Volume 4, Issue 8, August 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

# Implementation of Privacy Preserving delegated access in Public Cloud's using TLE Approach

**M. Venu Gopal**
M. TECH (CS)
Department of CSE,
Nist, Ibrahimpatnam,
Vijayawada,
Andhra Pradesh, India

**G. Jameson** M.TECH
Assistant Professor,
Department of CSE,
Nist, Ibrahimpatnam,
Vijayawada,
Andhra Pradesh, India

**Sd. Yasin** M.TECH (Ph.D)
Associate Professor & H.O.D
Department of CSE
Nist, Ibrahimpatnam,
Vijayawada,
Andhra Pradesh, India

***Abstract—Current approaches to enforce fine-grained access control on confidential data hosted in the cloud are based on fine-grained encryption of the data. Under such approaches, data owners are in charge of encrypting the data before uploading them on the cloud and re-encrypting the data whenever user credentials or authorization policies change. Data owners thus incur high communication and computation costs. A better approach should delegate the enforcement of fine-grained access control to the cloud, so to minimize the overhead at the data owners, while assuring data confidentiality from the cloud. We propose an approach, based on two layers of encryption, that addresses such requirement. Under our approach, the data owner performs a coarse-grained encryption, whereas the cloud performs a fine-grained encryption on top of the owner encrypted data. A challenging issue is how to decompose access control policies (ACPs) such that the two layer encryption can be performed. We show that this problem is NP-complete and propose novel optimization algorithms. We utilize an efficient group key management scheme that supports expressive ACPs. Our system assures 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 .***

## I. INTRODUCTION

Security and privacy represent major concerns in the adoption of cloud technologies for data storage. An approach to mitigate these concerns is the use of encryption. However, whereas 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-rained organizational access control policies (ACPs). Many organizations have today ACPs regulating which users can access which data; these ACPs are often expressed in terms of the properties of the users, referred to as *identity attributes*, using access control languages such as XACML. Such an approach, referred to as *attribute- based access control* (ABAC), supports fine-grained access control which is crucial for high-assurance data security and privacy. 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 thus
be strongly protected from the cloud, very much as the data themselves. Approaches based on encryption have been pro- posed for fine-grained access control over encrypted data [2], [3].

As shown in Figure 1, those approaches group data items based on ACPs and encrypt each group with a different symmetric key. Users then are given only the keys for the data items they are allowed to access. Extensions to reduce the number of keys that need to be distributed to the users have been proposed exploiting hierarchical and other relation- ships among data items. Such approaches however have several limitations:

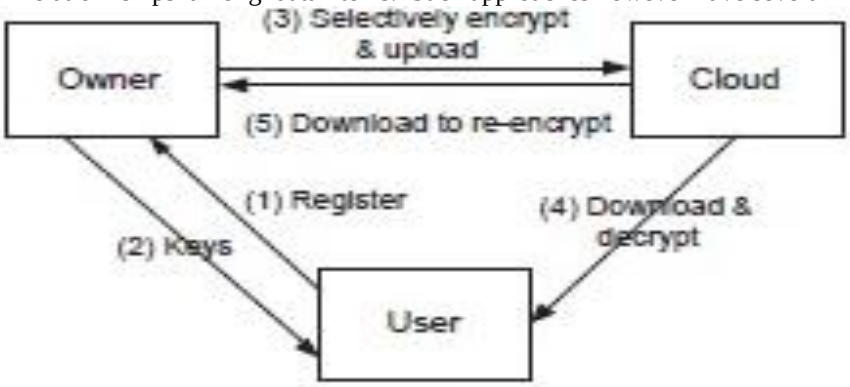


Fig. 1: Traditional Approach

 

As the data owner does not keep a copy of the data, whenever the user dynamics or ACPs change, the data owner needs to download and decrypt the data, re-encrypt it with the new keys, and upload the encrypted data. Notice also that this process must be applied to all the data items encrypted with the same key. This is inefficient when the data set to be re-encrypted is large. In order to issue the new keys to the users, the data owner needs to establish private communi- cation channels with the users. The privacy of the identity attributes of the users is not taken into account. Therefore the cloud can learn sensitive information about the users and their organization.

Recently proposed approaches based on broadcast key management schemes [4], [5], [6] address some of the above imitations. We refer to these approaches as *single layer encryption* (SLE) approaches, since, like previous approaches, they require the data owner to enforce access control through encryption performed at the data owner. However, unlike previous ap- proaches, SLE assures the privacy of the users and supports fine-grained ACPs.

However, while SLE addresses some limitations of previous approaches, it still requires the dataowner to enforce *all* the ACPs by fine-grained encryp- tion, both initially and subsequently after users areadded/revoked or the ACPs change. All these encryp- tion activities have to be performed at the owner that/thus incurs high communication and computation cost. For example, if an ACP changes, the owner must download from the cloud the data covered by this ACP, generate a new encryption key, re-encrypt the downloaded data with the new key, and then upload /the re-encrypted data to the cloud. In this paper, we propose a new approach to address this shortcoming.

The approach is based on two layers of encryption ap-/plied to each data item uploaded to the cloud. Under this approach, referred to as *two layer encryption* (TLE), 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. It should be noted that the idea of two layer encryption is not new. However, the way we perform coarse and fine grained encryption is novel and provides a better solution than existing solutions based on two layers of encryption [7]. We elaborate in details on the differences between our approach and existing solutions in the related work section.

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 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 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.

For example, if the ACP is $(C_1 \wedge C_2) \vee (C_1 \wedge C3)$, the ACP can be decomposed as two sub ACPs $C_1$ and $C_2 \vee C_3$. Notice that the decomposition is consistent; that is, $(C_1 \wedge C_2) \vee (C_1 \wedge C3) = C1 \wedge (C_2 \vee C3)$. The data owner enforces the former by encrypting the data for the users satisfying the former and the cloud enforces the latter by re-encrypting the data owner encrypted data for the users satisfying the latter. Since the cloud does not handle $C_1$, it cannot decrypt owner encrypted data and thus confidentiality is preserved. Notice that users should satisfy the original ACP to access the data by performing two decryptions. In this paper, we show that the problem of decomposing ACPs such that the data owner manages the minimum number of attribute conditions while at the same time assuring the confidentiality of the data in the cloud is NP-complete.

We propose two optimization algorithms to find the near optimal set of attribute conditions and decompose each ACP into two sub ACPs. The TLE approach has many advantages. When the policy or user dynamics changes, only the outer layer of the 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. Further, both the data owner and the cloud service utilize a broadcast key management scheme [8] whereby the actual keys do not need to be distributed to the users. Instead, users are given one or more secrets which allow them to derive the actual symmetric keys for decrypting the data. The rest of the paper is organized as follows.

Sec- tion 2 describes in detail the underlying building blocks used to construct our system. An overview of the TLE approach is given in Section 3. Section 4 provides a detailed treatment of the policy decom- position for the purpose of two layer encryption. Section 5 gives a detailed description of the TLE approach. Section 6 reports experimental results for policy decomposition algorithms and the SLE vs. the TLE approaches. We briefly analyze the trade-offs, the security and the privacy of the overall systems in Section 7. Section 8 discusses the related work and compare them with our approach. Finally, Section 9 concludes the paper providing future directions.

 

## II. BUILDING BLOCKS

In this section we first introduce broadcast encryption schemes [9], [10] and oblivious commitment based envelope protocols [11]. We present an abstract view of the main algorithms of those protocols and then describe how we use them to build our privacy-preserving attribute based group key man-agement(PPAB-GKM) scheme[8]. We then present an overview of the SLE approach[4],[5],[6] which is used as the base model for comparison with the TLE approach proposed in this paper.

### 2.1 Broadcast Encryption

Broadcast encryption(BE)[9] was introduced to solve the problem of how to efficiently encrypt a message and broadcast it to a subset of the users in a system. The subset of users can change dynamically. In the broadcast encryption literature, these users are called *privileged* and the non-authorized users *revoked*. We denote the set of users by U, the set of revoked users R. The set of privileged users is thus U\R. We set N = |U| and r = |R|. While all users can get the encrypted message, only the privileged users can decrypt it. The most simplest broadcast encryption scheme simply consists of encrypting a message for each privileged user separately and the broadcasting all the encrypted messages. Obviously, this scheme is very inefficient as the message length is prohibitively large (O(N−r)). Better broadcast encryption schemes aim to reduce the following parameters:

- The processing time at the server to encrypt the message for the privileged users.
- The processing time at privileged users to decrypt messages.
- The broadcast message size.
- The storage size at both the server and privileged users.

There are two approaches to broadcast encryption. The first approach assumes that users are stateful meaning that the keys given to users can be updated when a new user is added or an existing user is revoked. The second approach assumes that users are stateless meaning that the keys given to users cannot be updated and can only be discarded. We consider only the latter approach since in the outsourced scenarios the keys initially given to users are difficult to update and, therefore, remain unchanged.

We use an algorithm based on *subset-cover* algorithm that supports broadcast encryption with stateless users. The algorithm builds a binary tree and assigns users to the leaf nodes and thus results in a predefined user grouping. Each such group is called a subset. A user can be a member of several subsets. The *cover*, denoted by C, is defined as the set of subsets that contains all the privileged users, that is, users in U/R. The subsets in the cover are disjoint and hence each privileged user belongs to only one subset.

*Definition 1(Broadcast Encryption):* A subset-cover based Broadcast Encryption (BE) scheme consists of the algorithms: **Setup**, **GetSecKeys**, **GetCover**, **Broadcast**, **KeyDer** and **Decrypt**. Each of these algorithms are described below:

**Setup(**ℓ**,** N**)**: The server constructs a binary tree Λ where there are at least N leaf nodes. Each node in Λ is either assigned a unique key whose length is decided by the security parameter ℓ, or can computationally derive a unique key. The user ui, i =1,2,⋯ ,N, is assigned the $i^{th}$ leaf node.

**GetSecKeys(**ui**)**: The server gives all the keys assigned to ui in Λ.

**GetCover(**U\R**)**: Given the privileged user set U\R, the server outputs the cover C.

**Broadcast(**M**,** C**)**: The server generates a session key K and encrypts the message M with K and encrypts K with each key in the cover C.

**KeyDer(**ui**,** C**)**: The user ui identifies its subset in the cover C, outputs the key that decrypts the session key.

**Decrypt(**C**,** K**)**: It decrypts the encrypted message C with the key K, to output the message M.

Having defined the algorithms, we give a high-level description of the basic subset-cover technique. In the basic scheme, N users are organized as the leaves of a balanced binary tree of height logN. A unique secret is assigned to each vertex in the tree. Each user is given log N secrets that correspond to the vertices along the path from its leaf node to the root node. In order to provide forward secrecy when a single user is revoked, the updated tree is described by log N sub trees formed after removing all the vertices along the path from the user leaf node to the root node. To rekey, the server uses the log N secrets corresponding to the roots of these sub trees. Many improved subset-cover based broadcast encryption algorithms have been proposed. In this work, we consider the *complete subtree* algorithm[10].The complete subtree algorithm improves the basic technique for simultaneously revoking r users and describing the privileged users using r log(N/r)subsets. Each user stores log N keys.

### 2.2 Oblivious Commitment Based Envelope Protocols

The Oblivious Commitment Based Envelope(OCBE) protocols, proposed by Li and Li [11], provide a mechanism to obliviously deliver a message to the users who satisfy certain conditions. There are three entities in these protocols, a server Svr, a user Usr, and a trusted third party, called the identity provider (IdP). IdP issues to Usr identity tokens, expressed as Pedersen commitments[12], corresponding tothe identity attributes of Usr.

*Definition 2(OCBE):* An OCBE protocol consists of the algorithms: **Setup**, **GenCom** and **GetData**. Each of these algorithms are described below:

**Setup(**ℓ**)**: The IdP runs a Pedersen commitment setup protocol to generate the system parameters, a finite cyclic group G of large prime order p, two generators g and h of G. The size of p is dependent on the security parameter ℓ.

 

**GenCom(**x**)**:A Usr wants to commit to the value x. It submits x to the IdP. The IdP computes the Pedersen $x_h r$ commitment c = g where r is randomly chosen from Fp. The IdP digitally signs c and sends r, c and the signature of c to the Usr.

**GetData(**c**, cond,** d**)**: The Usr sends the signed commitment c and indicates the Svr's condition cond that it wants to satisfy. cond has the format "name predicate value" where the predicate can be >,≥,<,≤ or =. For example, cond "age ≥ 18" and c are for the attribute age. After an interactive session, the Svr encrypts the data d and sends the encrypted data, called envelope, to the Usr. The Usr can decrypt and access the data only if it satisfies the condition.

As mentioned above, the OCBE protocols always guarantee the following properties:

- The Svr does not learn the identity attributes of the users.
- AUsr can open the envelope onlyifits committed attribute value satisfies the condition.
- A Usr cannot submit fake commitments in order to satisfy a condition as the commitments are signed by the IdP.

### 2.3 Privacy Preserving Attribute Based Group Key Management

The privacy preserving attribute based group key management (PP AB-GKM) scheme uses the BE schemeintroducedinSection2.1 and the OCBE protocols introduced in Section 2.2. Such scheme combines the previous work on AB-GKM scheme[8],[13] and privacy preservation in Broadcast Group Key Management(BGKM)[4],[6].

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 to change the public information and does not affect the secret shares given to existing users. In general ,a BGKM scheme consists of the following five algorithms: **Setup**, **SecGen**, **Key-Gen**, **KeyDer**, and **ReKey**. Our overall construction is based on the AB-GKM scheme [8], [13] which is an expressive construct of the ACV-BGKM (Access Control Vector BGKM) scheme[4],[6].

Before we present details of the PP AB-GKM protocol, we first define certain terms that are useful to describe the protocol. Attribute conditions and access control policies are formally defined as follows.

*Definition 3(Attribute Condition):* An attribute condition C is an expression of the form: "nameattr opl", where nameattr is the name of an identity attribute attr, op is a comparison operator such as =, <, >, ≤, ≥, 6=, and l is a value that can be assumed by the attribute attr.

*Definition4(Access controlpolicy):* An access control policy ACP is a tuple (s,o) where: o denotes a set of data items {D1,..., Dt}; and s is a monotonic expression [1] over a set of attribute conditions that must be satisfied by a Usr to have access to o.

We denote the set of all attribute conditions as ACB and the set of all ACPs as ACPB. Example 1 shows an example ACP.

*Example 1:* The ACP

((“yos ≥ 5 ∧ “role = nurse ”)∨ “role = doctor ”, {physical exam,treatment plan})

states that a Usr, either playing the role of doctor or playing the role nurse with yos, years of service, no less than 5, has access to the data items "physical exam" and "treatment plan".

Before providing a detailed description of the PP AB-GKM scheme, we present the intuition and abstract algorithms. A separate BGKM instance for each attribute condition is constructed. The ACP is embedded in an access structure T . T is a tree in which the internal nodes represent threshold gates and the leaves represent BGKM instances for the attributes. T can represent any monotonic policy. The goal of the accesstreeisto allowthederivation of thegroup key for only the users whose attributes satisfy the access structure T . Figure 2 shows the access tree for the ACP in Example 1.

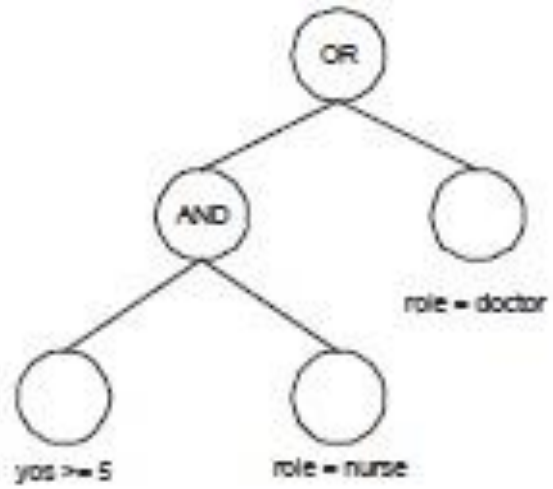


Fig. 2: An Example Access Tree

 

A high-level description of the access tree is as follows.Each threshold gate in the tree is described by its child nodes and a threshold value. The threshold value tx of a node x specifies the number of child nodes that should be satisfied in order to satisfy the node. Each threshold gate is modeled as a Shamir secret sharing polynomial [14] whose degree equals to one less than the threshold value. The root of the tree contains the group key and all the intermediate values are derived in a top-down fashion.A user who 1. Monotonic expressions are Boolean formulas that contain only conjunction and disjunction connectives, but no negation. satisfies the access tree derives the group key in a bottom-up fashion.

Due to space constraints, we only provide the abstract algorithms of the PP AB-GKM scheme. As a convention, we use the format <protocol name>::<algorithm> to refer to the constructs introduced earlier or elsewhere. Specifically, BE::<algorithm>, OCBE::<algorithm> and ACVBGKM::<algorithm> [4]to refer to algorithms of BE, OCBE and ACV-BGKM protocols respectively.

*Definition 5(PP AB-GKM):* The PP AB-GKM scheme consists of five algorithms: **Setup**, **SecGen**, **KeyGen**, **KeyDer** and **ReKey**.An abstractdescription of these algorithms are given below.

**Setup(**ℓ**,** N**,** Na**)**: It taking the security parameter ℓ, the maximum group size N, and the number of attribute conditions Na as input, initializes the system. It invokes BE:Setup(ℓ, N), OCBE::Setup(ℓ) and ACV-BGKM::Setup(ℓ, N)algorithms.

**SecGen(**γ**)**: The secret generation algorithm gives a Usrj , $1 \leq j \leq N$ a set of secrets for each commitment comi $\in$ γ, $1 \leq i \leq$ m. It invokes BE::GetSecGen and OCBE::GetData algorithms.

**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.

**KeyDer(**β**, PI,** T **)**: Given the set of identity attributes β, 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 β satisfy the access structure T . It invokes BE:KeyDer and ACV-BGKM::KeyDer algorithms.

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

### 2.4 Single Layer Encryption Approach

The SLE scheme [6] consists of the four entities, Owner, Usr, IdP and Cloud. They play the following roles:

- Owner, the data owner defines ACPs, and uploads encrypted data to the Cloud, the cloud storage service.
- The Cloud hosts the encrypted data of the Owner.
- IdP, the identity provider, a trusted third party, issues *identity tokens* to users based on the attribute attributes users have. An identity token is a signed Pedersen commitment that binds the identity attribute value to a Usr while hides it from others. There can be one or more certified IdPs. We assume that all 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

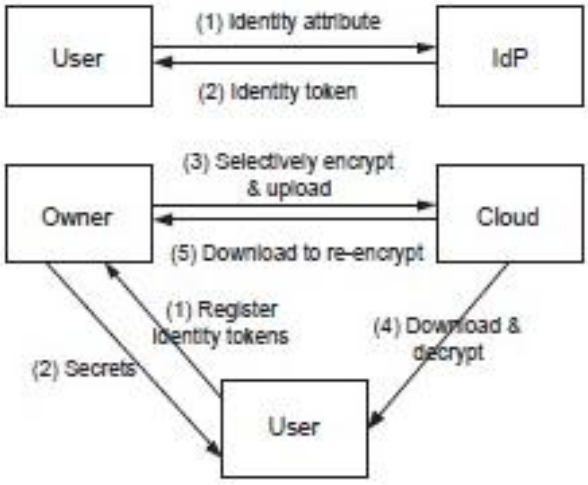


Fig. 3: Single layer Encyption Approach

As shown in Figure 3, the SLE approach follows the 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. We give an overview of the five phases below:

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

**Identity token registration:** Usrs 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::KeyGen algorithm and uploads to the Cloud.

**Data downloading and decryption:** Usrs download encrypted data from the Cloud and decrypt using the key derived from the AB-GKM::KeyDer algorithm.

 

**Encryption evolution management:** Over time, either access control polices 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.

## III. OVERVIEW

We now give an overview of our solution to the problem of delegated access control to outsourced data in the cloud. A detailed description is provided in Section 5. Like the SLE system described in Section 2.4, the TLE system consists of the four entities, Owner, Usr, IdP and Cloud. However, unlike the SLE approach, 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 delegates 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. Figure 4 shows the system diagram of the TLE approach. The system goes through one additional phase compared to the SLE approach. We give an overview of the six phases below:

**Identity token issuance:** IdPs issue identity tokens to Usrs based on their identity attributes.
**Policy decomposition:** The Owner decomposes each ACP into 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 enforce the original ACPs. The Owner enforces the confidentiality related sub ACPs and the Cloud enforces the remaining sub ACPs.
**Identity token registration:** Usrs register their identity tokens in order to obtain secrets to decrypt the data that they are allowed to access. Usrs 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. It should be noted that the Cloud does not learn the identity attributes of Usrs during this phase.
**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 public information generated by the AB-GKM::KeyGen algorithm and the remaining sub ACPs to the Cloud. The Cloud in turn encrypts the data based on the keys generated using its own AB-GKM::KeyGen algorithm. Note that the AB-GKM::KeyGen at the Cloud takes the secrets issued to Usrs and the sub ACPs given by the Owner into consideration to generate keys.
**Data downloading and decryption:** Usrs download encrypted data from the Cloud and decrypt the datausing the derived keys. Usrs decrypt twice to first remove the encryption layer added by the Cloud and then by the Owner. As access control is enforced through encryption, Usrs can decrypt only those data for which they have valid secrets.
**Encryption evolution management:** Over time, either ACPs or user credentials may change. Further, already encrypted data may go through frequent updates. In such situations, data already encrypted must be re- encrypted with a new key. As the Cloud performs the access control enforcing encryption, it simply re- encrypts the affected data without the intervention of the Owner.

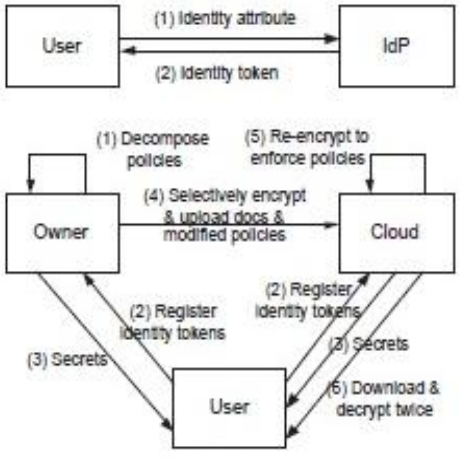


Fig. 4: Two Layer Encryption approach

## IV. POLICY DECOMPOSITION

Recall that in the SLE approach, the Owner incurs a high communication and computation overhead since it has to manage all the authorizations when user dynamics or ACPs change. 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 to enforce

 

authorization policies through encryption and avoid re-encryption by the Owner, the data may have to be encrypted again to have two encryption layers. We call the two encryption layers as *inner encryption layer* (IEL) and *outer encryption later* (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 data items using a single symmetric key and let the Cloud per- form 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 Usrs and the Cloud. Further, IEL updates 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 ttributes. An alternative solution is based on de composing 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. In what follow we propose such an approach to decompose and we also show that the policy decomposition problem is hard.

**4.1 Policy Cover**

We define the *policy cover problem* as the the optimization problem of finding 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 a set of attribute conditions as the *attribute condition cover*. For example, if ACPB consists of the three simple ACPs {C1 ∧ C2,C2 ∧ C3,C4}, the minimum set of attributes that covers ACPB is {C2,C4}. C2 should be satisfied in order to satisfy the ACPs C1 ∧ C2 and C2 ∧ C3. Notice that satisfying C2 is not sufficient to satisfy the ACPs. The set is minimum since the set obtained by removing either C2 or C4 does not satisfy the cover relationship. We define the related decision problem as follows.

*Definition 6(POLICY-COVER):* Determine whether ACPB has a cover of k attribute conditions.

The following theorem states that this problem is NP-complete.

*Theorem 1:* The POLICY-COVER problem is NP-complete.

*Proof:* We first show that POLICY-COVER ∈ NP. Suppose that we aregiven a set of ACPs ACPB which contains the attribute condition set AC, and integer k. For simplicity, we assume that each ACP is a conjunction of attribute conditions. However, the proof can be trivially extended to ACPs having any monotonic Boolean expression over attribute conditions. The certificate we choose has a cover of attribute conditions
AC ⊂AC. The verification algorithm affirms that |AC | = k, and then it checks, for each policy in the ′
ACPB, that atleast one attribute conditionin AC is in the policy. This verification can be performed trivially in polynomial time. Hence, POLICY-DECOM is NP.

Now we prove that the POLICY-COVER problem is NP-hard by showing that the vertex cover problem, which is NP-Complete, is polynomial time reducible to the POLICY-COVER problem. Given an undirected graph G =(V,E)and an integer k, we construct a set of ACPs ACPB that has a cover set of size k if and only if G has a vertex cover of size k. ′

Suppose G has a vertex cover V⊂ V with |V | = k. We construct a set of ACPs ACPB that has a cover of k attribute conditions as follows. For each vertex vi ∈ V, we assign an attribute condition Ci. For each vertex vj ∈ V , we construct an access control policy by obtaining the conjunction of attribute conditions as follows.

• Start with the attribute condition Cj as the ACP Pj
• For each edge (vj ,vr), add Cr to the ACP as a conjunctive literal(For example, if the edges are (vj ,va),(vj , vb)and (vj ,vc), weget Pj = Cj ∧ Ca∧ Cb ∧ Cc)

At the end of the construction we have a set of distinct access control policies ACPB with size k. We construct the attribute condition set AC = {C1,C2,⋯ ,Ck} such that Ci corresponds to each vertex in V . In order to satisfy all access control policies, the attribute conditions in AC must be satisfied. Hence, AC is an attribute condition cover of size k for the ACPs ACPB.

Conversely, suppose that ACPB has an attribute condition cover of size k. We construct G such that each attribute condition corresponds to a vertex in G and an edge between vi and vj if they appear in the same access control policy. Let this vertex set be V1. Then we add the remaining vertices to G corresponding to other attribute conditions in the access control policies and add the edges similarly. Since the access control policies are distinct there will be at least one edge (vi,u) for each vertex vi in attribute condition cover such that u 6∈ V1. Hence G has a vertex cover of size V1 = k.

Since the POLICY-COVER problem is NP-complete, one cannot find a polynomial time algorithm for finding the minimum attribute condition cover. In the following section we present two approximation algorithms for the problem.

The APPROX-POLICY-COVER1 algorithm 2 takes as input the set of ACPs ACPB and returns a set of attribute conditions whose size is guaranteed to be no more than twice the size of an optimal attribute condition cover. APPROX-POLICY-COVER1 utilizes the GEN-GRAPH algorithm1 to first represent ACPB as a graph.

 

**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 6∈ 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 8

We give a high-level overview of the GEN-GRAPH algorithm 1. It takes the ACPB as the input and converts each ACP nto 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.

**Algorithm 2** APPROX-POLICY-COVER1
1: G = GEN-GRAPH(ACPB)
2: ACC = φ
3: **for** Each disconnected subgraph Gi =(Vi,Ei) of G **do**
4: **if** |Vi| == 1 **then**
5: Add ACi corresponding to the vertex to ACC
6: **else**
7: **while** Ei 6= 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 v
11: **end while**
12: **end if**
13: **end for**
14: Return ACC

As shown in the APPROX-POLICY-COVER1 algo-rithm 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 the GEN-GRAPH algorithm 1. Then for each disconnected subgraph Gi of G, it finds the near optimal attribute condition cover and add to the ACC. The attribute condition to be added is re-lated at random by selecting a random edge in Gi.
Once an edge is considered, all its incident edges are removed from Gi. The algorithm continues until all edges are removed from each Gi. The running time of the algorithm is O(V + E) using adjacency lists to represent G. It can be shown that the APPROXPOLICY-COVER1 algorithm is a polynomial-time 2approximation algorithm.

We now present the idea behind our second approximation algorithm, APPROX-POLICY-COVER2, which uses a heuristic to select the attribute conditions. This algorithm is similar to the APPROXPOLICY-COVER1 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.

*Example 2:* 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 Usrs playing different roles such as receptionist (rec), cashier (cas), doctor (doc), nurse (nur), pharmacist (pha), and system administrator (sys). An EHR document consists of data items including BillingInfo (BI), ContactInfo (CI), MedicationReport (MR), PhysicalExam(PE), LabReports(LR), Treatment Plan (TP) and so on. In accordance with regulations such as health insurance portability and accountability act (HIPAA), the hospital policies specify which users canaccesswhichdataitem(s). Inourexamplesystem, there are four attributes, role (rec, cas, doc, nur, pha, sys), insurance plan, denoted as ip,(ACME, MedA, MedB, MedC),type (assistant, junior, senior)andyear of service, denoted as yos,(integer). The following is the re-arranged set of ACPs of the hospital such that each data item has a unique ACP. (“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)

Figure 5 shows the graph generated by the GENGRAPH algorithm for our running example. Notice that there are 5 disconnected graphs. Assume that APPROX-POLICY-COVER2 algorithm is used to construct the AC cover. As mentioned in the approximation algorithm, single vertex graphs are trivially included in the AC cover. The remaining attribute conditions are selected using the greedy heuristic. Thatgives ustheAC cover ACC = { "role = rec", "role = cas", "role = pha", "role = doc", "role = nur", "role = dat"}.

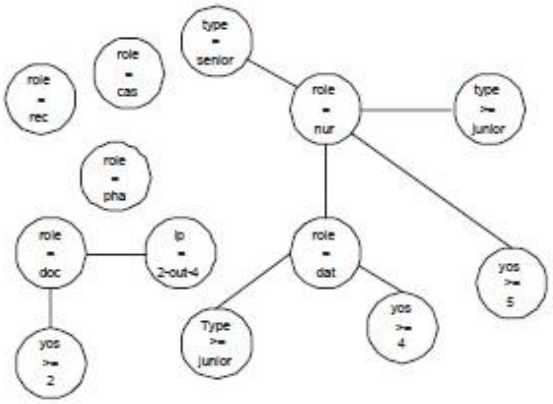


Fig. 5: The example graph

**4.2 Policy Decomposition**

The Owner manages only those attribute conditionsin ACC.The Cloud handles the remaining set of attribute 9 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 POLICYDECOMPOSITION algorithm 3 shows how the ACPs are decomposed into two sub ACPs based on the attribute conditions in ACC.

**Algorithm 3** POLICY-DECOMPOSITION

1: ACPBOwner = 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 6∈ 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: DecomposethemultiACterm c intoc1 and c2 such that ACs in c1 ∈ ACC, ACs in c2 6∈ 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 6∈ 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 ACPBOwner
27: Add ACPi(cloud)to ACPBCloud
28: **end for**
29: Return ACPBOwner and ACPBCloud

 

Algorithm 3 takes the ACPB and ACC as input and produces the two sets of ACPs ACPBOwner and ACPBCloud that are to be enforced at the Owner and the Cloud respectively. It first convertseachpolicyinto DNF and decompose each conjunctive term into two conjunctive terms such that one conjunctive term has only thoseACsin ACC and the other term may or may not have the ACs in ACC. It can be easily shown that the policy decomposition is consistent. That is, the conjunction of corresponding sub ACPsin ACPBOwner and ACPBCloud respectivelyproduces an original ACP in ACPB.

*Example 3:* For our example ACPs, the Owner handles the following sub ACPs. (“role = rec” ∨ “role = nur” , CI) (“role = cas” ∨ “role = pha”, BI) (“role = doc”, CR) (“role = doc” ∨ “role = pha”, TR) (“role = doc” ∨ “role = nur” ∨ “role = pha”, MR) (“role = nur” ∨ “role = dat” ∨ “role = doc”, LR) (“role = nur” ∨ “role = dat”, PE)

As shown in Algorithm 3, the Owner re-writes the ACPs that the Cloud should enforce such that the 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” ∨ “type ≥ junior”, CI) (“role = cas” ∨ “role = pha”, BI) (“ip = 2-out-4”, CR) (“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)

## V. TWO LAYER ENCRYPTION APPROACH

In this section, we provide a detailed description of the six phases of the TLE approach introduced in Section 3. The system consists of the four entities, Owner, Usr, IdP and Cloud. Let the maximum number of users in the system be N, the current number of users be n (<N), and the number of attribute conditions Na.

### 5.1 Identity token issuance

IdPs are trusted third parties that issue identity tokens to Usrs 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 Usr 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 IdP’s digital signature on nym, id-tag and c.

### 5.2 Policy decomposition

Using the policy decomposition algorithm 3, 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 the confidentiality related sub ACPs in ACPB Owner and the Cloud enforces the remaining sub ACPs in ACPB Cloud.

### 5.3 Identity token registration

Usrs register their IT s to obtain secrets in order to later decrypt the data they are allowed to access. Usrs register their IT s related to the attribute conditions in ACC with the Owner, and the rest of the identity tokens related to the attribute conditions in ACB/ACC with the Cloud using the AB-GKM::Sec Gen algorithm.

When Usrs register with the Owner, the Owner issues them two sets of secrets for the attribute conditions in ACC that are also present in the sub ACPs in ACPB Cloud. The Owner keeps one set and gives the other set to the Cloud. Two different sets are used in order top event the Cloud from decrypting the Owner encrypted data.

### 5.4 Data encryption and upload

The Owner encrypts the data based on the sub ACPs in ACPB Owner and uploads them along with the corresponding public information tuples to the Cloud. The Cloud in turn encrypts the data again based on the sub ACPs in ACPB Cloud. Both parties execute ABGKM::Key Gen algorithm individually to first generate the symmetric key, the public information tuple PI and access tree T for each sub ACP. We now give a detailed description of the encryption process.

The Owner arranges the sub ACPs such that each data item has a unique ACP. Note that the same policy maybe applicable to multiple data items. Assume that the set of data items D = {d1,d2,⋯ ,dm} and the set of sub ACPs ACPB Owner = {ACP1,ACP2,⋯ , ACPn}. The Owner assigns a unique symmetric key, called an ILE key, $K^{ILE}$ for each sub ACPi ∈ ACPB Owner, i encrypts all related data with that key and executes the AB-GKM::Key Gen to generate the public PIi and Ti. The Owner uploads those encrypted data (id, $E_K$ILE (di), i) along with the indexed public in for I mation tuples (i, PIi, Ti), 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 ACPj ∈ ACPB Cloud, the Cloud assigns a unique symmetric key $K_j^{OLE}$, called an OLE key, encrypts each affected data item $E_K$ILE (di) and produces the tuple (id, I $E_K$OLE ($E_K$ILE (di)), i, j), where i and j gives the index ji of the public information generated by the Owner and the Cloud respectively.

### 5.5 Data downloading and decryption

Usrs download encrypted data from the Cloud and decrypt twice to access the data. First, the Cloud generated public information tuple is used to derive the OLE key and then the Owner generated public information tuple is used to derive the ILE key using the AB-GKM::Key Der algorithm. These two keys allow a Usr to decrypt a data item only if the Usr satisfies the original ACP applied to the data item.

 

For example, in order to access a data item di, Usrs download the encrypted data item $E_K$OLE ($E_K$ILE (di)) ji and the corresponding two public information tuples PIi and PIj . PIj is used to derive the key of the outer layer encryption $K^{OLE}$ and PIi used to derive the j key of the inner layer encryption $K^{ILE}$. Once those i two keys are derived, two decryption operations are performed to access 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 Usrs is a strict superset of the old group of Usrs, and backward secrecy is enforced.

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

## VI. EXPERIMENTAL RESULTS

In this section we first present experimental results concerning the policy decomposition algorithms. We then present an experimental comparison between the SLE and TLE approaches. The experiments were performed on a machine running GNU/Linux kernel version 2.6.32 with an Intel R Core 2 Duo CPU T9300 2.50GHz and 4 Gbytes memory. Only one processor was used for computation. Our prototype system is implemented in C/C++. We use V. Shoup's NTL library [15] version 5.4.2 for finite field arithmetic, and SHA-1 and AES- 256 implementations of OpenSSL [16] version 1.0.0d for cryptographic hashing and incremental encryption. We use boolstuff library [17] version 0.1.13 to convert policies into DNF. Adjacency list representation is used to construct policy graphs used in the two approximation algorithms for finding a near optimal attribute condition cover. We utilized the AB-GKM scheme with the subset cover optimization. We used the complete subset algorithm introduced by Naor et. al. [10] as the subset cover. We assumed that 5% of attribute credentials are revoked for the AB-GKM related experiments. All finite field arithmetic operations in our scheme are performed in an 512-bit prime field. For our experiments, we selected the total number of attribute conditions and the number of attribute conditions per policy based on past case studies [18], [19].

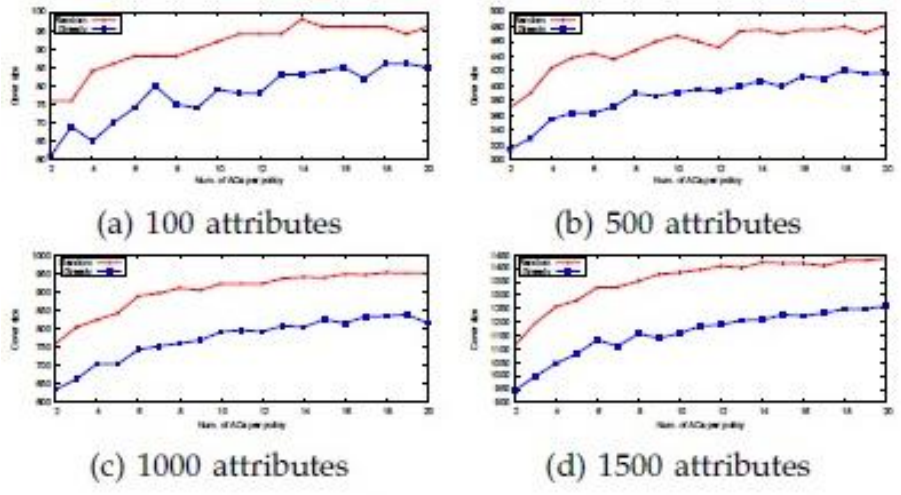


Fig. 6: Size of ACCs for different number of ACs

According to the case studies, the number of attribute conditions varies from 50 for a web based onference management system to 1300 for a major European bank. These real systems have upto about 20 attribute conditions per policy. We set the total attribute condition count between 100-1500 and the the attribute conditions per policy count between 2-20. We generate random Boolean expressions consisting of conjunctions and disjunctions as policies. Each term in the Boolean expression represents a attribute condition.

Figure 6 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. In all experiments, the greedy policy cover algorithm performs better. 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 terms in the DNF increases. Figure 7 shows the break down of the running time for the complete policy decomposition process. In this experiment, the number of attribute condition is set to 100, 500, 1000 and the maximum number of attribute conditions per policy is set to 5. The total execution time is divided into the execution times of three different components of our scheme. The"DNF Graph" time refers to the time required to convert the policies to DNF and construct a in-memory graph of policies using an adjacency

 

list. The "Cover" time refers to the time required to to find the optimal cover and the "Decompose" time refers to time required to to create the updated policies for the data owner and the cloud based on the cover. As can be seen from the graphs, most of the time is spent on finding a near optimal attribute condition cover. It should be noted that the random approximation algorithm runs faster than the greedy algorithm. One reason for this behavior is that each time the latter algorithm selects a vertex it iterates through all the unvisited vertices in the policy graph, whereas he former algorithm simply picks a pair of unvisited vertices at random. Consistent with the worst-cast running times, the"DNF + Graph" and "Decompose" components demonstrate near linear running time, and 'the 'Cover" component shows non-linear running time.

Figure 8 reports the average time spent to execute the AB-GKM::KeyGen with SLE and TLE approaches for different group sizes. We set the number of attribute conditions to 1000 and the maximum number of attribute conditions per policy to 5. We utilize the greedy algorithm to find the attribute condition cover. As seen in the diagram, the running time at the Owner in the SLE approach is higher since the Owner has to enforce all the attribute conditions. Since the TLE approach divides the enforcement cost between the Owner and the Cloud, the running time at the Owner is lower compared to the SLE approach. The running time at the Cloud in the TLE approach is higher than that at the Owner since the Cloud performs fine grained encryption whereas the Owner only performs coarse grained encryption. As shown in Figure 9, a similar pattern is observed in the AB-GKM::KeyDer as well.

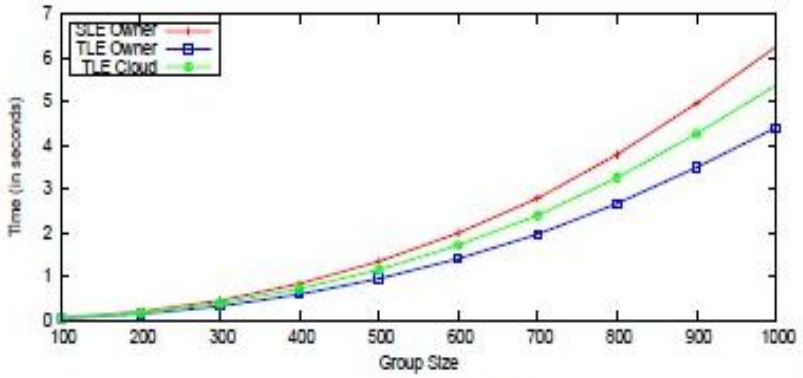


Fig. 8: Average time to generate keys for the two approaches

## VII. ANALYSIS

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

### 7.1 SLE vs. TLE

Recall that 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 Usr 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 AB- GKM::ReKey or re-encryption as the Owner has to download, decrypt, re-encrypt and re-upload the data, which could 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 reencrypt 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 OEL keys and encryptions. However, the improvements in the performance comes at the cost of security and privacy. In this approach, the Cloud learns some information about the ACPs.

### 7.2. 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 Usrs who satisfy the ACPs can derive the encryption keys. Therefore, only the authorized Usrs 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. Usr can access a data item only if it can decrypt both encryptions. As the AB-GKM scheme makes sure that only those Usrs who satisfy these decomposed policies can derive the corresponding keys, a Usr 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 Usrs is assured. Recall that the AB- GKM::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 Usr satisfies a given attribute condition. Therefore, neither the Owner nor the Cloud learns the identity attributes of Usrs. Note that the privacy does not weaken the security as the AB-GKM::SecGen algorithm makes sure that Usrs can access the issued secrets only if their identity attributes satisfy the attribute conditions.

## VIII. RELATED WORK

Fine-grained Access Control: Fine-grained access control (FGAC) allows one to enforce selective access to the content based on expressive policy specifi- cations. 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 [2], [3] subdocuments are encrypted with different keys, which are provided to users at the registration phase. The encrypted subdocuments are then broadcasted to all users.

However, such approaches require that all [2] or some [3] keys be distributed in advance during user registration phase. This requirement makes it difficult to assure forward and backward key secrecy when user groups are dynamic or the ACPs change. Further, the rekey process is not transparent, thus shifting the burden of acquiring new keys on users. Shang et al. [4] proposes approach to solve such problem.

It lays the foundation to make rekey transparent to users and protect the privacy of the users who access the content. However, it does not support expressive access control policies as in our approach and also it is not directly applicable to pull based approaches.

Under the pull-based model, the content publisher is required to be online in order to provide access to the content. Recent research efforts [20], [21], [5], [22] have proposed approaches to construct privacy preserving access control systems using a third-party storage service. In such approaches, the data owner

A major drawback of all the above approaches is that they do not consider the management of encrypted data hosted in a third party when users are added or removed from the system or when the ACPs/subdocuments are updated. All the approaches require the data owner to handle encryption.

Di Vimercati et al. [7] first identifies this problem and proposes an initial solution. While their solution improves over existing solutions, such solution does not support expressive attribute based policies and does not protect the privacy of the users.

**Attribute Based Encryption**: The concept of attribute- based encryption (ABE) has been introduced by Sahai and Waters [23]. The initial ABE system is limited only to threshold policies in which there are at least out of n attributes common between the attribute used to encrypt the plaintext and the attributes users possess. Pirretti et al. [24] gave an implementation of such a threshold ABE system using a variant of the Sahai-Waters Large Universe construction [23]. Since this initial threshold scheme, a few variants have been introduced to provide more expressive ABE systems. Goyal et al. [25] introduced the idea of key- policy ABE (KP-ABE) systems and Bethencourt et al. [26] introduced the idea of ciphertext-policy ABE (CP-ABE) systems. Even though these constructs are expressive and provably secure, they are not suitable for group management and especially in supporting forward security when a user leaves the group (i.e. attribute revocation) and in providing backward security when a new user joins the group. Some of the above schemes suggest using an expiration attribute along with other attributes. However, such a solution is not suitable for a dynamic group where joins and departures are frequent.

**Proxy Re-Encryption**: In a proxy re-encryption (PRE) scheme [27] one party A delegates its decryption rights to another party B via third party called a "proxy." More specifically, the proxy transforms a ciphertext computed under party A's public key into a different ciphertext which can be decrypted by party B with B's private key. In such a scheme neither the proxy nor party B alone can obtain the plaintext. Recently Liang et al. [28] has extended the traditional PRE to attribute based systems and independently Chu et al. [29] has extended the traditional PRE to support conditions where a proxy can re-encrypt only if the condition specified by A is satisfied. These improved PRE techniques alone or combined with ABE schemes [30] could be utilized to implement delegated access control in the cloud. However, they do not protect the identity attributes of the users who access the system and are difficult to manage.

## IX. 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 ap- proaches incur high communication and computa- tion cost to manage keys and encryptions when- ever user

 

credentials or Organizational authorization policies/data change. In this paper, we proposed a two layer encryption based approach to solve this problem by delegating as much of the access con- trol enforcement responsibilities as possible to the Cloud while minimizing the information exposure risks due to colluding Usrs 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. We showed that the policy decomposition problem is NP-Complete and provided approxima- tion algorithms. Based on the decomposed ACPs, we proposed a novel approach to privacy preserving fine- grained delegated access control to data in public clouds. 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, we plan to investigate the alternative choices for the TLE approach further. We also plan to further reduce the computational cost by exploiting partial relationships among ACPs.

## ACKNOWLEDGMENTS

The work reported in this paper has been partially supported by the MURI awardFA9550-08-1-0265from the Air Force Office of Scientific Research.

## REFERENCES

[1] M. Nabeel and E. Bertino, "Privacy preserving delegated access control in the storage as a service model," in *EEE International 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. Secur.*, 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 the 29th international conference on Very large data bases*. VLDB Endowment, 2003, pp. 898–909.

[4] N. Shang, M. Nabeel, F. Paci, and E. Bertino, "A privacy-preserving approach to policy-based content dissemination," in *ICDE '10: Proceedings of the 2010 IEEE 26th International Conference on Data Engineering*, 2010.

[5] M. Nabeel, E. Bertino, M. Kantarcioglu, and B. M. Thuraisingham, "Towards privacy preserving access control in the cloud," in *Proceedings of the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing*, ser. CollaborateCom '11, 2011, pp. 172–180.

[6] M.Nabeel,N.Shang,andE.Bertino, "Privacypreservingpolicy based content sharing inpublic clouds," *IEEE Transactions on Knowledge and Data Engineering*, 2012. 14

[7] S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Over-encryption: Management of access control evolution on outsourced data," in *Proceedings of the 33rdInternationalConferenceonVeryLarge DataBases*, ser.VLDB '07. VLDB Endowment, 2007, pp. 123–134.

[8] M. Nabeel and E. Bertino, "Towards attribute based group key management," in *Proceedings of the 18th ACM conference on Computer and communications security*, Chicago, Illinois, USA, 2011.

[9] A. Fiat and M. Naor, "Broadcast encryption," in *Proceedings of the 13th Annual International Cryptology Conference on Advances inCryptology*,ser.CRYPTO '93. London, UK:Springer-Verlag, 1994, pp. 480–491.

[10] D. Naor, M. Naor, and J. B. Lotspiech, "Revocation and tracing schemes for stateless receivers," in *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '01. London, UK: Springer-Verlag, 2001, pp. 41–62.

[11] J. Li and N. Li, "OACerts: Oblivious attribute certificates," *IEEE Transactions on Dependable and Secure Computing*, vol. 3, no. 4, pp. 340–352, 2006.

[12] T.Pedersen,"Non-interactiveandinformation-theoreticsecure verifiable secret sharing," in *CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*. London, UK: Springer-Verlag, 1992, pp. 129–140.

[13] M.NabeelandE.Bertino, "Attributebasedgroupkey management," *IEEE Transactions on Dependable and Secure Computing*, 2012.

[14] A. Shamir, "How to share a secret," *The Communication of ACM*, vol. 22, pp. 612–613, November 1979.

[15] V. Shoup, "NTL library for doing number theory," http://www.shoup.net/ntl/.

[16] "OpenSSL the open source toolkit for SSL/TLS," http://www.openssl.org/.

[17] "boolstuff a boolean expression tree toolkit," http://sarrazip.com/dev/boolstuff.html.

[18] A. Schaad, J. Moffett, and J. Jacob,"The role-based access control system of a european bank: a case study and discussion," in *Proceedings of the sixth ACM symposium on Access control models and technologies*, ser. SACMAT '01. New York, NY, USA: ACM, 2001, pp. 3–9.

[19] K. Fisler, S. Krishnamurthi, L. A. Meyerovich, and M. C. Tschantz, "Verification and change-impact analysis of access-control policies," in *Proceedings of the 27th international conference on Software engineering*, ser. ICSE '05. New York, NY, USA: ACM, 2005, pp. 196–205.

[20] S.Coull,M.Green,andS.Hohenberger,"Controlling accessto an oblivious database using stateful anonymous credentials," in *Irvine: Proceedings of the 12th International Conference onPractice and Theory in Public Key Cryptography*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 501–520.

 

**About Authors:**

**M.Venu Gopal** Pursuing M.Tech in Nimra Institute of Science and Technology, JNTUK and Completed B.Tech in Priyadarshini College of Engineering Sullurpet,JNTUH and I Interested Research Area in Cloud Computing .

**JAMESON GANTA** working as Assistant Professor in Nimra Institute of science and Technology and completed B.Tech in BVCITS COLLEGE , JNTUH and M.Tech in V R SIDDHARTHA ENGG COLLEGE, JNTUK .

**SAYEED YASIN** working as Associate Professor &H.O.D in CSE Department, Nimra Institute of science and Technology and completed MCA in OSMANIA UNIVERSITY , M.Tech in JNTU Hyderabad and Pursuing my Ph.D from RAYALASEEMA UNIVERSITY.I am doing my Research on Wireless Networks.

 