IJCSEC- International Journal of Computer Science and Engineering Communications.
Vol.3, Issue 3, 2015, Page.1026-1033, ISSN: 2347–8586

# Enforcing Access Control to Cloud With Preserved Users Privacy

**[1]Sabitha.D, [2]M.C. Babu**
[1] PG Scholar, [2] Assistant Professor,
Department of Computer Science, St. Peter's University, Chennai, India.
*[1] dsabithadevaraj@gmail.com, [2] mcbabu@yahoo.com*

## Abstract

Access Control Policies defines the user roles and their access rights to the confidential data. Fine-grained access control on confidential data hosted in the cloud are based on fine-grained encryption of the data in which data owners are in charge of encrypting the data before uploading them to the cloud and re-encrypting the data whenever user credentials change. When data owners perform the re-encryption they incur high communication and computation costs. To reduce the overhead at data owner, delegate the enforcement of access control to cloud, while assuring data confidentiality from the cloud. In order to delegate access control to cloud, an approach of two layers of encryption is proposed, in which the data owner performs a lower level encryption; whereas the cloud performs a higher level encryption. Using Policy Decomposition algorithm, decompose the ACP between the owner and cloud to perform the two layers of encryption. With TLE, the system guarantees the confidentiality of the data from cloud and preserves the privacy of users from the cloud while delegating most of the access control enforcement to the cloud.

***Keywords****: Privacy, Identity Attribute, Cloud Computing, Policy Decomposition, Encryption, Access Control.*

## 1. Introduction

Security and privacy represents the major concerns in the adoption of cloud technologies for data storage. An approach to mitigate these concerns is the use of encryption. However, encryption assures the confidentiality of the data against the cloud, the use of conventional encryption approaches is not sufficient to support the enforcement of fine-grained organizational access control policies (ACPs). Many organizations have today ACPs regulating which users can access which data; these ACPs are often expresses in terms of the properties of the users, referred to as identity attributes. Such an approach, referred to as attribute- based access control (ABAC), supports fine-grained access control which is crucial for high-assurance. Supporting ABAC over encrypted data is a critical requirement in order to utilize cloud storage services for selective data sharing among different users. Notice that often user identity attributes encode private information and should be strongly protected from the cloud, very much as the data themselves. The approach that overcomes the shortcomings of fine-grained encryption and supports ABAC policy is based on two layers of encryption applied to each data item uploaded to the cloud.

The data owner performs a coarse grained encryption over the data in order to assure the confidentiality of the data from the cloud. Then the cloud performs fine grained encryption over the encrypted data provided by the data owner based on the ACPs provided by the data owner. A challenging issue in the TLE approach is how to decompose the ACPs so that fine-grained ABAC enforcement can be delegated to the cloud while at the same time the privacy of the identity attributes of the users and confidentiality of the data are assured. In order to delegate as much as access control enforcement as possible to the cloud, one need 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 results in the original ACP. The two layer encryption should be performed such that the data owner first encrypts the data based on one set of sub ACPs and the cloud re-encrypts the encrypted data using the other set of ACPs. The two encryptions together enforce the ACP as users should perform two decryptions to access the data. The TLE approach has many advantages. When the policy or user dynamics changes then only the outer layer of encryption needs to be updated. Since the outer layer encryption is performed at the cloud, no data transmission is required between the data owner and the cloud.

## 2. Group Key Management

Early approaches to Group Key Management (GKM) rely on a key server to share a secret with users to distribute decryption keys. Such approaches do not efficiently handle join and leave operations, as in order to achieve forward and backward security, they require sending O(n) private rekey information, where n is the number of users.

### 2.1 Privacy Preserving Attribute Based Group Key Management

The privacy preserving attribute based group key management (PP AB-GKM) scheme combines the work on AB-GKM scheme and privacy preservation in Broadcast Group key Management (BGKM).

The BGKM schemes are a special type of GKM scheme where the rekey operation is performed with a single broadcast without requiring the use of private communication channels. Unlike conventional GKM schemes, the BGKM schemes do not give users the private keys. Instead users are given a secret which is combined with public information to obtain the actual private keys. Such schemes have the advantage of requiring a private communication only once for the initial secret sharing. The subsequent rekeying operations are performed using one broadcast message. Further, in such schemes achieving forward and backward security requires only changing the public information and does not affect the secret shares given to existing users.

**Definition (PP AB-GKM):** The PP AB-GKM scheme consists of five algorithms: Setup, SecGen, KeyGen, KeyDer, and ReKey. An abstract description of these algorithms is given below.

**Setup($\ell$, N, $N_a$)**: It takes the security parameter $\ell$, the maximum group size N, and the number of attribute conditions $N_a$ as input, initializes the system.It invokes BE::Setup($\ell$,N), OCBE::Setup(l) and ACV-BGKM::Setup($\ell$,N) algorithms [15],[5],[11],[9].

**SecGen ($\gamma$)**: The secret generation algorithm gives a user $Usr_j$, $1 \leq j \leq N$ a set of secrets for each commitment $com_i \in \gamma, 1 \leq i \leq m$. It invokes BE::GetSecGen () and OCBE::GetData algorithms [15],[5].

**KeyGen (ACP)**:The key generation algorithm takes the access control policy ACP as the input and outputs a symmetric key K, a set of public information tuples PI and an access tree T. It invokes BE::GetCover() and ACV-BGKM::KeyGen algorithms [15],[11],[9].

**KeyDer ($\beta$, PI, T)**: Given the set of identity attributes $\beta$, the set of public information tuples PI and the access tree T, the key derivation algorithm outputs the symmetric K only if the identity attributes in $\beta$ satisfy the access structure T. It invokes BE::KeyDer and Acv-BGKM::KeyDer algorithms [15],[11].

**ReKey (ACP)**: The rekey algorithm is similar to the KeyGen algorithm. It is executed whenever the dynamics in the system change.

## 3. Single Layer Encryption approach

The SLE scheme consists of the four entities, owner, user, Identity Provider and cloud. They play the following roles:

- Owner, the data owner defines ACPs, and uploads encrypted data to the cloud, the cloud storage service.
- Cloud, hosts the encrypted data of the owner.
- IdP, the identity provider, a trusted third party, issues identity tokens to users based on the attributes users have. An identity token is a singed Pedersen commitment that binds the identity attribute value to a user while hides it from others.
- User, uses one or more identity tokens to gain access to the encrypted data hosted in the Cloud.

As shown in Figure 1, the SLE approach follows conventional data outsourcing scenario where the owner enforces all ACPs through selective encryption and uploads encrypted data to the untrusted Cloud. The system goes through five different phases. An overview of the five phases is given below:

**(i) Identity token issuance**

IdPs issue identity tokens to users based on their identity attributes.

**(ii) Identity token registration**

Users register their identity tokens to obtain secrets in order to later decrypt the data that they are allowed to access.

**(iii) Data encryption and uploading**

Based on the secrets issued and Access Control Policy, the owner encrypts data using the keys generated from AB-GKM:: KeyGen algorithm and uploads data to the cloud.

**(iv) Data downloading and decryption**

Users download encrypted data from the cloud and decrypt using the key derived from the AB-GKM::KeyDer algorithm.

**(v) Encryption evolution management**

Overtime, either access control policies or user credentials may change. Further, already encrypted data may go through frequent updates. In such situations, it may be required to re-encrypt already encrypted data. The owner alone is responsible to perform such re-encryptions. The owner downloads all affected data from the cloud, decrypts them and then follows the data encryption and upload step.

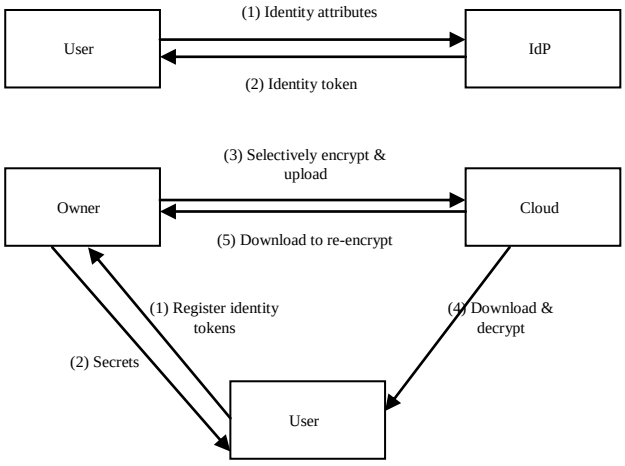



**Fig. 1**: Single layer encryption approach

## 4. Policy Decomposition

In SLE approach, the owner incurs a high communication and computation overhead since it has to manage all the authorizations when user dynamics or ACP changes. If the access control related encryption is somehow delegated to the cloud, the owner can be freed from the responsibility of managing authorizations through re-encryption and the overall performance would thus improve. Since the cloud is not trusted for the confidentiality of the outsourced data, the owner has to initially encrypt the data and upload the encrypted data to the cloud. Therefore, in order for the cloud to allow enforcing authorization policies through encryption and avoiding re-encryption by the owner, the data may have to be encrypted again to have two encryption layers, called as inner encryption layer (IEL) and outer encryption layer(OEL). IEL assures the confidentiality of the data with respect to the cloud and is generated by the owner. The OEL is for fine grained authorization for controlling accesses to the data by the users and is generated by the cloud.

An important issue in the 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 the data items using a single symmetric key and let the cloud perform the complete access control related encryption. The second approach is for the owner and the cloud to perform the complete access control related encryption twice. The first approach has the least overhead for the owner, but it has the highest information exposure risk due to collusions between users and the cloud. Further, IEL updated require re-encrypting all data items. The second approach has the least information exposure risk due to collusions, but it has the highest overhead on the owner as the owner has to perform the same task initially as in the SLE approach and, further, needs to manage all identity attributes. An alternative solution is based on decomposing ACPs so that the information exposure risk and key management overhead are balanced. The problem is then how to decompose the ACPs such that the owner has to manage the minimum number of attributes while delegating as much access control enforcement as possible to the cloud without allowing it to decrypt the data.

### 4.1 Policy Cover

The 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. This set of attribute conditions are called attribute condition cover. For example, if ACPB consists of the three simple ACPs $\{C_1 \wedge C_2, C_2 \wedge C_3, C_4\}$, the minimum set of attributes that covers ACPB is $\{C_2, C_4\}$. C2 should be satisfied in order to satisfy the ACPs $C_1 \wedge C_2$ and $C_2 \wedge C_3$. But, satisfying C2 is not sufficient to satisfy the ACPs. Minimum is the set since the set obtained by removing either $C_2$ or $C_4$ does not satisfy the cover relationship.

**Algorithm 1** GEN-GRAPH

*1: C=φ*
*2:* ***for*** *Each $ACP_i \in ACPB$, i= 1 to $N_P$ do*
*3: $ACP_i \leftarrow$ Convert $ACP_i$ to DNF*
*4:* ***for*** *Each conjunctive term c of $ACP_i$ do*
*5: Add c to C*
*6:* ***end for***
*7:* ***end for***

*8: G = (E, V), E = φ, V = φ*
*9:* ***for*** *Each conjunctive term $c_i \in C$, i = 1 to $N_c$ do*
*10: Create vertex v, if v ?∈ V, for each AC in $c_i$*
*11: Add an edge $e_i$ between $v_i$ and each vertex already added for $c_i$*
*12:* ***end for***
*13: Return G*

GEN-GRAPH algorithm 1 takes the ACPB as the input and converts each ACP into DNF (disjunctive normal form). The unique conjunctive terms are added to the set C. For each attribute condition in each conjunctive term in C, it creates a new vertex in G and adds edges between the vertices corresponding to the same conjunctive term. Depending on the ACPs, the algorithm may create a graph G with multiple disconnected sub graphs.
As shown in APPROX-POLICY-COVER1 algorithm 2, it takes the ACPB as the input and outputs a near-optimal attribute condition cover ACC. First the algorithm converts the ACPB to a graph G as shown in GEN-GRAPH algorithm 1. Then for each disconnected subgraph $G_i$ of G, it finds the near optimal attribute condition cover and add to the ACC. The attribute condition to be added is related at random by selecting a random edge in $G_i$. Once an edge is considered, all its incident edges are removed from $G_i$. The algorithm continues until all edges are removed from each $G_i$. The running time of the algorithm is O(V+E) using adjacency lists to represent G. It can be shown that the APPROX-POLICY-COVER1 algorithm is a polynomial-time 2-approximation algorithm.
The second approximation algorithm APPROX-POLICY-COVER2 uses a heuristic to select the attribute conditions. This algorithm is similar to APPROX-POLICY-COVER1 algortithm2 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.

**Algorithm 2** APPROX-POLICY-COVER1
*1: G = GEN-GRAPH(ACPB)*
*2: ACC = φ*
*3:* ***for*** *Each disconnected subgraph $G_i = (V_i, E_i)$ of G do*
*4:* ***if*** *$|V_i| == 1$ then*
*5: Add $AC_i$ corresponding to the vertex to ACC*
*6:* ***else***
*7:* ***while****$E_i$?=φ do*
*8: Select a random edge (u, v) of $E_i$*
*9: Add the attribute conditions $AC_u$ and $AC_v$ corresponding to {u, v} to ACC*
*10: Remove from $E_i$ every edge incident on either u or v*
*11:****end while***
*12:****end if***
*13:* ***end for***
*14: Return ACC*

**4.2 Policy decomposition**

The owner manages only those attribute conditions in the ACC. The cloud handles the remaining set of attribute conditions, ACB/ACC. The owner re-writes its ACPs such that they cover ACC. In other words, the owner enforces the parts of the ACPs related to the ACs in ACC and cloud enforces the remaining parts of the policies along with some ACs in ACC. The POLICY-DECOMPOSITION algorithm 3 shows how the ACPs are decomposed into two sub ACPs based on the attribute conditions in ACC.

## 5. Two Layer Encryption approach

The TLE system consists of the four entities, Owner, User, Identity Provider (IdPs) and Cloud. The Owner and the cloud collectively enforce ACPs by performing two encryptions on each data item. This two layer enforcement allows one to reduce the load on the Owner and delegate as much access control enforcement duties as possible to the cloud. Specifically, it provides a better way to handle data updates, user dynamics and policy changes. The system diagram of TLE approach is shown in fig. 1. The detailed description of the six phases of the TLE approach is introduced in this section. Let the maximum number of users in the system be N, current number of users be n (<N), and the number of attribute
conditions $N_a$.

**Algorithm 3 POLICY-DECOMPOSITION**
*1: $ACPB_{Owner} = \varphi$*
*2: $ACPB_{Cloud} = \varphi$*

*3:* ***for*** *Each $ACP_i$ in ACPB do*
*4: Convert $ACP_i$ to DNF*
*5: $ACP_i(owner) = \varphi$*
*6: $ACP_i(cloud) = \varphi$*
*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: $ACP_i(owner) = c1$*
*10: $ACP_i(cloud) = c2$*
*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: $ACP_i(owner) = c$*
*14: $ACP_i(cloud) = c$*
*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: $ACP_i(owner) \vee = c1$*
*18: $ACP_i(cloud) \vee = c2$*
*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: $ACP_i(owner) \vee = c1$*
*23: end for*
*24: $ACP_i(cloud) = ACP_i$*
*25:* ***end if***
*26: Add $ACP_i(owner)$ to $ACPB_{Owner}$*
*27: Add $ACP_i(cloud)$ to $ACPB_{Cloud}$*
*28:* ***end for***
*29: Return $ACPB_{Owner}$ and $ACPB_{Cloud}$*

### 5.1 Identity token issuance

IdPs are trusted third parties that issue identity tokens to users based on their identity attributes. It should be noted that IdPs need not be online after they issue identity tokens. An identity token, denoted by IT has the format {nym, id-tag, c,σ} where nym is a pseudonym uniquely identifying a user in the system, id-tag is the name of the identity attribute, c is the Pedersen commitment for the identity attribute value x and σ is the IdPs digital signature on nym, id-tag and c.

### 5.2 Policy decomposition

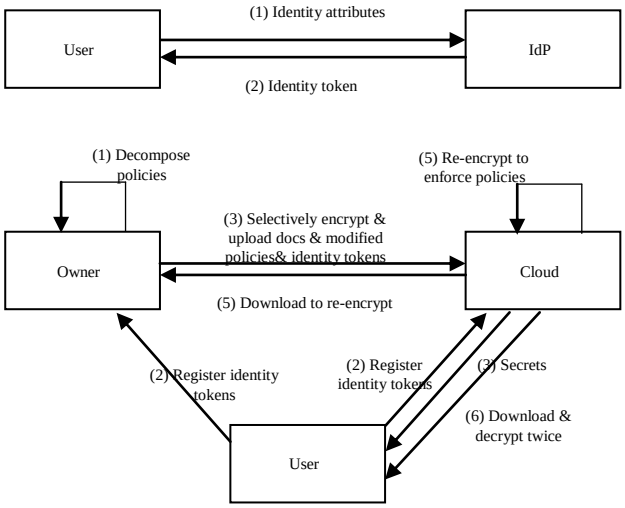



**Fig. 2**: Two Layer encryption approach

Using the policy decomposition algorithm, the Owner decomposes each ACP into two sub ACPs such that the Owner enforces the minimum number of attributes to assure confidentiality of data from the Cloud. The algorithm produces two sets of sub ACPs, $ACPB_{owner}$ and $ACPB_{cloud}$. The owner enforces confidentiality related sub ACPs in $ACPB_{owner}$ and the

cloud enforces the remaining sub ACPs in $ACPB_{cloud}$. It is important to make sure that the decomposed AC Policies are consistent so that the sub ACPs together enforces the original ACPs.

### 5.3 Identity token registration

Users register their identity tokens to obtain the keys on order to later decrypt the data they are allowed to access. Users register only those identity tokens related to the owner's sub ACPs and register the remaining identity token with the cloud in a privacy preserving manner. Cloud provides two decryption keys to user to decrypt twice. It should be noted that the cloud does not learn the identity attributes of the users during this phase.

### 5.4 Data encryption and uploading

The owner first encrypts the data based on the owner's sub ACPs in order to hide the content from the cloud and then uploads them along with the corresponding public information to the cloud. The cloud in turn encrypts the data again based on the sub ACPs in $ACPB_{cloud}$. Both the parties execute ABGKM::KeyGen algorithm individually to first generate the symmetric key, the public information PI and the access tree T for each sub ACP. The detailed description of the encryption process is presented in this section.

The owner arranges the sub ACPs such that each data item has a unique ACP. Note that the same policy may be applicable to multiple data items. Assume the set of data items $D = \{d_1, d_2, \ldots d_m\}$ and the set of sub ACPs $ACPBowner = \{ACP_1, ACP_2, \ldots ACP_n\}$. The owner assigns a unique symmetric key called an ILEkey, $K_i^{ILE}$for each sub $ACP_i \in ACPB_{owner}$, encrypts all the related data with that key and executes the AB-GKM:: KeyGen to generate the public information $PI_i$ and the tree $T_i$. The owner uploads those encrypted data along with the generated $PI_i$ and $T_i$ where i=1, 2, …, n to the cloud. The cloud handles the key management and encryption based access control for the ACPs in $ACPB_{cloud}$. For each sub $ACP_j \in ACPB_{cloud}$, the cloud assigns a unique symmetric key $K_j^{OLE}$ called an OLE key, encrypts each affected data item from owner and produces its information $PI_j$.

### 5.5 Data downloading and decryption

Users download encrypted data from the cloud and decrypt twice to access the data. When the user registered identity token of cloud matches with the identity token received by cloud from owner, two keys are derived. First, the cloud generated pubic information is used to derive the OLE key and then the owner generated public information tuple received from the cloud is used to derive the ILE key using the AB-GKM::KeyDer algorithm thus reducing the waiting time for the user to derive the key from the owner. These two keys allow a user to decrypt a data item only if the user satisfies the original ACP applied to the data item.

### 5.6 Encryption evolution management

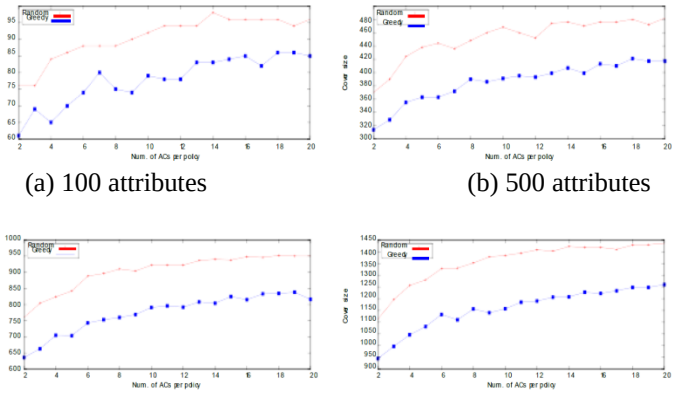
After the initial encryption is performed, affected data items need to be re-encrypted with a new symmetric key if credentials are added/ removed or ACPs are modified. Unlike the SLE approach, when credentials are added or revoked or ACPs are modified, the owner does not have to involve. The cloud generates a new symmetric key and re-encrypts the affected data items. The cloud follows the following conditions in order to decide if re-encryption is required.

1) For any ACP, the new group of users is a strict superset of the old group of users, and backward secrecy is enforced.
2) For any ACP, the new group of users is a strict subset of the old group of users, and forward secrecy is enforced for the already encrypted data items.

## 6. Experimental Results and Analysis

For the experiments, the total number of attribute conditions and the number of attribute conditions per policy are selected based on the case studies [16], [17]. According to case studies, number of attribute conditions varies from 50 for a web based conference management system to 1300 for a major European bank. These real time systems have about 20 attribute condition counts between 100-1500 and the attribute conditions per policy count between 2-20. Random Boolean expressions are generated consisting of conjunctions and disjunctions as policies. Each term in the Boolean expression represents an attribute condition.

Figure 3 shows the size of the attribute condition cover, that is, the number of attribute conditions the data owner enforces, for systems having 100, 500, 1000 and 1500 attribute conditions as the number of attribute conditions per policy is increased. As the number of attribute conditions per policy increases, the size of the attribute condition cover also increases. This is due to the fact that as the number of attribute conditions per policy increases, the number of distinct disjunctive terns in the DNF increases.




(a) 100 attributes (b) 500 attributes

(c) 1000 attributes (d) 1500 attributes

**Fig. 3**: Size of ACCs for different number of ACs



### 6.1 SLE vs. TLE

In 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 cloud merely acts as a storage repository. Such an approach has the advantage of hiding the ACPs from the cloud. Further, since the owner performs all access control related encryptions, a user colluding with the cloud is unable to access any data item that is not allowed to access. However, the SLE approach incurs high overhead. Since the owner has to perform all re-encryptions when user dynamics or policies change, the owner incurs a high overhead in communication and computation. Further, it is unable to perform optimizations such as delayed ABGKM::ReKey or re-encryption as the owner has to download, decrypt, re-encrypt and re-upload the data, which cloud considerably increase the response time if such optimizations are to be performed.

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 only the minimal set of attribute conditions and most of the key management tasks are performed by the cloud. Further, when identity attributes are added or removed, or the owner updates the cloud's ACPs, the owner does not have to re-encrypt the data as the cloud performs the necessary re-encryptions to enforce the ACPs. Therefore, the TLE approach reduces the communication and computation overhead at the owner. Additionally, the cloud has the opportunity to perform delayed encryption during certain dynamic scenarios as the cloud itself manages the OLE keys and encryptions. However, the improvements in the performance come at the cost of security and privacy. In this approach cloud learns some information about the ACPs.

### 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 AB-GKM scheme makes sure that only those users who satisfy the ACPs can derive the encryption keys. Therefore, only the authorized users are able to access the data. The TLE approach correctly enforces the ACPs through two encryptions. Each ACP is decomposed into two ACPs such that the conjunction of them is equivalent to the original ACP. The owner enforces one part of the decomposed ACPs through attribute based encryption. The cloud enforces the counterparts of the decomposed ACPs through another attribute based encryption. User can access data item only if it can decrypt both encryptions. As the AB-GKM scheme makes sure that only those users who satisfy these decomposed policies can derive the corresponding keys, a user can access a data item by decrypting twice only if it satisfies the two parts of the decomposed ACPs, that is, the original ACPs.

In both approaches, the privacy of the identity attributes of users is assured. ABGKM::SecGen algorithm issues secrets to users based on the identity tokens which hide the identity attributes. Further, at the end of the algorithm neither the owner nor the cloud knows if a user satisfies a given attribute condition. Therefore, neither the owner nor the cloud learns the identity attributes of users. The privacy does not weaken the security as the AB-GKM::SecGen algorithm makes sure that users can access the issued secrets only if their identity attributes satisfy the attributes conditions.

### Conclusion.

Current approaches to enforce ACPs on outsourced data using selective encryption require organizations to manage all keys and encryptions and upload the encrypted data to the remote storage. Such approaches incur high communication and computation cost to manage keys and encryptions whenever user credentials or organizational authorization policies/data change.

The proposed approach of two layer encryption solves this problem by delegating as much of the access control enforcement responsibilities as possible to the cloud while minimizing the information exposure risks due to colluding users and cloud. A key problem in this regard is how to decompose ACPs so that the owner has to handle a minimum number of attribute conditions while hiding the content from the cloud. Based on the decomposed ACPs, a novel approach for privacy preserving fine grained delegated access control to data in public clouds is proposed. This approach is based on privacy preserving attribute based key management scheme that protects the privacy of users while enforcing attribute based ACPs. As future work, the alternative choices for the TLE approach will be investigated and also plan to further reduce the computational cost by exploiting partial relationships among ACPs.

**References**

[1] J. Bethencourt, A. Sahai and B. Waters, "Ciphertext-policy attribute-based encryption", the IEEE Symposium on Security and Privacy. Washington, DC,USA: IEEE Computer Society, pp. 321-334(2007)

[2] J. Camenisch, M. Dubovitskaya and G. Neven, "Oblivious transfer with access control", ACM conference on Computer and communications security, NY, USA:ACM, pp. 131-140(2009)

[3] C-K Chu, J. Weng, S. Chow, J.Zhou and R.Deng, " Conditional proxy based re-encryption", in the proceedings of 14thAustralasian Conference on Information Security and Privacy, pp. 327-342 (2009)

[4] J-M. Do, Y-J,Song and N.Park, "Attribute based proxy re-encryption for data confidentiality in cloud computing environments", International Conference on Computers, Networks, Systems and Industrial Engineering. Los Alamitos, CA, USA: IEEE Computer Society, pp. 248-251(2011)

[5] Li and N. Li, "OACerts: Oblivious attribute certificates", IEEE Transaction on Dependable and Secure Computing, vol. 3, no. 4, pp. 340-352(2006)

[6] M. Nabeel and E. Bertino, "Attribute based group key management", IEEE Transaction on Dependable and Secure Computing(2012)

[7] M. Nabeel and E. Bertino, "Privacy preserving delegated access control in the storage as a service model", EEE International Conference on Information Reuse and Integration (IRI), (2012)

[8] M. Nabeel, E. Bertino, M. Kantarcioglu and B.M. Thuraisingham, "Towards privacy preserving access control in the cloud", International Conference on Collaborative Computing: Networking, Applications and Work sharing, ser. Collaborate Com' 11, pp. 172-180 (2011)

[9] M. Nabeel, N. Shang and E. Bertino, "Privacy preserving policy based content sharing in public clouds", IEEE Transaction on Knowledge and Data Engineering (2012)

[10] K.PN. Puttaswamy, C. Kruegel and B.Y. Zhao, "Silverline: Toward data confidentiality in storage-intensive cloud applications", ACM Symposium on Cloud Computing, ser. Socc ' 11, NY, USA: ACM, pp. 10:1-10:13, (2011)

[11] N. Shang, M. Nabeel, F. Paci and E. Bertino, "A privacy preserving approach to policy-based content dissemination", IEE International Conference on Data Engineering (2010)

[12] S.D.C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P.Samarati, "Over-encryption: Management of access control evolution on outsourced data", International Conference on Very Large Data Bases, ser. VLDB' 07. VLDB Endowment, pp. 123-134 (2007)

[13] V. Goyal, O. Pandey, A. Sahai and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data", ACM Conference on Computer and communications security, NY, USA: ACM, pp. 89-98 (2006)

[14] X. Liang, Z. Cao, H. Lin, and J.Shao, "Attribute based proxy re-encryption with delegating capabilities", International Symposium on Information, Computer and Communications Security, ser. ASIACCS' 09. NY, USA: ACM, 2009, pp. 276-286.

[15] A. Fiat and M. Naor, "Broadcast encryption", Annual International Cryptology Conference on Advances in Cryptology, ser. CRYPTO' 93. London, UK: Springer-ssVerlag, pp. 480-491 (1994)