# Delegated Access Control For Secure Sharing Based Cloud Computing Environment

P.U.Lekshmi Dhivya[#1], Dr.J.ArokiaRenjith[*2]
*Department of Computer Science and Engineering , Anna University*
*Jeppiaar Engineering College,Chennai,TamilNadu,India.*
[1]dhivipadmanabhan18@gmail.com
[2]arokiarenjith@gmail.com

***Abstract*-**Adoption of cloud services that have biggest obstacles is that security challenges when considering it. Current approach is that hosting the confidential data in cloud based on fine-grained encryption of data by fine grained access control. In this approach, before uploading the data, data owners should encrypt the data and whenever the user changes. Data owners incur high communication and computation costs. To achieve security and confidentiality, we make use of approach base don't layers of encryption, that addresses such requirement. Under this approach, the data owner performs a coarse-grained encryption, where in cloud performs a fine-grained encryption on to the owner encrypted data. We utilize an efficient group key management scheme that supports ACPs. Our system as sure the confidentiality of the data and preserves the privacy of users from the cloud while delegating most of the access control enforcement to the cloud.

*Index Terms*—Privacy, Identity, Cloud Computing, Policy Decomposition, Encryption, Access Control

**INTRODUCTION:**

The data storage is mainly used for privacy and security challenges, during the adoption of cloud technologies. The encryption assures the data confidentiality against the cloud (ACPs) these ACPs are often expressed in terms of the properties of the users, referred to as *identity attributes*. Such approach, referred to *attribute based access control* (ABAC). As shown in Figure those approaches group data items based on ACPs and encrypt each group with a different symmetric key. Such approaches have some limitations:

- Whenever user dynamics changes, as the data owner does not keep a copy of the data, the data owner needs to down load and decrypt the data.
- Private communication channels with the users should be established by the data owner to issue the new keys to the users.

The cloud can learn sensitive information about the users and their organization and are in efficient in supporting fine-grained ABAC policies

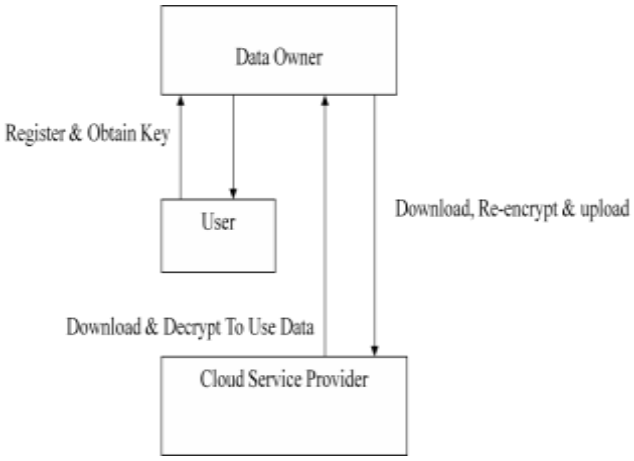



**Figure1:TraditionalApporoach**

Recentlyproposedapproachesbasedonbroadcastkeymanagementschemesaddresssomeoftheabovelimitations. Wemakeuseof*singlelayerencryption*(SLE)approachesandtheyrequirethedataownertoenforceaccesscontrolthroughencryptionperformedatthedataowner.SLEassuresthe privacyoftheusers andsupports fine-grainedACPs.SLEstillrequiresthedataownertoenforce*all*theACPsbyfine-grained encryption, both initially and

subsequently after users are added/revoked In order to delegate as much access control enforcement as possible to the cloud, one needs to decompose the ACPs such that the data owner manages minimum number of attribute conditions in those ACPs that assures the confidentiality of data from the cloud. Each ACP should be decomposed to two sub ACPs such that the conjunction of the two sub ACPs result in the original ACP. The two encryptions to gather enforce the ACP as users should perform two decryptions to access the data.

Example: We use the following running example where a hospital (Owner) supports fine-grained access control on electronic health records (EHRs) and makes these records available to hospital employees (Usrs) through a public cloud (Cloud). Typical hospital employees includes Users playing different roles such as receptionist(rec), cashier(cas), doctor(doc), nurse(nur), pharmacist(pha), and system administrator(sys). An HER document consists of data items including Billing Info(BI), Contact Info(CI), Medication-Report(MR), Physical Exam(PE), Lab Reports(LR), Treatment Plan(TP) and soon. In accordance with regulations such as health insurance portability and accountability act (HIPAA), the hospital policies specify which users can access which data item(s). In our example system, there are four attributes, role (rec, cas, doc, nur, pha, sys), insurance plan, denoted as ip,(ACME, MedA, MedB, MedC), type (assistant, junior, senior) and year of service, denoted as yos,(integer). The system has the ACP("role=doc"∧"ip=2-out-4")∨("role=doc"∧ "yos≥2").The ACP can be decomposed as two sub ACPs "role=doc" and "ip=2-out-4"∨"yos≥2".

## 2. BUILDINGBLOCKS:

In this section we have, overview of SLE approach which is used as the base mode lf or comparison with the TLE approach proposed in this paper.

### 2.1. Single Layer Encryption Approach

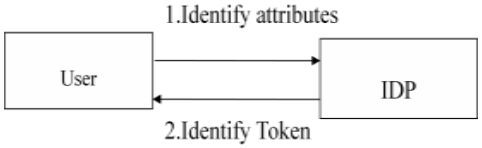
The SLE approach consists of the four entities**: Owner, Usr(User), IdP (Identity Provider) and Cloud**. They play the following roles:

**Owner**: The data owner defines ACPs, and uploads encrypted data to the Cloud and the Cloud hosts encrypted data of the Owner.

**IdP**: The identity provider, a trusted third party, issues *identity tokens* to users based on the attributes that users have. IdPs issue identity tokens in the same format.

**Usr**: The user uses one or more identity tokens to gain access to the encrypted data hosted in the Cloud.

As shown in Figure2, the SLE approach follows the conventional data outsourcing scenario where the Owner enforces all ACPs through elective encryption and uploads encrypted data to the un-trusted Cloud. The system goes through five different phases:




**Identity token issuance:** IdPs issue identity tokens to Users based on their identity attributes.

**Identity token registration:** Users register all their identity tokens to obtain secrets in order to later decrypt the data that they are allowed to access.

**Data encryption and uploading:** Based on the secrets issued and the ACPs, the Owner encrypts the data using the keys generated using the AB-GKM::Key Gen algorithm and uploads to the Cloud.

**Data downloading and decryption:** Encrypted data are downloaded by the user from the Cloud and decrypt using the key derived from the AB-GKM::KeyDer algorithm

**Encryption evolution management:** Over time, either access control police so user may change. Already encrypted data may go through frequent updates. In such situations, it should be re-encrypt already encrypted data. The Owner alone is responsible to perform such encryptions. The Owner downloads all affected data from the Cloud, decrypts the mend then follows the data encryption and upload step.

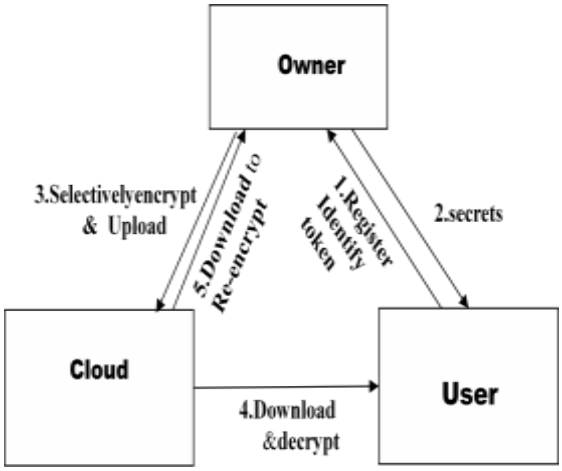
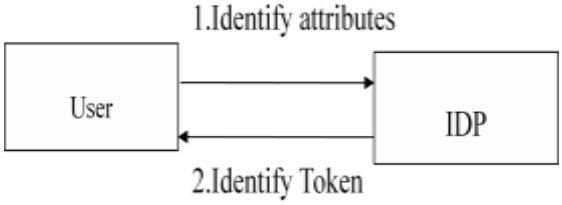



**Figure2: Single LayerEncryption**

### 3. OVERVIEW:

Our solution to the problem of delegated access control to out sourced data in the cloud. Like the SLE system described we have another system called the TLE (Tow Layer Encryption) system. This TLE system consists of the four entities, **Owner, Usr, IdPand Cloud.** The SLE approach, the Owner and the Cloud collectively enforce ACPs by performing two encryptions one ach data item. This two layer enforce mental lows one to reduce the load on the Owner and delegates as much access control enforcement duties as possible to the Cloud. Overview of six phase is shown below:




**Identity token issuance:** IdPs issue identity tokens to Users based on their identity attributes.

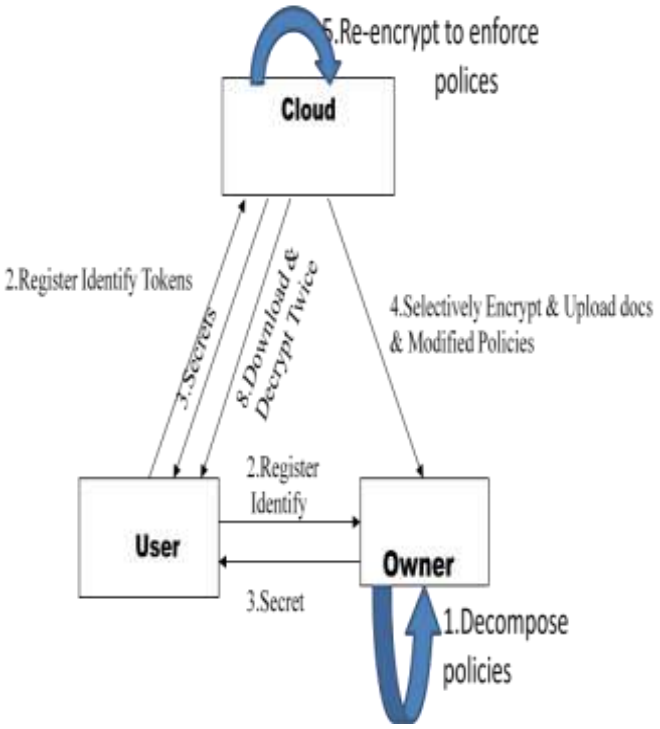



**Figure3: TWO LAYER ENCRYPTION**

**Policy decomposition:** The Owner decomposes each ACP in to at most two sub ACPs such that the Owner enforces the minimum number of attributes to assure confidentiality of data from the Cloud. It is important to make sure that the decomposed ACPs are consistent so that the sub ACPs together enforces the original ACPs.

**Identity token registration:** Users register their identity tokens in order to obtain secrets to decrypt the data that they are allowed to access. Uses register only those identity tokens related to the Owner's sub ACPs and register the remaining identity tokens with the Cloud in a privacy preserving manner.

**Data encryption and uploading:** The Owner first encrypts the data based on the Owner's sub ACPs in order to hide the content from Cloud and then uploads them along with the public information generated by the AB-GKM::KeyGen algorithm and the remaining sub ACPs to the Cloud.

**Data downloading and decryption:** Users download encrypted data from the Cloud and decrypt the data using the derived keys. Users decrypt twice to first

remove the encryption layer added by the Cloud and then by the Owner. As access controls enforced through encryption.

**Encryption evolution management:** Overtime, user credentials may change. Further, already encrypted data may go through frequent updates. In such situations, data already encrypted must before encrypted with any key.

**Attribute Based Encryption (ABE)**

The proposed system provides secure patient-centric PHR access and efficient key management at the same time. The key idea is to divide the system into multiple security domains (namely, *public domains* (PUDs) and *personal domains* (PSDs)) according to the different users' data access requirements. The PUDs consist of users who make access based on their professional roles, such as doctors, nurses and medical researchers. In practice, a PUD can be mapped to an independent sector in the society, such as the health care, government or insurance sector. For each PSD, its users are personally associated with a data owner (such as family members or close friends), and they make accesses to PHRs based on access rights assigned by the owner. In both types of security domains, weutilize ABE to realize cryptographically enforced, patient-centric PHR access. Especially, in a PUD multi-authority ABE is used, in which there are multiple "attribute authorities" (AAs), each governing a disjoint subset of attributes. *Role attributes* are defined for PUDs, representing the professional role or obligations of a PUD user. Users in PUDs obtain their attribute-based secret keys from the AAs, without directly interacting with the owners. Since the users are personally known by the PHR owner, to realize patient-centric access, the owner is at the best position to grant user access privileges on a case-by-case basis. For PSD, *data attributes* are defined which refer to the intrinsic properties of the PHR data, such as the category of a PHR file. For the purpose of PSD access, each PHR file is labelled with its data attributes, while the key size is only linear with the number of file categories a user can access. Since the number of users in a PSD is often small, it reduces the burden for the owner. When encrypting the data for PSD, all that the owner needs to know is the intrinsic data properties. The multi-domain approach best models different user types and access requirements in a PHR system. The use of ABE makes the encrypted PHRs self-protective, i.e., they can be accessed by only authorized users even when storing on a semi-trusted server, and when the owner is not online. Figure 2 shows the proposed system architecture for data access and encryption.

**4. SYSTEM ARCHITECTURE**

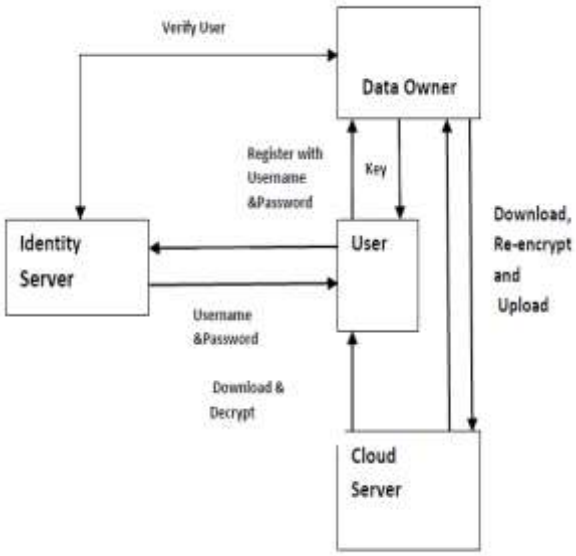



**Figure 4: Proposed System Architecture**

**PHR Encryption and Access**

The owners upload ABE-encrypted PHR files to the server. Each owner's PHR file is encrypted both under a certain fine grained and role-based access policy for users from the PUD to access, and under a selected set of data attributes that allows access from users in the PSD. Only authorized users can decrypt the PHR files, excluding the server. For improving efficiency, the data attributes will include all the intermediate file types from a leaf node to the root. For example, in Fig. 2, an "allergy" file's attributes are *{PHR, medical history, allergy}*. The data readers download PHR files from the server, and they can decrypt the files only if they have suitable attribute based keys. The data contributors will be granted write

access to someone's PHR, if they present proper write keys.

**User Revocation**

Here we consider revocation of a data reader or her attributes/access privileges. There are several possible cases: 1) revocation of one or more role attributes of a public domain user 2) revocation of a public domain user which is equivalent to revoking all of that user's attributes 3) Revocation of a personal domain user's access privileges 4) revocation of a personal domain user

**Policy Updates**

A PHR owner can update her sharing policy for an existing PHR document by updating the attributes (or access policy) in the ciphertext. The supported operations include add/delete/modify, which can be done by the server on behalf of the user.

**Break-glass Access**

When an emergency happens, the regular access policies may no longer be applicable. To handle this situation, break-glass access is needed to access the victim's PHR. In our proposed system, each owner's PHR's access right is also delegated to an emergency department (ED). To prevent from abuse of break-glass option, the emergency staff needs to contact the ED to verify her identity and the emergency situation, and obtain temporary read keys. After the emergency is over, the patient can revoke the emergent access via the ED.

**5. POLICY COVER**

We define the *policy cover problem* as the optimization problem off including the minimum number of attribute conditions that "covers" *all* the ACPs in the ACPB. We say that a set of attribute conditions covers the ACPB if in order to satisfy *any* ACP in the ACPB, it is necessary that at least one of the attribute conditions in the set is satisfied. We call such as to attribute conditions as the attribute condition cover.

**Example:** If ACPB consists of the three simple ACPs{"role = doc" ∧"ip = 2-out-4", "role = doc" ∧"yos≥2", "role = nur" }, the minimum set of attributes that covers ACPB is {"role = doc", "role = nur"}. "role =doc" should be satisfied in order to satisfy the first two ACPs. Notice that satisfying "role = doc" is not sufficient to satisfy the ACPs. The set is minimum since the set obtained by removing either "role = doc" or "role = nur" does not satisfy the cover relationship. T the related decision problem as follows.

**Algorithm 1** GEN-GRAPH
1: C = ϕ
2: **for** Each ACPi∈ ACPB, i = 1 to Np**do**
3: ACP_i← Convert ACPi to DNF
4: **for** Each conjunctive term c of ACP_ i**do**
5: Add c to C
6: **end for**
7: **end for**
8: //Represent the conditions as a graph
9: G = (E, V ), E = ϕ, V = ϕ
10: **for** Each conjunctive term ci ∈ C, i = 1 to Nc**do**
11: Create vertex v, if v _∈V , for each AC in ci
12: Add an edge ei between vi and each vertex already added for ci
13: **end for**
14: Return G

**Algorithm 2** RANDOM COVER
1: G = GEN-GRAPH(ACPB)
2: ACC = ϕ
3: **for** Each disconnected subgraphGi = (Vi,Ei) of G **do**
4: **if** |Vi| == 1 **then**
5: Add ACi corresponding to the vertex to ACC
6: **else**
7: **while** Ei_= ϕ **do**
8: Select a random edge (u, v) of Ei
9: Add the attribute conditions ACu and Acv corresponding to {u, v} to ACC.
10: Remove from Ei every edge incident on either u or
11: **end while**
12: **end if**
13: **end for**
14: Return ACC

**Example**: The algorithm is similar to the random cover algorithm 2, except that instead of randomly selecting the edges to be included in the cover, it selects the vertex of highest degree and removes all of its incident edges. The following is the complete set of ACPs of the hospital :

(“role = rec” ∨(“role = nur” ∧“type ≥ junior”), CI)
(“role = cas” ∨“role = pha”, BI)
(“role = doc” ∧“ip = 2-out-4”, CR)
((“role = doc” ∧“ip = 2-out-4”) ∨“role = pha”, TR)
((“role = doc” ∧“ip = 2-out-4”) ∨(“role = nur” ∧“yos≥ 5”) ∨
“role = pha”, MR)
((“role = nur” ∧“type ≥ junior”) ∨(“role = dat” ∧“type ≥ junior”) ∨(“role = doc” ∧“yos≥ 2”), LR) ((“role = nur” ∧“type = senior”) ∨(“role = dat” ∧“yos≥ 4”),PE)

**POLICY DECOMPOSITION**

The SLE approach, the Owner in curs a high communication and computation over head since it has to manage all the authorizations when user dynamics change. Since the Cloud is not trusted for the confidentiality of the out sourced data, the Owner has to initially encrypt the data and upload the encrypted data to the cloud. May be to enforce authorization policies of cloud through encryption and avoid re-encryption by the Owner, the data ought to be encrypted with two encryption layers. We call the two encryption layers as *inner encryption layer* (IEL) and *outer encryption layer* (OEL). The IEL assures the confidentiality of the data The OEL is for fine grained authorization for controlling accesses to the data by the users and is generated by the Cloud. TLE approach Is how to distribute the encryptions between the Owner and the Cloud. There are two possible extremes. The first approach is for the Owner to encrypt all data it misusing a single symmetric key and let the Cloud perform the complete access control related encryption. This approach is for the Owner and the Cloud to perform the complete access control related encryption twice. The first approach has the least over head for the Owner as the Owner does not manage any attributes and perform fine grained access control related encryption.

**Algorithm3** POLICY-DECOMPOSITION
1: ACPB Owner = ϕ
2: ACPBCloud = ϕ
3: **for** Each ACPi in ACPB **do**
4: Convert ACPi to DNF
5: ACPi(owner) = ϕ
6: ACPi(cloud) = ϕ
7: **if** Only one conjunctive term **then**
8: Decompose the conjunctive term c into c1 and c2 such that ACs in c1 ∈ ACC, ACs in c2 _∈ ACC and c = c1 ∧ c2
9: ACPi(owner) = c1
10: ACPi(cloud) = c2
11: **else if** At most one term has more than one AC **then**
12: **for** Each single AC term c of ACP_i**do**
13: ACPi(owner) ∨= c
14: ACPi(cloud) ∨= c
15: **end for**
16: Decompose the multi AC term c into c1 and c2 such that ACs in c1 ∈ ACC, ACs in c2 _∈ ACC and c = c1 ∧ c2
17: ACPi(owner) ∨= c1
18: ACPi(cloud) ∨= c2
19: **else**
20: **for** Each conjunctive term c of ACP_i**do**
21: Decompose c into c1 and c2 such that ACs in c1 ∈ ACC, ACs in c2 _∈ ACC and c = c1 ∧ c2
22: ACPi(owner) ∨= c1
23: **end for**
24: ACPi(cloud) = ACP_i
25: **end if**
26: Add ACPi(owner) to ACPB Owner
27: Add ACPi(cloud) to ACPB Cloud
28: **end for**
29: Return ACPB Owner and ACPB Cloud

Algorithm 3 takes the ACPB and ACC as input and produces the two sets of ACPs ACPBOwner and ACPBCloud It first converts each policy into DNF and decompose each conjunctive term into two conjunctive terms(ie)the conjunction of corresponding sub ACPs in ACPB Owner and ACPB Cloud produces an original ACP in ACPB.

Example 2:

For our example ACPs, the Owner handles the following sub ACPs.

(“role = rec” V“role = nur” , CI)
(“role = cas” V“role = pha”, BI)
(“role = doc”, CR)
(“role = doc” V“role = pha”, TR)
(“role = doc” V“role = nur” V“role = pha”, MR)
(“role = nur” V“role = dat” V“role = doc”, LR)
(“role = nur” V“role = dat”, PE)

As shown in Algorithm 3, the Owner re-writes the ACPs that the Cloud should enforce conjunction of the two decomposed sub ACPs yields an original ACP. In our example, the sub ACPs that the Cloud enforces look like follows.

(“role = rec” V“type ≥ junior”, CI)
(“role = cas” V“role = pha”, BI)
(“ip = 2-out-4”, CR)(“ip = 2-out-4” V“role = pha”, TR)
((“role = doc” Λ“ip = 2-out-4”) V(“role = nur” Λ“yos≥ 5”) V “role = pha”, MR)
((“role = nur” Λ“type ≥ junior”) V(“role = dat” Λ“type ≥ junior”) V(“role = doc” Λ“yos≥ 2”), LR)
((“role = nur” Λ“type = senior”) V(“role = dat” Λ“yos≥ 4”),PE)

## 6. RESULT AND ANALYSIS:

In this section, we compare the SLE and the TLE approaches, and then give a high level analysis of the security and the privacy of both approaches

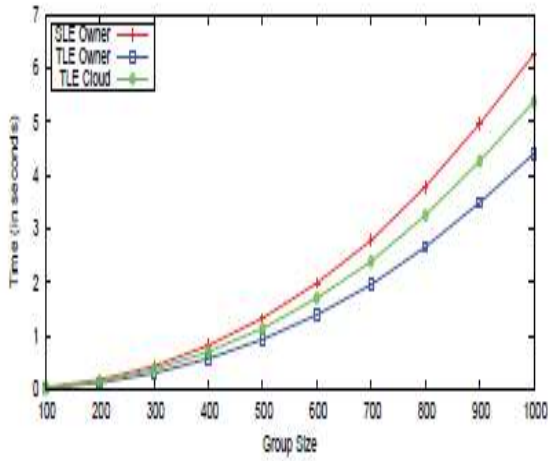



**Figure 5:** Average Time to Generate Keys for Tow approaches.

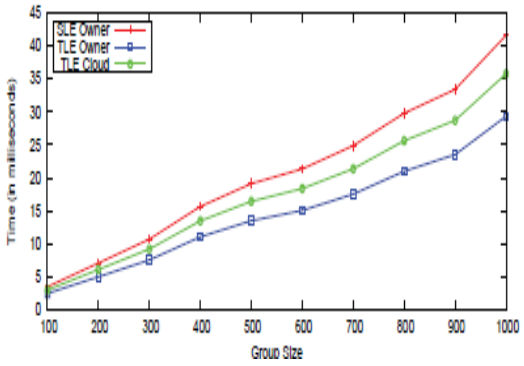



**Figure 6**: Average Time To Drive keys For the Tow Approaches

### 6.1 SLE vs. TLE

The SLE approach, the Owner enforces all ACPs by fine-grained encryption. If the system dynamics change, the Owner updates the keys and encryptions. The SLE approach incurs high overhead and the Owner incurs a high overhead in communication and computation.

The TLE approach reduces the overhead incurred by the Owner during the initial encryption as well as subsequent re-encryptions. In this approach, the Owner handles the minimal set of attribute conditions. TLE reduces the communication and computation overhead at the Owner.

### 6.2 Security and Privacy

The SLE approach correctly enforces the ACPs through encryption. In the SLE approach, the Owner itself performs the attribute based encryption based on ACPs. The scheme makes sure that only those Usrs who satisfy the ACPs can derive the encryption keys. Therefore, only the authorized Usrs are able to access the data.

## 7. RELATED WORK:

### 7.1 Fine-grained Access Control:

Fine-grained access control (FGAC) allows one to enforce selective access to the content based on expressive policy specifications. Research in FGAC can be categorized into two dissemination models: *push-based* and *pull-based* models. Our work focuses on the

pull-based model. In the push-based approaches subdocuments are encrypted with different keys, which are provided to users at the registration phase. The encrypted subdocuments are then broadcasted to all users.

**7.2 Attribute Based Encryption:**

The concept of attribute based encryption (ABE) has been introduced by Sahai and Waters. The initial ABE system is limited only to threshold policies in which there are at least k out of n attributes common between the attributes used to encrypt the plaintext and the attributes users possess. Pirretti et al gave an implementation of such a threshold ABE system using a variant of the Sahai-Waters Large Universe construction .Since this initial threshold scheme, a few variants have been introduced to provide more expressive ABE systems

**8. CONCLUSIONS:**

Current approaches to enforce ACPs on outsourced data using selective encryption require tomanage all keys and encrypts data to upload to the remote storage. This approaches have high communication and computation cost to manage keys and encryptions whenever user change. A key problem is how to decompose ACPs so that the Owner has to handle a minimum number of attribute conditions. Our approach is based on a privacy preserving attribute based key management scheme that protects the privacy of users while enforcing attribute based ACPs. As the experimental results show, decomposing the ACPs and utilizing the two layer of encryption reduce the overhead at the Owner. As future work is to investigate the alternative choices for the TLE approach and to reduce the computational cost by exploiting partial relationships among ACPs.

**9. REFERENCE:**

[1] M. Nabeel and E. Bertino, “Privacy preserving delegated access control in the storage as a service model,” in EEEInternational Conference on Information Reuse and Integration (IRI), 2012.

[2] E. Bertino and E. Ferrari, “Secure and selective dissemination of XML documents,” ACM Trans. Inf. Syst. Secure., vol. 5, no. 3, pp. 290–331, 2002.

[3] G. Miklau and D. Suciu, “Controlling access to published data using cryptography,” in VLDB ’2003: Proceedings of the29th international conference on Very large data bases. VLDB Endowment, 2003, pp. 898–909.

[4] N. Shang, M. Nabeel, F. Paci, and E. Bertino, “A privacypreserving approach to policy-based content dissemination,”inICDE ’10: Proceedings of the 2010 IEEE 26th International Conference on Data Engineering, 2010.

[5] M. Nabeel, N. Shang, and E. Bertino, “Privacy preserving policy based content sharing in public clouds,” IEEE Transactionson Knowledge and Data Engineering, 2012.