^{x_j^{'}}\left(g^{z_2 h_{j,k}}\right)^{a_{j,k}b_{j,k}y_j^{'}}$, where $y_j^{'}$ is chosen by $\mathcal{C}$ and $y_j = \frac{-z_1}{a_{j,k}b_{j,k}h_{j,k}} + y_j^{'}$. Now $\left\{\hat{D}_j, \overset{Ç}{D}_j\right\}$ are computed in the same manner as $\tilde{D}_j$ and for $i \neq j$, $\left\{\tilde{D}_i, \hat{D}_i, \overset{Ç}{D}_i\right\}$ are computed like in the original scheme. Finally compute the component, $D_0 = g^{\alpha - v - x} = g^{z_1 z_2 - v - z_1 z_2 - \sum_{i=1}^{m+n} x_i^{'}} = g^{-v-\sum_{i=1}^{m+n} x_i^{'}}$, where *v* is generated while computing secret key components corresponding to T_r.

Finally, $\mathcal{C}$ sends the secret key *SK* to $\mathcal{A}$ and using the secret key credentials, $\mathcal{C}$ can also answer the trapdoor query for any keyword, *w*. In addition, $\mathcal{C}$ maintains a list, L_t of the keywords and the corresponding trapdoor value for which adversary had asked the trapdoor. If the queried keyword $w \notin L_t$, then $\mathcal{C}$ answers the query by looking into L_t. If $w \notin L_t$, then $\mathcal{C}$ generates the credentials to answer the query and adds *w* and the corresponding trapdoor value to L_t.

4. Challenge Phase: $\mathcal{A}$ submits two keywords w_0, w_1 of equal length such that $\left\{w_0, w_1\right\} \notin L_t$ and send them to $\mathcal{C}$. Now, $\mathcal{C}$ flips a fair coin, *b* and encrypts the corresponding keyword w_b to generate the challenge ciphertext as follows: First, $\mathcal{C}$ implicitly sets $s = z_3$ and for ver^* computes $C_0 = g^{z_3}$, $C_1 = Z$, $C_{i,k} = \left(g^{z_3}\right)^{r_{i,k}R_i^{\left(ver^*\right)}}$, $C_{i',k} = \left(g^{z_3}\right)^{H\left(w_b\right)r_{i',k}R_{i'}^{\left(ver^*\right)}}$, $C_u = Z^{-\theta}$. The remaining components $\left\{\hat{C}_{j,k}, \overset{Ç}{C}_{j,k}\right\}$ can also be generated correctly because $B_{j,k}$ does not contain the unknown z_2 if $A_{j,k} \in \xi_H^*$ (the kth occurrence of attribute *j* belongs to ξ_H^*), otherwise $\left\{\hat{C}_{j,k}, \overset{Ç}{C}_{j,k}\right\}$ are randomly chosen. Note that if $Z = e\left(g, g\right)^{z_1 z_2 z_3}$ then the challenge ciphertext is a valid encryption of keyword w_b, otherwise it represents some random value.
5. Phase 2: $\mathcal{A}$ continues to query like in phase 1 with the restriction that $\mathcal{A}$ cannot query the trapdoor for w_b if $\gamma\left(\xi^*\right) = 1$.
6. Guess Phase: $\mathcal{A}$ submits her guess b' of *b*. If $Z = e\left(g, g\right)^{z}$ where *z* is a random number from Z_p then $\mathcal{A}$ gets no information about *b* but a random guess. So $Pr\left[b = b'\right] = \frac{1}{2} = Pr\left[b \neq b'\right]$.

So, the challenger correctly guesses z to be some random when $b \neq b'$ with probability $\frac{1}{2}$. If $Z = e\left(g, g\right)^{z_1 z_2 z_3}$, then ciphertext represents a valid encryption of w_b and $Pr\left[b = b'\right] = \frac{1}{2} + \varepsilon$. So the challenger correctly guesses z to be $z_1 z_2 z_3$ when $b = b'$ with probability $\frac{1}{2} + \varepsilon$. Therefore, the probability of correctly guessing z given the DBDH challenge instance $\left(g^{z_1}, g^{z_2}, g^{z_3}, Z\right)$ is $\frac{1}{2}\left(\frac{1}{2} + \frac{1}{2} + \varepsilon\right) = \frac{1}{2} + \frac{\varepsilon}{2}$ and the advantage with which the challenger can solve the DBDH problem is $\frac{1}{2} + \frac{\varepsilon}{2} - \frac{1}{2} = \frac{\varepsilon}{2}$, if $\mathcal{A}$ wins the sCKA game with advantage ε. However, the DBDH problem is a known hard problem having negligible advantage so ε is also negligible and this proves that the proposed scheme is secure against sCKA.

Theorem 5.2: In the proposed scheme it is hard to distinguish the tracing ciphertext and the normal ciphertext under Decision Linear assumption.

Proof: Let there exists an adversary $\mathcal{A}$ who can win the Indistinguishability game (*IND - G*) with advantage ε then we can build a simulator $\mathcal{B}$ that can solve the Decision Linear problem with advantage $\frac{\varepsilon}{2}$, given an instance $\left(g, g^{z_1}, g^{z_2}, Z, g^{z_1 z_4}, g^{z_3 + z_4}\right)$ of decision linear assumption.

Before the beginning of IND-game, $\mathcal{A}$ commits two attribute sets $\xi_0 = \xi_{PN} \cup \xi_{HN} \cup \xi_{HID}$ and $\xi_1 = \xi_{PN} \cup \xi_{HN} \cup \xi_{HID}^*$, where in ξ_{HID} contains the attributes corresponding to identity of the suspicious user and ξ_{HID}^* represents the identity where each bit is don't care. The indistinguishability of tracing and normal ciphertext uses a sequence of games where the original game takes the identity of suspicious user in ξ_0. The *IND* – G_1 is same as the original game except that in ξ_0, ξ_{HID} represents the identity $I_1 * * * \cdots *$, where only the first bit is kept same as in the original game and remaining *n* - 1 bits are set as don't care. In *IND* – G_2 the first two bits are kept same and the remaining *n* - 2 bits are set as don't care and so on. Thus, the *IND* – G_n represents the original game. Now, to prove the indistinguishability of tracing and normal ciphertext, it is sufficient to prove the *IND* – G_1 and *IND* – G_{1+1} are indistinguishable. In the sequence of games from *IND* – G_1 to *IND* – G_n every time by replacing the original bits from the upper side with the don't care bits, we can embed the decision linear challenge in the ciphertext in such a way that if *IND* – G_1 and *IND* – G_{1+1} are distinguishable then it leads to the distinguishability of the decision linear challenge.

$\mathcal{B}$ simulates the IND-Game defined in security model as follows:

1. Initialization: In this phase, $\mathcal{A}$ selects two challenge attribute set ξ_0 and ξ_1, where the difference lies only in the set of hidden attributes related to identity, ξ_0 contains ξ_{HID} attributes related to the identity of suspected user while ξ_1 contains don't care values for each attribute in ξ_{HID} and gives them to the simulator $\mathcal{B}$. Now, $\mathcal{B}$ flips a fair coin *b* and
2. Setup: In this phase, $\mathcal{B}$ sets the *pp* based on the decision linear challenge. $\mathcal{B}$ sets $Y = e\left(g, g\right)^{\alpha}$ where α is known to $\mathcal{B}$. Randomly choose $\theta \leftarrow Z_p$ and set $Y_u = Y^{\theta}$. For the normal attributes, parameters are set in the similar manner as in proof of theorem 1. For each value of an attribute $A_j \in Att_{HID}$, choose a random number $h_{j,k} \leftarrow Z_p$, and if $A_j \in \xi_{HID} \cap \xi_{HID}^*$ set $B_{j,k} = g^{h_{j,k}}$ else

set $B_{j,k} = g^{z_2 h_{j,k}}$. Finally, $\mathcal{B}$ publishes the pp by choosing the random numbers $\left\{a_{j,k}, b_{j,k}\right\}_{1 \le k \le v_j} \leftarrow Z_p$ but for a_{j_l,k_l}, b_{j_l,k_l} set $a_{j_l,k_l} = z_1$ and $b_{j_l,k_l} = z_2$ and compute $B_{j_l,k_l}^{a_{j_l,k_l}} = \left(g^{z_1}\right)^{h_{j_l,k_l}}$ and $B_{j_l,k_l}^{b_{j_l,k_l}} = \left(g^{z_2}\right)^{h_{j_l,k_l}}$ without knowing z_1 and z_2.

3. Phase 1: In this phase, $\mathcal{A}$ can ask the secret key for any access structure T such that $\left(\xi_0 \vDash T \wedge \xi_1 \vDash T\right)$ or $\left(\xi_0 \nvDash T \wedge \xi_1 \nvDash T\right)$. The secret key components corresponding to subtree T_R and the hidden normal attributes in T are generated in the same manner as the original scheme. If there exists a HID attribute $A_{j_l,k_l} \in Att_{HID}$ such that $A_{j_l,k_l} \notin T$ then it is easy to find the corresponding secret key component, otherwise $\mathcal{B}$ needs to compute the corresponding secret key components as follows where $a_{j_l,k_l} = z_1$ and $b_{j_l,k_l} = z_2$:

$$\tilde{D}_{j_l} = g^{x_{j_l}} \left(B_{j_l,k_l}\right)^{a_{j_l,k_l} b_{j_l,k_l} y_{j_l}} = g^{x_{j_l}} \left(g^{h_{j_l,k_l}}\right)^{z_1 z_2 y_{j_l}} = g^{x'_{j_l}}$$

Where x_{j_l} is randomly chosen such that $x_{j_l} = x'_{j_l} - h_{j_l,k_l} z_1 z_2 y_{j_l}$, and x'_{j_l} is a random number chosen by $\mathcal{B}$. Similarly, $\mathcal{B}$ can compute the remaining secret key components $\left[\hat{D}_{j_l}, \overset{\text{ç}}{D}_{j_l}\right]$. To find the secret key components corresponding to HID attributes in T for the given challenge attribute set, ξ_b, we consider only the case where $\left(\xi_0 \nvDash T \wedge \xi_1 \nvDash T\right)$ because as per the definition in the security model if $\left(\xi_0 \vDash T \wedge \xi_1 \vDash T\right)$ then the challenge keywords will be equal and $\mathcal{B}$ simply terminates. Since, $\left(\xi_0 \nvDash T \wedge \xi_1 \nvDash T\right)$ hence there exists some attribute $A_m \notin \xi_b$, and in this case $\mathcal{B}$ computes the secret key components by selecting a random number $x'_m \leftarrow Z_p$ and set $x_m = x'_m + h_{j_l,k_l} z_1 z_2 y_{j_l}$ and compute:

$$\tilde{D}_m = g^{x_m} \left(B_{m.k}\right)^{a_{m,k} b_{m,k} y_m} = g^{x'_m + h_{j_l,k_l} z_1 z_2 y_{j_l}} \left(g^{z_2 h_{m,k}}\right)^{a_{m,k} b_{m,k} y_m} = g^{x'_m} \left(g^{z_2 h_{m,k}}\right)^{a_{m,k} b_{m,k} y'_m}$$

Here y_m is chosen at random such that $y_m = y'_m - \dfrac{z_1 h_{j_l,k_l} y_{j_l}}{a_{m,k} b_{m,k} h_{m,k}}$ where $y'_m \leftarrow Z_p$ is chosen randomly by $\mathcal{B}$.

Similarly, $\mathcal{B}$ can compute $\left[\hat{D}_m, \overset{\text{ç}}{D}_m\right]$. Finally, for $j \ne j_l, m$, $\mathcal{B}$ computes

$$x = \sum_{j=1}^{n} x_j = x_{j_l} + x_m + \sum_{j \ne j_l, m}^{n} x_j = x'_{j_l} - h_{j_l,k_l} z_1 z_2 y_{j_l} + x'_m + h_{j_l,k_l} z_1 z_2 y_{j_l} + \sum_{j \ne j_l, m}^{n} x_j = x'_{j_l} + x'_m + \sum_{j \ne j_l, m}^{n} x_j.$$

Now, $\mathcal{B}$ can compute $D_0 = g^{\alpha - v - x}$ where v is calculated while computing secret key components for T_R like in the original scheme and a is already known to $\mathcal{B}$.

4. Challenge Phase: In this phase, $\mathcal{A}$ submits two challenge keywords, $\left\{w_0, w_1\right\}$ of equal length and if $\mathcal{A}$ had asked for the secret key corresponding to T in phase 1 such that $\left(\xi_0 \vDash T \wedge \xi_1 \vDash T\right)$

then w_0 should be equal to w_1. Now, $\mathcal{B}$ tosses a fair coin *b* and encrypts w_b using the attribute set ξ_b by setting $C_0 = g^{z_3+z_4}$, $C_1 = e\left(g, g^{z_3+z_4}\right)^{\alpha}$ which implies $s = z_3 + z_4$. $\mathcal{B}$ generates the ciphertext components corresponding to ξ_{PN} and ξ_{HN} as in the original game and the ciphertext components corresponding to HID attributes in ξ_b are generated in the same manner as $IND - G_l$ with the exception that the components $\left\{\hat{C}_{j_l,k_l}, \overset{\text{Ç}}{C}_{j_l,k_l}\right\}$ are computed as follows: $\hat{C}_{j_l,k_l} = \left(B_{j_l,k_l}^{a_{j_l,k_l}}\right)^{r_{j_l,k_l}} = \left(g^{z_1 z_4}\right)^{h_{j_l,k_l}}$ and $\overset{\text{Ç}}{C}_{j_l,k_l} = \left(B_{j_l,k_l}^{b_{j_l,k_l}}\right)^{s-r_{j_l,k_l}} = Z^{h_{j_l,k_l}}$ without knowing $z_1 z_4$ and $z_2 z_3$. If $Z = g^{z_2 z_3}$ then the components are well-formed and $\mathcal{A}$ is in the game *IND* – G_l, otherwise $\mathcal{A}$ is in the game *IND* – G_{l+1}.

5. Phase 2: $\mathcal{A}$ can continue secret key queries like in phase 1.
6. Guess Phase: Here, $\mathcal{A}$ outputs b' as its guess for bit *b* and gives it to $\mathcal{B}$ and the difference of probability that $\mathcal{A}$ guesses *b* correctly in *IND* – G_l and in *IND* – G_{l+1} is negligible because if $Z = g^{z_2 z_3}$ then $\mathcal{A}$ is in the game *IND* – G_l and if *Z* is random then $\mathcal{A}$ is in the game *IND* – G_{l+1}. This implies distinction of *IND* – G_l and *IND* – G_{l+1} leads to the distinction of decision linear challenge which is a known hard problem.

Storage and Computational Cost Analysis

This section compares the storage and computational cost of the proposed scheme with the existing key policy attribute-based keyword search schemes.

As shown in the Table 3, in the proposed scheme the size of the secret key, the encrypted index and the trapdoor varies proportionally with the number of attributes like in the similar existing schemes except in the scheme proposed by Mamta and Gupta (2019a) where the target was to make them independent of the number of attributes. Therefore, with respect to performance nothing remarkable has been achieved but the authors have managed to incorporate phenomenal features like the proposed scheme can trace the secret key abusers, thus ensures accountability and also it can handle the event of user revocation efficiently by delegating the computational intensive tasks to the cloud server, thus reduces computational burden over the data owner and the trusted third party.

Performance Analysis

To evaluate the performance, the authors have implemented the proposed scheme in JAVA using Netbeans-8.1 IDE and java pairing based cryptography library (JPBC) (Caro & Iovino, 2011) on a 64-bit windows-10 system with Intel core i3 processor 2.00 GHz and 4 GB RAM. In JPBC to instantiate Bilinear map we have used Type A pairing constructed on elliptic curve, $y^2 = x^3 + x$ over a field F_q, where $q \equiv 3 mod 4$ is some prime. In this pairing both G_1 and G_2 are the group of points from $E\left(F_q\right)$ and hence it is called symmetric pairing. The size of the base field is set to be 512-bit which offers a security equivalent to 1024-bit DLOG (Caro & Iovino, 2011) and the order, *p* of source group *G* and target group G_T is set to be 160-bit. To demonstrate the performance, the authors have varied the number of attributes in the attribute universe, the access policy and in the set ξ from 10 to 50 with a step length of 10 and in each step the experiment has been executed 10 times to find the average time taken by each algorithm which is listed below in Table 4.

From Table 4, it is observed that the running time of all the algorithms varies linearly with the number of attributes as all of the algorithms contain components proportional to the number of attributes. For the better demonstration of the experimental results, the authors have plotted the average execution time taken by each algorithm against the number of attributes in Figure 3.

Table 3. Comparison of storage and computational cost with similar existing scheme

Scheme	Algorithm	Storage Cost	Computational Cost
Zheng et al. (2014)	KeyGen	$\left(2N\right)\left\|G\right\|$	$3NE+NH$
	GenIndex	$\left(\xi+3\right)\left\|G\right\|$	$\left(\xi+4\right)E+\xi H$
	GenTrap	$\left(2N+2\right)\left\|G\right\|$	$\left(2N+2\right)E$
	Search		$\left(2\xi+2\right)P+\xi E_T$
Li et al. (2017)	KeyGen	$\left(2N+4\right)\left\|G\right\|+1\left\|Z_p\right\|$	$\left(2N+6\right)E$
	GenIndex	$K\left\|G_T\right\|+\left(\xi+2\right)\left\|G\right\|$	$\left(\xi+2\right)E+2KE_T+KH$
	GenTrap	$\left(2N+2\right)\left\|G\right\|$	$2E+1H$
	Search		$\left(2\xi+2\right)P+\xi E_T+1H$
Ameri et al. (2018)	KeyGen	$\left(2N\right)\left\|G\right\|$	$\left(2N+1\right)E+NH$
	GenIndex	$\left(\xi+3\right)\left\|G\right\|+1Z_p$	$\left(\xi+4\right)E+\left(\xi+2\right)H$
	GenTrap	$\left(2N+t+1\right)\left\|G\right\|$	$\left(2N+t+1\right)E+\left(t+1\right)H$
	Search		$\left(2\xi+2\right)P+\xi E_T+tE$
Mamta & B. B. Gupta (2019a)	KeyGen	$1\left\|Z_p\right\|+3\left\|G\right\|$	$3E+1H$
	GenIndex	$\left(\xi+3+2\left\|U\right\|\right)\left\|G\right\|$	$\left(\xi+6\right)E+\left(K+1\right)H$
	GenTrap	$6\left\|G\right\|$	$7E+1P+1H$
	Search		$\left(\xi+2\right)E+6P$
Proposed Scheme	KeyGen	$\left(mN'+3mN''+1\right)\left\|G\right\|+1Z_p$	$\left(mN'+4mN''+1\right)E+1E_T$
	GenIndex	$\left(m\xi'+2m\xi''+1\right)\left\|G\right\|+1G_T$	$\left(m\xi'+2m\xi''+1\right)E+1E_T+KH$
	GenTrap	$\left(mN'+3mN''+1\right)\left\|G\right\|+1Z_p$	$\left(mN'+3mN''+1\right)E+KH$
	Search		$\left(m\xi'+3m\xi''+1\right)P+\xi' E_T+1E$

Notations used in Table 3:

G - Source group; G_T - Target group; *K* - # of keywords; *H* - Hash operation; Z_p - Group of integers of prime order, *p P* - Pairing operation in Bilinear Maps; *E* - Exponent operation in Source group E_T - Exponent operation in Target group; *N* - # of attributes associated with access policy ξ — # of attributes associated with users; ξ'— # of public normal attributes associated with users

ξ''— # of hidden attributes associated with users; N'— # of public normal attributes associated with the access policy; N''— # of hidden attributes associated with the access policy; *m* - # of possible values an attribute can take

Table 4. Average execution time (second) of the proposed scheme (# of attributes in attribute universe, access policy and the set ξ are kept same and m is set to 1)

# of attributes	10	20	30	40	50
KeyGen	0.812493	1.672628	2.449628	3.181871	3.967871
GenIndex	1.0353108	1.336525	1.745529	2.051441	2.324317
GenTrap	0.597493	0.917628	1.20123	1.586872	1.803499
Search	0.4749731	0.68453	0.957417	1.175047	1.312767

Figure 3. Average execution time of the proposed scheme

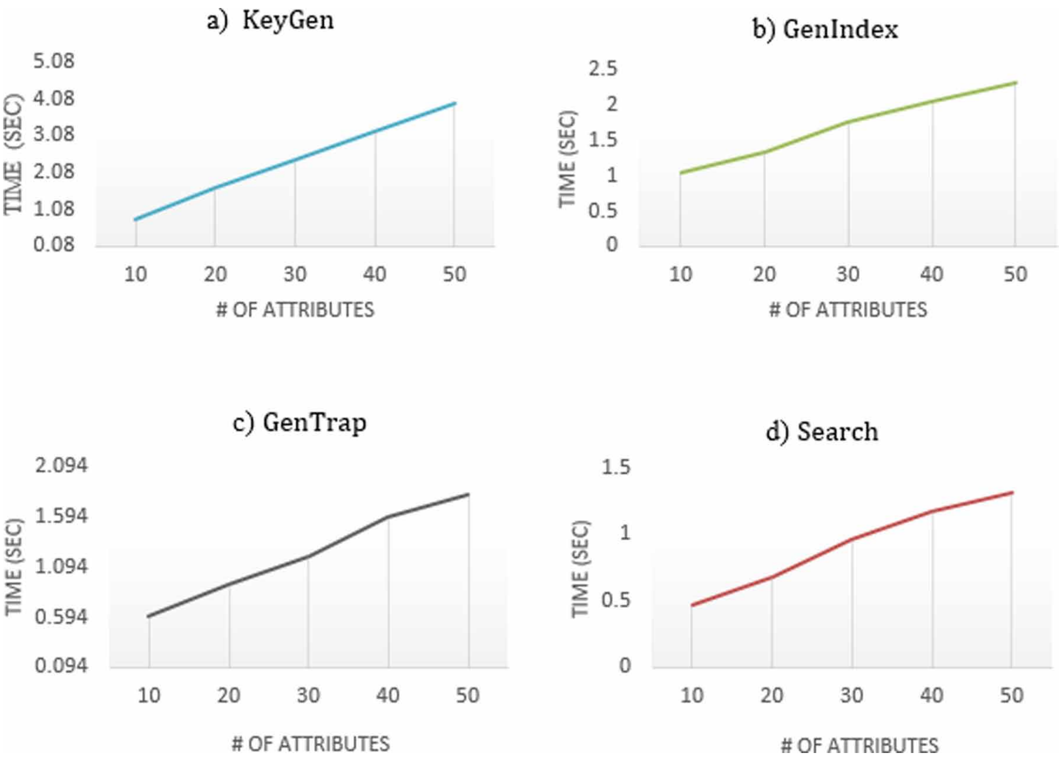


Figure 3-a) shows the average execution time of KeyGen algorithm and it can be observed that the average time varies linearly with the number of attributes. The reason for this linear relation are the components, $\{D_i\}_{\forall i \in L_r}$ and $\{\tilde{D}_j, \widehat{D}_j, \breve{D}_j\}_{\forall j \in Att'_H}$ which depends upon the number of attributes (public normal and hidden attributes respectively). Figure 3-b) shows that the average execution time of GenIndex algorithm also increases with the increase in number of attributes because of the components, $\{C_{i,k}\}_{\forall i \in \xi_{PN}, 1 \le k \le v_i}$ and $\left\{\hat{C}_{j,k}, \overset{\c{}}{C}_{j,k}\right\}_{\forall j \in \xi_{HN}, 1 \le k \le v_j}$ which varies with the number of attributes. Also, the average execution time shown in Figure 3-c) and 3-d) is directly proportional to the number of attributes. The trapdoor is generated using secret key which varies with the number of attributes and thus GenTrap algorithm time will also vary in the same way. In the Search algorithm, the number of pairing operations and the exponentiation operations in the target group increases with the increase in the number of attributes which causes the linear graph shown in Figure 3-d). The asymptotic complexities shown in Table 3 exactly matches the simulation results shown in Table 4 and Figure 3, thus ensures the proposed scheme is correct.

CONCLUSION

In this paper, the authors have developed a secure fine-grained multi-keyword scheme using key policy design framework. The proposed efficiently handles user revocation using proxy and lazy re-encryption techniques. Since in the proposed scheme the access right is associated with the secret key of the user so any user can misuse his/her access privilege by giving his secret key to other users. In order to keep a check on such users the authors have added traceability feature where such key abusers can be traced. Finally, the authors have proved that the proposed scheme is secure against selective chosen keyword attack under Decisional Bilinear Diffie-Hellman assumption and also proved that the ciphertext generated during normal operation and during trace activity are completely indistinguishable under Decision Linear assumption in the selective security model. In future, the aim is to reduce the computational cost and make it independent of the number of attributes involved.

ACKNOWLEDGMENT

This publication is an outcome of the R&D work undertaken under the project Visvesvaraya Ph.D. Scheme of Ministry of Electronics & Information Technology, Government of India and being implemented by Digital India Corporation.

REFERENCES

Ameri, M. H., Delavar, M., Mohajeri, J., & Salmasizadeh, M. (2018). *A key-policy attribute-based temporary keyword search scheme for secure cloud storage*. IEEE Transactions on Cloud Computing. doi:10.1109/TCC.2018.2825983

Blaze, M., Bleumer, G., & Strauss, M. (1998, May). Divertible protocols and atomic proxy cryptography. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques* (pp. 127-144). Springer.

Canetti, R., Halevi, S., & Katz, J. (2003, May). A forward-secure public-key encryption scheme. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques* (pp. 255-271). Springer.

Chaudhari, P., & Das, M. L. (2019). *Privacy Preserving Searchable Encryption with Fine-grained Access Control*. IEEE Transactions on Cloud Computing. doi:10.1109/TCC.2019.2892116

Chen, Z., Zhang, F., Zhang, P., Liu, J. K., Huang, J., Zhao, H., & Shen, J. (2018). Verifiable keyword search for secure big data-based mobile healthcare networks with fine-grained authorization control. *Future Generation Computer Systems*, *87*, 712–724. doi:10.1016/j.future.2017.10.022

Cui, J., Zhou, H., Xu, Y., & Zhong, H. (2019). OOABKS: Online/Offline Attribute-based Encryption for Keyword Search in Mobile Cloud. *Information Sciences*, *489*, 63–77. doi:10.1016/j.ins.2019.03.043

Cui, J., Zhou, H., Zhong, H., & Xu, Y. (2018). AKSER: Attribute-based keyword search with efficient revocation in cloud computing. *Information Sciences*, *423*, 343–352. doi:10.1016/j.ins.2017.09.029

De Caro, A., & Iovino, V. (2011, June). jPBC: Java pairing based cryptography. In Proceedings of the 2011 IEEE symposium on computers and communications (ISCC) (pp. 850-855). IEEE.

Goyal, V., Pandey, O., Sahai, A., & Waters, B. (2006, October). Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security* (pp. 89-98). ACM. doi:10.1145/1180405.1180418

Gupta, B., Agrawal, D. P., & Yamaguchi, S. (Eds.). (2016). Handbook of research on modern cryptographic solutions for computer and cyber security. Hershey, PA; IGI global. doi:10.4018/978-1-5225-0105-3

Gupta, B. B. (Ed.). (2018). *Computer and cyber security: principles, algorithm, applications, and perspectives*. CRC Press.

Gupta, B. B., Gupta, S., & Chaudhary, P. (2017). Enhancing the browser-side context-aware sanitization of suspicious HTML5 code for halting the DOM-based XSS vulnerabilities in cloud. *International Journal of Cloud Applications and Computing*, *7*(1), 1–31. doi:10.4018/IJCAC.2017010101

Hu, B., Liu, Q., Liu, X., Peng, T., Wang, G., & Wu, J. (2017, May). DABKS: Dynamic attribute-based keyword search in cloud computing. In *Proceedings of the 2017 IEEE International Conference on Communications (ICC)* (pp. 1-6). IEEE. doi:10.1109/ICC.2017.7997108

Li, J., Lin, X., Zhang, Y., & Han, J. (2016). KSF-OABE: Outsourced attribute-based encryption with keyword search function for cloud storage. *IEEE Transactions on Services Computing*, *10*(5), 715–725. doi:10.1109/TSC.2016.2542813

Mamta, & Gupta, B. B. (2019a). An efficient KP design framework of attribute-based searchable encryption for user level revocation in cloud. *Concurrency and Computation: Practice and Experience*, e5291.

Mamta, & Gupta, B. B. (2019b). Dynamic Policy Attribute Based Encryption and its Application in Generic Construction of Multi-Keyword Search. *International Journal of E-Services and Mobile Applications*, *11*(4).

Qiu, S., Liu, J., Shi, Y., & Zhang, R. (2017). Hidden policy ciphertext-policy attribute-based encryption with keyword search against keyword guessing attack. *Science China. Information Sciences*, *60*(5), 052105. doi:10.1007/s11432-015-5449-9

San Nicolas-Rocca, T., & Olfman, L. (2013). End user security training for identification and access management. *Journal of Organizational and End User Computing*, *25*(4), 75–103. doi:10.4018/joeuc.2013100104

Subramaniyaswamy, V., Logesh, R., Abejith, M., Umasankar, S., & Umamakeswari, A. (2017). Sentiment analysis of tweets for estimating criticality and security of events. *Journal of Organizational and End User Computing*, *29*(4), 51–71. doi:10.4018/JOEUC.2017100103

Sun, W., Yu, S., Lou, W., Hou, Y. T., & Li, H. (2014). Protecting your right: Verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud. *IEEE Transactions on Parallel and Distributed Systems*, *27*(4), 1187–1198. doi:10.1109/TPDS.2014.2355202

Wang, H., Dong, X., & Cao, Z. (2017). Multi-value-independent ciphertext-policy attribute based encryption with fast keyword search. *IEEE Transactions on Services Computing*, 1. doi:10.1109/TSC.2017.2753231

Yin, H., Zhang, J., Xiong, Y., Ou, L., Li, F., Liao, S., & Li, K. (2019). CP-ABSE: A Ciphertext-Policy Attribute-Based Searchable Encryption Scheme. *IEEE Access*, *7*, 5682–5694. doi:10.1109/ACCESS.2018.2889754

Yin, H., Zhang, J., Xiong, Y., Ou, L., Li, F., Liao, S., & Li, K. (2019). CP-ABSE: A Ciphertext-Policy Attribute-Based Searchable Encryption Scheme. *IEEE Access*, *7*, 5682–5694. doi:10.1109/ACCESS.2018.2889754

Yu, C., Li, J., Li, X., Ren, X., & Gupta, B. B. (2018). Four-image encryption scheme based on quaternion Fresnel transform, chaos and computer generated hologram. *Multimedia Tools and Applications*, *77*(4), 4585–4608. doi:10.1007/s11042-017-4637-6

Yu, S., Ren, K., Lou, W., & Li, J. (2009, September). Defending against key abuse attacks in KP-ABE enabled broadcast systems. In *Proceedings of the International Conference on Security and Privacy in Communication Systems* (pp. 311-329). Springer. doi:10.1007/978-3-642-05284-2_18

Yu, S., Wang, C., Ren, K., & Lou, W. (2010, March). Achieving secure, scalable, and fine-grained data access control in cloud computing. In Proceedings IEEE INFOCOM 2010 (pp. 1-9). IEEE. doi:10.1109/INFCOM.2010.5462174

Yu, Z., Gao, C. Z., Jing, Z., Gupta, B. B., & Cai, Q. (2018). A practical public key encryption scheme based on learning parity with noise. *IEEE Access*, *6*, 31918–31923. doi:10.1109/ACCESS.2018.2840119

Zheng, Q., Xu, S., & Ateniese, G. (2014, April). VABKS: verifiable attribute-based keyword search over outsourced encrypted data. In *Proceedings of the IEEE INFOCOM 2014-IEEE Conference on Computer Communications* (pp. 522-530). IEEE. doi:10.1109/INFOCOM.2014.6847976

Mamta is a Ph.D. Scholar under the Supervision of Dr. B. B. Gupta in the Department of Computer Engineering at National Institute of Technology (NIT), Kurukshetra, India. Her research interests include number theory and cryptography, searchable encryption, information security, and cloud computing. She has done her M. Tech. (Computer Engineering) from the Department of Computer Engineering at National Institute of Technology (NIT), Kurukshetra, India. B. B. Gupta received PhD degree from Indian Institute of Technology Roorkee, India in the area of information security. He has published more than 50 research papers in international journals and conferences of high repute. He has visited several countries to present his research work. His biography has published in the Marquis Who's Who in the World, 2012. At present, he is working as an Assistant Professor in the Department of Computer Engineering, National Institute of Technology Kurukshetra, India. His research interest includes information security, cyber security, cloud computing, web security, intrusion detection, computer networks, and phishing.

B. B. Gupta received PhD degree from Indian Institute of Technology Roorkee, India in the area of information security. He has published more than 50 research papers in international journals and conferences of high repute. He has visited several countries to present his research work. His biography has published in the Marquis Who's Who in the World, 2012. At present, he is working as an Assistant Professor in the Department of Computer Engineering, National Institute of Technology Kurukshetra, India. His research interest includes information security, cyber security, cloud computing, web security, intrusion detection, computer networks and phishing. He is the corresponding author for this paper. (gupta.brij@gmail.com)