Updating attribute in CP-ABE: A New Approach

Nishant Doshi and Devesh Jinwala

National Institute of Technology, Surat, India
{doshinikki2004,dcjinwala}@gmail.com

Abstract. In Ciphertext-Policy Attribute Based Encryption (CP-ABE), attributes are attached to the user's secret key and access policy is attached to the ciphertext. If attributes in the secret key of a user satisfy the policy then only the genuine user can decrypt the ciphertext. However, such scenario also necessitates periodic updating of the secret key with the changing attributes. According to our observations, the existing attempts at doing so are not efficient. In this paper, we propose a newer approach to add, update or delete the value of particular attribute efficiently without the knowledge of the other attributes.

Keywords: Attribute, Attribute Based Encryption, Dynamic Attributes, Network Security.

1 Introduction

The Public Key Cryptography (PKC) was proposed by Rivest et al. principally to overcome the limitation in ensuring secure group communications in the Symmetric Key Cryptography based cryptosystems [1]. However, the PKC based cryptosystems involve costly and complex public key authentication framework known as the public key infrastructure. In 1984, Shamir in [2] proposed Identity Based Encryption (IBE) to reduce the complexity associated with the pure PKC based systems. The IBE emphasizes using a user's identifier such as an e-mail address or an IP address as his public key, instead of using digital certificates, for the public key authentication.

However, in PKC as well as in IBE based cryptosystems, if one requires to multicast a message, then it has to be encrypted using different public keys and this unnecessarily increases the associated computational overhead. In [3], Sahai et al. proposed a fuzzy identity based encryption approach, which aimed overcoming this limitation. In fuzzy identity based encryption, only the recipient whose attributes match defined on a *set overlap distance metric* can decrypt a message encrypted with the same identity.

Sahai's work is further extended in the form of *Key Policy Attribute Based Encryption (KP-ABE)*, in which attributes are attached to the ciphertext and a *monotonic* formula is attached with the secret key of user [4]. The KP-ABE was complemented with the Ciphertext-Policy Attribute Based Encryption (CP-ABE) in [5] that aimed to give more power to the sender as compared to KP-ABE. CP-ABE uses the approach of threshold secret sharing [6].

Subsequently, there have been numerous attempts that one can find in the literature to improve the basic CP-ABE scheme too [7-15]. We give an overview of these attempts in section 2. However, all these approaches support only the *static* attributes. It is emphasized that in a typical CP-ABE implementation, attributes play an important role because they essentially determine a user's secret key.

Now, in the real world, the attributes of any entity often undergo periodic updates. For example, if we consider *Harry* as a student at *Stanford* who resides in *Dorm1*. Then, the access restricting the identity of Harry has the attributes *{Name="Harry"; Residence="Dorm1"; Institute="Stanford"}*. Now, when Harry is transferred from *Dorm1* to *Dorm7* say, the corresponding attribute must also change. As mentioned before, the basic CP-ABE and its variants enlisted support only the static attributes.

Realizing the utility of supporting the dynamic attributes, four specific approaches have been proposed that attempt to do so [16-19]. However, as per our observations, these approaches either entail significant overhead or violate the mandatory requirements for the support of the dynamic attributes or lack the flexibility required in periodic updates to the attributes. We further elaborate on these limitations in section 2.1.

Motivated by these limitations, we propose in this paper an improved approach for supporting the dynamic attributes in a CP-ABE that overcomes the limitations, mentioned above. To the best of our knowledge, this is a simple yet unique approach for handling the dynamic attributes.

The rest of the paper is organized as follows: In section 2 we discuss the existing approaches with limitations. Section 3 gives the preliminaries, which we use in remaining part of paper. Section 4 discusses the proposed approach. Section 5 gives the formal security proof for proposed system. Section 6 gives the experimental setup for the proposed AB-OR protocol. Finally, conclusion and references are at the end.

Our contribution: As we observed that the approaches till now has some kind of limitation so that they are not trustworthy as well as not applicable in real life. To deal with this problem we proposed a new approach in which CA extract the old values from the secret key and change the value of required attribute and replace the new value with the old value and give secret key to user.

2 Background

In [7] authors proposed the approach based on AND gate only where attributes have negative and positive values. In [8] authors support the non-monotonic access structure. In [9] authors give the construction of bounded CP-ABE scheme. In [10] authors the construction of hidden access structure where no one can able to see the policy attached which ciphertext. In [11-13] authors give the construction of constant length ciphertext approach. In [14] authors add the *of* operators in access policy. In [15] authors use the attribute set to denote the particular ID of a user. The approach gives till now deal with static attributes, in other word once the value for particular attribute was set in secret key than it never delete/modify/add, which makes them useless in real world scenario where user ‘s attribute value changes frequently. To overcome this limitation we have to use the dynamic attributes i.e. the attributes that change its value in a secret key.

2.1 Dynamic Attributes

In this section, we had discussed the different approaches that deal with dynamic attributes with limitations. In the CP-ABE scheme, attributes play an important role because they are attached to a user’s secret key. However, in the real world, the attributes never remain static. For example, consider Harry as a student at Stanford and resides in the Dorm1. Thus, access restricting identity of Harry has the attributes {Name=”Harry”; Residence=”Dorm1”; Institute=”Stanford”}. The attribute most likely to change herein is Residence. Thus, when Harry is transferred from Dorm1 to Dorm7 say, the corresponding attribute must also change. This is shown in Figure 1.

Thus, it is necessary to deal with the dynamic attributes and update the same as desired. Let us investigate different approaches that can be applied to deal with the dynamic attributes. At the minimum, the following are the requirements of any dynamic attribute-updating scheme:

1. One must be able to *add/delete/update* any dynamic attribute, in any number and at any desired instance.
2. One must be able to assign any desired value to a chosen dynamic attribute.
3. The modification of one attribute value must be independent of the same to the other.
4. A user must not be compelled to re-produce the proof of old attributes and their values when updating the same.

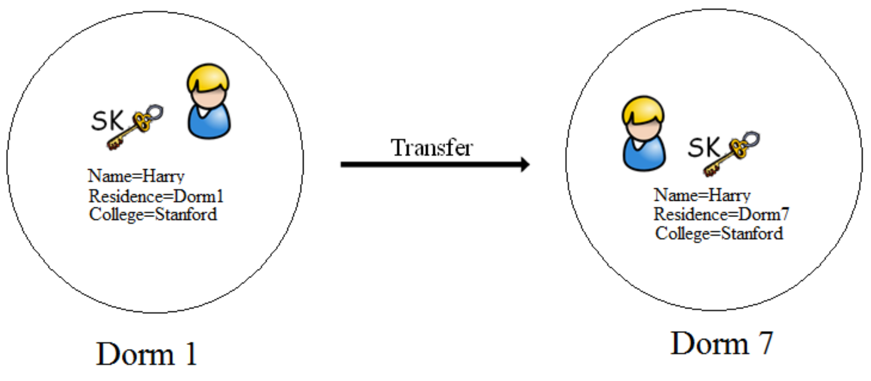


Fig. 1. Dynamic attribute example

1.1.1 Naïve Approach [16]

We discuss a naïve approach that facilitates the dynamic attribute update with reference to the previous example. When Harry is transferred from Dorm1 to Dorm7, he must follow the following steps through CA.

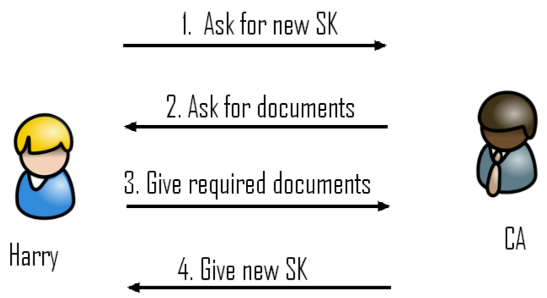


Fig. 2. Naïve approach for dynamic attributes

As we can see in Fig. 2, Harry is required to submit all his documents for proving his attributes. If the set of attributes associated with Harry is a large set, then the CA is required to verify all the associated documents (and a user is required to produce) before regenerating the new attributes. This obviously puts greater overhead and increases the complexity on the CA. In addition, it violates one of the requirements for dynamic update enlisted before.

1.1.2 Fading function based approach [16] [18]

This approach was first proposed by Chen et. al. first in [16] [18]. It advocates changing the value of attributes without changing the secret key and is based on the notion of a fading function. A fading function viz. Z=F(x, y): takes two parameters x and y and outputs a unique value Z based on that. The concept of fading function is used as described below. Consider x as the original attribute value and y as the time parameter.

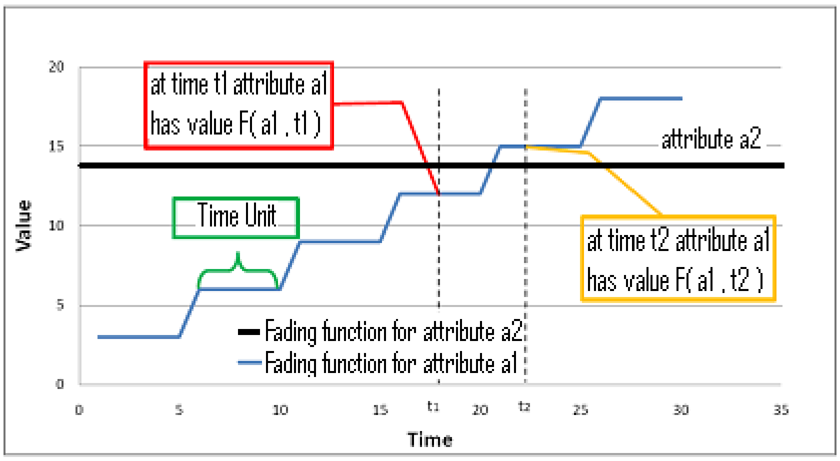


Fig. 3. Static and Dynamic Attribute [16]

As shown in Fig. 3, a1 is a dynamic attribute a2 is a static attribute in the system. Since, the dynamic attribute value may change periodically with time, we see that at t_1 the value is F(a1, t_1) and at t_2 it is F(a2, t_2).

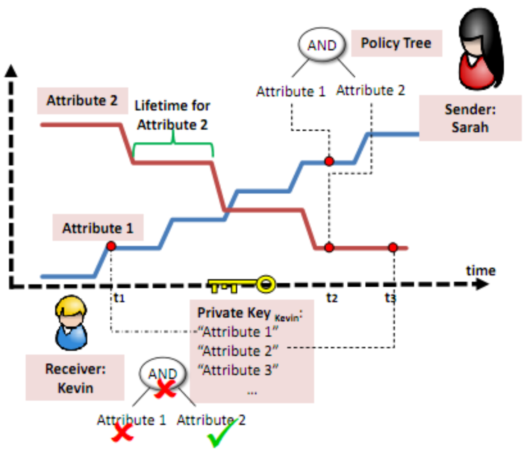


Fig. 4. Working of dynamic attributes [18]

As shown in Fig. 4, assuming that the sender Sarah wants to send a message to receiver Kevin, she encrypts a message with two attributes using AND gate at time t_2. Kevin tries to decrypt the message using F(attribute1, t_1) and F(attribute2, t_3). Note that F(attribute2,t_3) = F(attribute2,t_2) whereas F(attribute1,t_2) $\neq$ F(attribute1,t_1).

Hence, Kevin is not able to decrypt the message because one of the attributes does not match.

Let us revisit the example of *Harry*. If we know that at a particular time Harry will be in *Dorm1*, then we can compute *Z=F (Dorm1, 10:30)* and put Z in the policy condition. Thus, the attribute "Residence" is dynamic because it has many values relative to its temporal context.

However, this scheme has subtle intricacies. What if that the temporal context under reference has already passed? In such cases, user would never be able to decrypt the ciphertext thereafter. In the above example, Harry would never able to decrypt data before or after 10:30. Thus, the update to the attribute values is not aptly taken care off in this approach. This violates all the characteristics of dynamic attributes, specified earlier.

1.1.3 Fixed values for dynamic attributes [17]

In this approach, a provision is made at the time of initialization, in terms of a list of a fixed number of values – one of which to be chosen as a dynamic attribute. Let us take an example - say all the *Institutes* in a state in India, can be classified into three categories viz. *fully-state-funded, partly–state-funded or self-financed*. If we add one more attribute *type* in the example *Harry*, then for this newly added attribute we have a list of three possible values to choose from it. In this scenario, we can make a list containing these 3 values at key generation time. Similarly, if *Stanford* has ten different Dormitories, then one would make a list of these 10 different values for *Residence* attribute. This list has to be setup again at the time of key generation. However, this adds extra overhead during key generation. In addition, one does not have the flexibility to add a new attribute value, as and when desired. A side effect associated is that it is not feasible to have a large set of attribute values to choose from, in terms of initial provisioning.

1.1.4 Using access tree [19]

In CP-ABE to encrypt a message we require to generate the access tree. Hence, in this approach, we use access tree in such a way that it supports dynamic nature of an attribute.

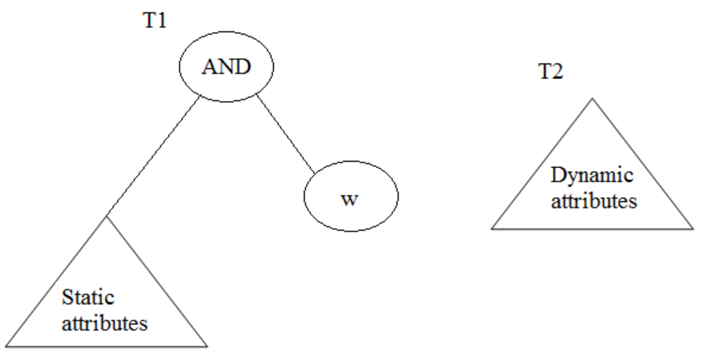


Fig. 5. Access tree representing static and dynamic attributes [19]

As shown in Fig. 5, we can divide the entire access tree in two parts T_1 for static attributes and dummy node w (represent by T_2) and T_2 for dynamic attributes. To encrypt a message m the sender performs the operation CT=Encrypt (M, PK, T_1) where M=message, PK=public key of CA.

As shown in Fig. 3, there is a component in CT that will correspond to the leaf node w in T_1. Let us refer to this component as CT_w. The rest of CT is referred to as CT^* i.e. CT= [CT_w, CT^*]. Now the sender re-encrypts CT_w and gets CT'_w=Encrypt (CT_w, PK, T2). Hence, the final ciphertext will be [CT*,CT'_w]. Then, the receiver has to get the value of the dynamic component from the local key server and apply it to CT'_w, whereas apply its original secret key to CT*.

For our example, if student Harry is transferred from *Dorm1* to *Dorm7* then he requires to get the secret key component for dynamic attribute from CA for *Dorm7*. Thus there is no need to change the existing secret key. As we observe, this approach requires double encryption i.e. one for static and one for dynamic attributes that inflicts addition overhead on the sender side. Naturally, it becomes too inefficient when a sender has multiple messages to communicate. If there are multiple dynamic attributes than user requires to keep the private key components for each dynamic attribute.

Thus, none of approaches for dynamic attributes fulfills the 4 basic requirements as stated before. Therefore, we have proposed the scheme that provides basic requirements.

3 Preliminaries

3.1 Notations

Most cryptographic protocol requires randomness, for example generating random secret key. We use $x \in_R A$ to represent the operation of selecting element x randomly and uniformly from element set *A*. At some places we use "ϕ" to denote the NULL output. This paper deals with the computational security setting where security was defined based on the string length. For £ ∈ N where *N* is the set of natural numbers, $1^{£}$ denotes the strings of length £. If *x* is a string then $|x|$ denotes its length, e.g. $|1^{£}|$ = £.

3.2 Attribute based encryption

Definition 1 (Bilinear map). Assume G1, G2 and G3 are three multiplicative cyclic group of some prime order p. A bilinear map e: G1 × G2 → G3 is a deterministic function that takes as input one element from G1, one element from G2, and output an element in group G3, which satisfies the following criteria.

a) Bi-linearity : For all x ∈ G1, y ∈ G2, a,b ∈ Z_p , $e(x^a,y^b)=e(x,y)^{ab}$.
b) Non degeneracy: $e(g1, g2) \neq 1$ where *g1* and *g2* are generator of G1 and G2 respectively.
c) *e* must be computed efficiently.

Definition 2 (Discrete Logarithm Problem). Given two group elements *g* and *h*, find an integer a ∈ Z_p such that $h=g^a$ mod p whenever such integer exist.

Definition 3 (Access Structure). Let (A1,A2,…,An) be a set of attributes. A collection $A \subseteq 2^{\{A1,A2,\ldots An\}}$ is monotone if ∀B,C : if B ∈ A and B ⊆ A then C ∈ A. An (monotone) access structure is a (monotone) collection A of non-empty subsets of (A1,A2,…,An), i.e. $A \subseteq 2^{\{A1,A2,\ldots An\}} \setminus \{\emptyset\}$. The sets in A are called authorized and the sets that are not in A called unauthorized sets.

3.3 Proposed construction:

It consists of five polynomial algorithms as follows. The **Setup**, **Encrypt** and **Decrypt** are same as in [5].

1. **Setup**: It will take implicit security parameter and output public parameter PK and master key MK.
2. **KeyGen** (MK, L): The key generation algorithm runs by CA. It takes as input the master key of CA and the set of attributes L for user, then generate the secret key SK.
3. **KeyUpdate** (MK, SK, old_value, new_value) : The key updation algorithm runs by CA. It takes as input the master key of CA, old SK and old attribute value *old_value*, and then updates the secret key SK by updating (add/delete/update) *old_value* with *new_value*.
4. **Encrypt** (PK, M, A) : The encryption algorithm takes as input the message M, public parameter PK and access structure A over the universe of attributes. Generate the output CT such that only those users who had valid set of attributes that satisfy the access policy can only able to decrypt. Assume that the CT implicitly contains access structure A.
5. **Decrypt**(PK,CT,SK) : The decrypt algorithm run by user takes input the public parameter, the ciphertext CT contains access structure A and the secret key SK contain of user attribute set S. If S satisfies the access tree then algorithm decrypt the CT and give M otherwise gives "ϕ".

3.4 Selective security Game

Set-Up: The challenger runs *Setup* and gives MPK to A.

Phase 1: A sends an attribute list L to the challenger for a *KeyGen* query with attribute list L, where $L \mid\neq W^*$. The challenger answers with a secret key for these attribute list L. Note that these queries can be repeated adaptively.

Challenge: A sends two equal length messages M_0 and M_1 to the challenger. The challenger selects $\mu \in_R \{0, 1\}$, and runs $C^* =$ *Encrypt*$(PK, M\mu, W^*)$.The challenger gives the ciphertext C^* to A.

Phase 2: Attacker A can sends q number of queries $S_1, S_2, \ldots, S_q$ to *KeyUpdate* regarding add/update/delete of attributes in secret key. After add/update/delete the L' is modified list of attributes. The challenger answers with a secret key for these attributes. Note that at any point of time L' $|\neq$ W*, and these queries can be repeated adaptively.

Guess: A outputs a guess $\mu' \in \{0, 1\}$.

The advantage of A is define as Adv(A):= $|\Pr(\mu' = \mu) - 1/2|$.

4 Proposed approach

The *Setup*, Encrypt and Decrypt algorithms are same as in [5]. Let us look at *KeyGen* algorithm for CP-ABE as given in [5]. The key generation algorithm *KeyGen* (MK, S) will take master secret key (MK) of CA and the required attribute set S of user. The algorithm first selects the random $r \in_R Z_p$ and for each attribute $j \in S$, it generates a random r_j. Then it computes the key as follows.

$$SK = (D = g^{(\alpha+r)/\beta}, \forall j \in S : D_j = g^r H(j)^{r_j}, D'_j = g^{r_j})$$

Now with reference to the example discussed in section 1, if assume we require two attributes "Harry" and "Student." Then, the SK = (D, D1,D1',D2,D2') for some random *r*. Assume that after completing the study in *Stanford* user Harry joins the same *Institute* as a faculty, so he is required to change the 2nd attribute from student to faculty. Thus, in an ideal scenario, user would perhaps give all his relevant credentials, based on which the CA is required to regenerate the SK.

In our approach, the user is required to give old SK and proof for new attribute value; that is used by the CA to update the required attribute value if present in old SK and give the new SK. A user is not required to give proof for old attributes. In other words, in our approach, the old attribute values are collected from old SK and update the required attribute.

If we apply this system in semi trusted environment than CA is required to give random $r \in_R Z_p$, D and g^r to a user in encrypted form so that only one of the semi trusted entity would be able to decrypt it and update the required attribute. However, in this case, if one semi trusted authority is compromised, then the entire system would fail. As we observed the limitation of original CP-ABE is, the $r \in_R Z_p$, D and g^r of one user can also be applied to other user for updating or adding the attributes. Therefore, if two users had the same *r* value then they can collude to make collusion attack. To resolve this issue we can put ID of user with them. The proposed algorithms are shown in Table 1 and Table 2. One question can arise that, what to do with old SK? CA can add the time parameter with attributes in SK to make the old SK invalid after particular time.

KeyGen : Run by CA	
Input: attributes of user and parameters of CA.	
Output: Secret Key (SK) of user and encrypted file.	
1	User give related document and ask for SK.
2	CA verifies the documents and apply standard *KeyGen* algorithm to generate the SK for user.
3	CA generates secret.txt file and put $r \in_R Z_p$, D and g^r in that file.
4	CA outputs SK and E_{key}(secret.txt).

Table 1. : KeyGen Algorithm

KeyUpdate : Run by CA	
Input: SK, old attribute value, new attribute value and parameters of CA	
Output: Updated SK of user	
1.	User provide document for new value of attribute and give his SK.
2.	CA verifies the documents.
3.	CA decrypts the secret.txt file and get $r \in_R Z_p$, D and g^r. Here CA can also generate a new $r \in_R Z_p$, D and g^r.
4.	CA checks for particular attribute in SK, if found at *i* then replace with new attribute and generate D_i and D_i'. Put them into SK of user and regenerate the new SK.
5.	If user wants to add new attribute then CA generate D_{new} and D_{new}' for that attribute value and regenerate the SK.
6.	CA outputs SK and secret file (if $r \in_R Z_p$, D and g^r are regenerated)

Table 2. : KeyUpdate Algorithm

5 Security Analysis

Theorem 1: The advantage of attacker in the selective game for proposed approach of dynamic attribute is $O(q^2/p)$.

Proof: In the selective game the challenge ciphertext C can be either $M_0\, e(g,g)^{\alpha s}$ or $M_1\, e(g,g)^{\alpha s}$. Here we can consider a modified game in which attacker has to distinguish between $e(g,g)^{\alpha s}$ and $e(g,g)^{\theta}$, where $\theta \in_R Z_p$. there will be teo\wo cses (1) if adversary can disnstinguish between $e(g,g)^{\alpha s}$ and $e(g,g)^{\theta}$ (2) if not distinguish. In both cases, the adversary has at least ${}^{\epsilon}/_{2}$ advantage in modified game. Our goal is to bound adversary's advantage in the modified game.

Setup: the simulation algorithm chooses $\alpha, \beta \in_R Z_p$. Here if $\beta = 0$ (which can happen with probability ${}^{1}/_{p}$) than *Setup* is aborted as in actual scheme. The public parameters $h = g^{\beta}, f = g^{1/\beta}$ and $e(g,g)^{\alpha}$ will be given to adversary.

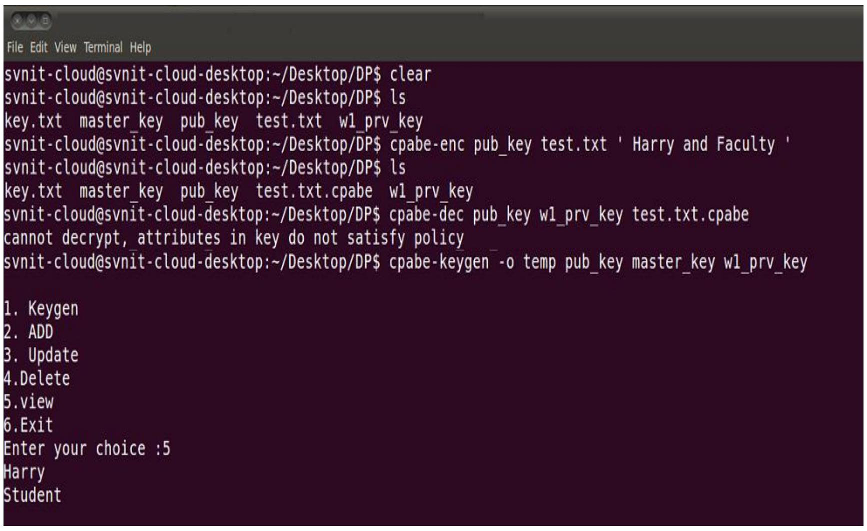
Phase-1: For every evaluation of H (Probabilistic universal hash function) on **Key-Gen** query for S_i which is called by adversary A. The simulator takes $t_i \in_R Z_p$ and returns g^{t_i} as the response to the H (i). For the list of attributes L, simulator generates $r^{(i)} \in_R Z_p$ and give $SK = (D = g^{(\alpha + r^{(i)})/\beta}, \forall j \in L : D_j = g^{r^{(i)} + t_i r_j^{(i)}}, D_j' = g^{r_j^{(i)}})$, where $\forall j \in L : r_j^{(i)} \in_R Z_p$. The SK is given to adversary.

Challenge: Adversary submit access structure W* such that L |≠ W* and two messages $M_0, M_1 \in G$ to simulator. The simulator selects $s \in_R Z_p$ and then flips a coin and get outcome μ so it will do $e(g,g)^{\mu}$ and set *s* as root of the access tree A. The simulator selects $\gamma_i \in_R Z_p$ for each relevant attribute *i*, here γ_i are the parts of share *s* based on Shamir's secret sharing scheme [6]. This process will apply recursively till the leaf nodes. The simulator gives C'=$e(g,g)^{\mu}$, $C = h^s, \forall i \in L\ C_i = g^{\gamma_i}$ and $C_i' = g^{\gamma_i t_i}$. Here simulator gives $E_{key}(r^{(i)})$ to attacker where *key* is only known to simulator and *E* is an encryption algorithm, which no PPT adversary can break without valid *key*.

Phase-2: Now attacker sends q number of queries to **KeyUpdate** for add/delete/update an attribute in SK. Here it assumed that any add/delete/update query will make L to L' such that L' $|\neq$ W*. If L' $|=$ W* or any combination of previous add/delete/update query make L' such that L' $|=$ W* than that query is simply aborts by simulator as in original scheme. At each operation (1) Add(i) : $L' = L + i$ (2) Delete(i) : $L' = L - i$ (3) Update(i,i') : $L' = L + i' - i$. For update/delete operation simulator generates new $\gamma_i \in_R Z_p$ for attribute i . Collusion attack will not be possible as $r^{(i)}$ binds every user attribute of SK. The rest of the proof will be as in [5].

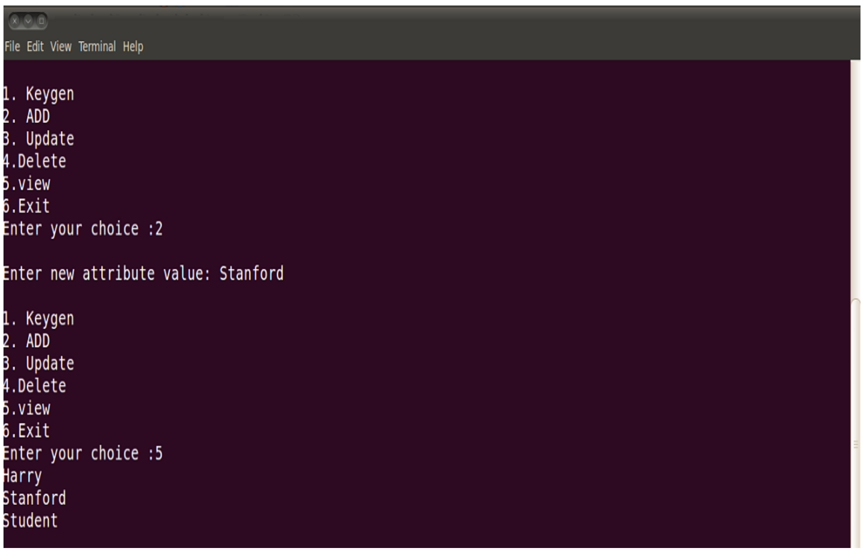
6 Implementation and Analysis

We had modified the CP-ABE toolkit code [20] to add the new functionalities, implement the system and evaluate empirically. We generate a new r $\in_R Z_p$, D and g^r for user's dynamic attributes. We had combined the **KeyGen** and **KeyUpdate** algorithms. We also use *temp* as a temporary file. Fig.6 to Fig. 9 is the snapshots of our application to illustrate its working. Here *w1_prv_key* is the secret key of the user.



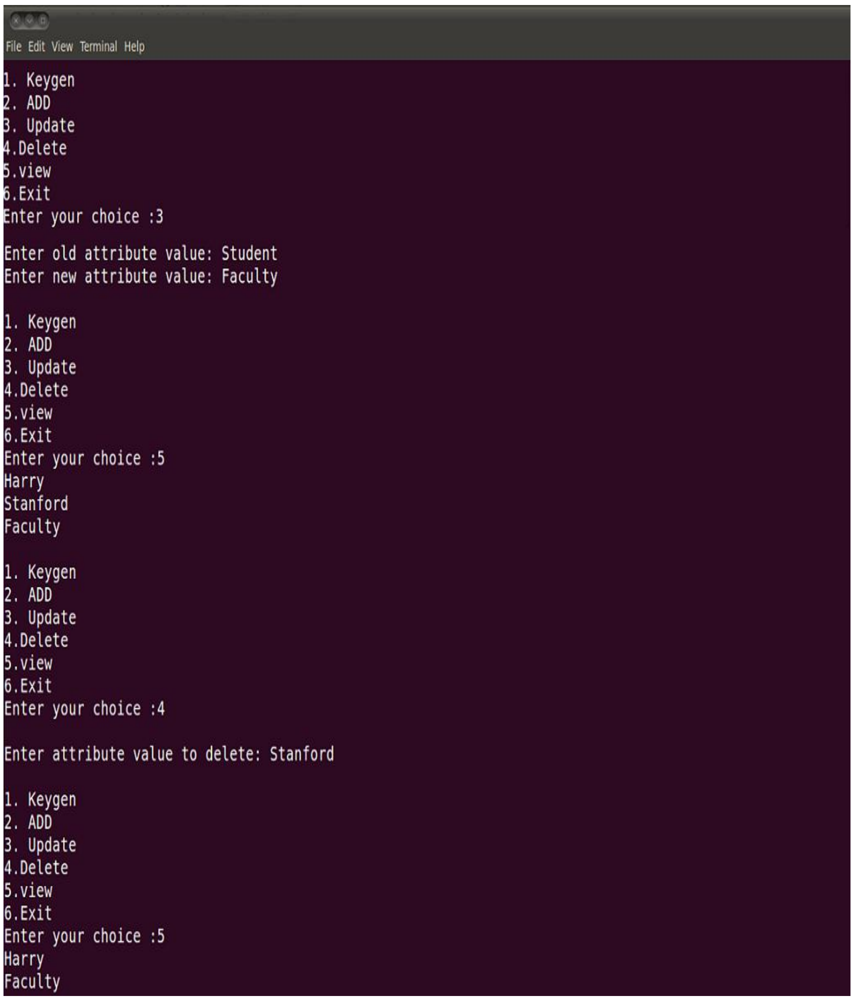
```
svnit-cloud@svnit-cloud-desktop:~/Desktop/DP$ clear
svnit-cloud@svnit-cloud-desktop:~/Desktop/DP$ ls
key.txt  master_key  pub_key  test.txt  w1_prv_key
svnit-cloud@svnit-cloud-desktop:~/Desktop/DP$ cpabe-enc pub_key test.txt ' Harry and Faculty '
svnit-cloud@svnit-cloud-desktop:~/Desktop/DP$ ls
key.txt  master_key  pub_key  test.txt.cpabe  w1_prv_key
svnit-cloud@svnit-cloud-desktop:~/Desktop/DP$ cpabe-dec pub_key w1_prv_key test.txt.cpabe
cannot decrypt, attributes in key do not satisfy policy
svnit-cloud@svnit-cloud-desktop:~/Desktop/DP$ cpabe-keygen -o temp pub_key master_key w1_prv_key

1. Keygen
2. ADD
3. Update
4.Delete
5.view
6.Exit
Enter your choice :5
Harry
Student
```

Fig. 6. secret key generation



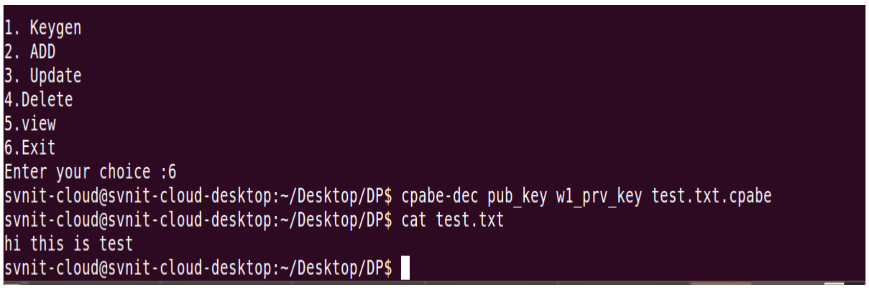
```
1. Keygen
2. ADD
3. Update
4.Delete
5.view
6.Exit
Enter your choice :2

Enter new attribute value: Stanford

1. Keygen
2. ADD
3. Update
4.Delete
5.view
6.Exit
Enter your choice :5
Harry
Stanford
Student
```

Fig. 7. adding new attribute



```
1. Keygen
2. ADD
3. Update
4.Delete
5.view
6.Exit
Enter your choice :3

Enter old attribute value: Student
Enter new attribute value: Faculty

1. Keygen
2. ADD
3. Update
4.Delete
5.view
6.Exit
Enter your choice :5
Harry
Stanford
Faculty

1. Keygen
2. ADD
3. Update
4.Delete
5.view
6.Exit
Enter your choice :4

Enter attribute value to delete: Stanford

1. Keygen
2. ADD
3. Update
4.Delete
5.view
6.Exit
Enter your choice :5
Harry
Faculty
```

Fig. 8. updating and deleting existing attribute



```
1. Keygen
2. ADD
3. Update
4.Delete
5.view
6.Exit
Enter your choice :6
svnit-cloud@svnit-cloud-desktop:~/Desktop/DP$ cpabe-dec pub_key w1_prv_key test.txt.cpabe
svnit-cloud@svnit-cloud-desktop:~/Desktop/DP$ cat test.txt
hi this is test
svnit-cloud@svnit-cloud-desktop:~/Desktop/DP$
```

Fig. 9. Applying new secret key

Now after viewing the list of attributes user or CA can encrypt the message using the specific policy and check the validity of attributes.

7 Conclusion and future work

In this paper, we outline an approach for dynamic updates of the attributes in existing secret key of a user. In existing approaches dynamic attributes depends on time, and contain only fixed values, that can be difficult to maintain for large number of users, each one having a large number of attributes. In the proposed approach, we can add/delete/update any number of attributes in a secret key of user and make less burden for the CA by taking old attributes from secret key itself.

The limitation of the approach proposed, is that it requires updating the secret key of a user. Currently we assume that only CA can add/delete/update the attributes. In future, we can consider the Multi authority based setup to deal with the issue. We can apply this updating feature to user revocation, group signature, etc. and may get better understanding of protocol. Currently CA can change any attribute but in future, we can divide into static and dynamic such that CA can update only dynamic attributes. Our current work is focus on this.

References

1. Rivest, R., Shamir, A., and Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. Comm. A CM 21, 2 (Feb. 1978), 120-126.
2. Shamir. Identity Based Cryptosystems and Signature Schemes. In Advances in Cryptology CCRYPTO, volume 196 of LNCS, pages 47-53.Springer, 1984.
3. Sahai A,Waters B. Fuzzy identity-based encryption. Proceeding of EUROCRYPT 2005. Berlin : Springer 2005,LNCS 3494:457-473.
4. Goyal V,Pandey O,Sahai A, et al. attribute based encryption for fine-grained access control of encrypted data. Proceedings of the 13th ACM conference on Computer and communications security. New York: ACM, 2006:89-98.

5. Bethencourt J,Sahai A,Waters B. ciphertext-policy attribute-based encryption. Proceedings of the 2007 IEEE Symposium on Security and Privacy (S&P 2007), Piscataway:IEEE, 2007: 321-334.
6. Shamir. How to share a secret. Communications of ACM, 22(11):612C613,1979.
7. Cheung l, Newpost C. Provably secure ciphertext policy ABE, Proceedings of the 14th ACM conference on computer and communications. New York : ACM,2007:456-465
8. R. Ostrovsky, A. Sahai and B. Waters, Attribute-Based Encryption with Non-Monotonic Access Structures. Cryptology ePrint Archive, Report 2007/323, 2007. Available: http://eprint.iacr.org/
9. Goyal v, Jian A, Pandey O, et al. Bounded Ciphertext policy attribute based encryption. Proceedings of ICALP 2008. Berlin :Springer,2008, LNCS 5126:579 – 591.
10. Nishide T., yoneyama K., ohta K.: 'Attribute-based encryption with partially hidden encryptor-specified access structures'. Applied cryptography and network security, New York, USA, June 2008 (LNCS, 5037),pp. 111–129.
11. Herranz, Javier and Laguillaumie, Fabien and Ràfols, Carla. Constant Size Ciphertexts in Threshold Attribute-Based Encryption. LNCS:6056,Springer 2010:19-34.
12. Emura, K., Miyaji, A., Nomura, A., Omote, K., Soshi, M.: A ciphertext-policy attribute-based encryption scheme with constant ciphertext length. In: Bao, F., Li, H., Wang, G. (eds.) ISPEC 2009. LNCS, vol. 5451, pp. 13–23. Springer, Heidelberg (2009).
13. Zhibin Z., and Dijiang H. On Efficient Ciphertext-Policy Attribute Based Encryption and Broadcast Encryption. http://eprint.iacr.org/2010/395.pdf.
14. Luan Ibraimi, Qiang Tang, Pieter Hartel and Willem Jonker. Efficient and Provable Secure 1Ciphertext-Policy Attribute-Based Encryption Schemes. In: Bao, F., Li, H., Wang, G. (eds.) ISPEC 2009. LNCS, vol. 5451, pp. 1–12. Springer, Heidelberg (2009).
15. Bobba, r., khurana, H., and prabhakaran, M. Attribute-sets: A practically motivated enhancement to attribute-based encryption. In ESORICS, M. Backes and P.

Ning, Eds. Lecture Notes in Computer Science, vol. 5789. Springer 2009: 587–604.

16. Nanxi Chen, Mario Gerla Dynamic Attributes Design in Attribute Based Encryption. Annual Conference of ITA (ACITA), University of Maryland University College, September. 2009.
17. S. G. Weber, "Securing first response coordination with dynamic attribute-based encryption," n Proceedings of 2009 World Congress on Privacy, Security, Trust and the Management of e-Business (CONGRESS 2009). IEEE Computer Society, 2009, pp. 58 – 69.
18. Chen, Nanxi; Gerla, Mario; Huang, Dijiang; Hong, Xiaoyan; Secure, selective group broadcast in vehicular networks using dynamic attribute based encryption. Ad Hoc Networking Workshop (Med-Hoc-Net), 2010 The 9th IFIP Annual Mediterranean , IEEE Computer Society, 2010, pp. 1-8, http://dx.doi.org/10.1109/MEDHOCNET.2010.5546877.
19. M. Chuah, S. Roy, I. Stoev. Secure Descriptive Message Dissemination in DTNs. Proceeding MobiOpp '10 Proceedings of the Second International Workshop on Mobile Opportunistic Networking. ACM 2010.
20. Bethencourt J,Sahai A,Waters B., CP-ABE toolkit, Available online at http://acsc.cs.utexas.edu/cpabe/ cpabe toolkit.