International Journal for Modern Trends in Science and Technology
Volume: 03, Issue No: 02, February 2017
ISSN: 2455-3778
http://www.ijmtst.com

# Homomorphic Encryption Technique on Minimizing Overhead of CCP-ABE

P.Venkata Hari Prasad[1] | Dr.K.Gangadhara Rao[2] | Dr.B.Basaveswara Rao[3]

[1]Associate Professor, Department of CSE, DIET | Research Scholar, Acharya Nagarjuna University, Guntur, India.
[2]Associate Professor, Department of CSE, Acharya Nagarjuna University, Guntur, India.
[3]Department of CSE, Acharya Nagarjuna University, Guntur, India.

**To Cite this Article**
P.Venkata Hari Prasad, Dr.K.Gangadhara Rao and Dr.B.Basaveswara Rao, "Homomorphic Encryption Technique on Minimizing Overhead of CCP-ABE", *International Journal for Modern Trends in Science and Technology*, Vol. 02, Issue 11, 2016, pp. 75-82.

## ABSTRACT

*Shared Content Distribution is one of the major focusing area in many organizations to protect information which is distributed within the network or outside network. Information in the cloud or server provides access to all the legal users within the specified domain people. But the information which is broadcasted or distributed must need to provide security from the unauthorized access. Existing attribute based and traditional encryption techniques doesn't give better solution due to increase in message size and communication overhead. Attribute based mechanism uses tree based structure to check the user's policy. With the increase in attribute and user policies, tree size growth also exponentially increases which is very difficult to the user to access the information. In constant based attribute based encryption communication overhead occurs due to the increase in users network access. In this proposed work a pattern based access policy is introduced to reduce the communication overhead and time to access the information. This approach includes three phases i.e Setup phase, Key Generation Phase, Encryption and Decryption.*

*This system uses pattern based policy access structure with homomorphic encryption mechanism. Experimental results give minimum time in terms of computation overhead, time to generate secret key, time to access the shared information and storage overhead in terms of encrypted data and key sizes.*

***Index Terms***—*Encryption, Policy, Communication overhead, cryptography.*



## I. Introduction

Today's computing technologies have attracted larger numbers of people to store their private data on third-party servers either for simplicity of sharing or for cost saving. Naturally, trusted authorities want to make their private data only accessible to legal users. In many cases, additionally it is desirable to access services so that data access policies are defined over user attributes. It's pretty easy to foresee these kinds of security concerns and requirements would become more severe concern in cloud environment wherein individuals, organizations, and businesses may outsource their various types data recovery, which includes highly sensitive data, directly into cloud/server. Traditional access control strategies are not going to be as effective under attribute policy to discover the data owners to trusted domains, and to discover the third-party storage servers themselves is probably not fully trustworthy.

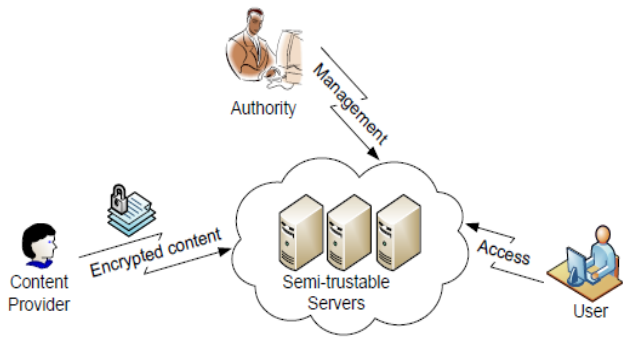



Fig.1, shows semi-trustable proxy servers will always be available for providing various types of content services. These servers, are trusted authority execute the duties assigned by legitimate parties inside the system. In accordance to this assumption, proxy re-encryption technique with CP-ABE,enables the authority to delegate most laborious tasks of user revocation to proxy servers without leaking any confidential information to the people.

On each revocation event, the authority just generates several proxy re-encryption keys and transmits them to be able to proxy servers. Proxy servers will update secret keys for those users but the a person to be revoked. Unlike solutions suggested by existing CP-ABE schemes, our construction places minimal load upon the authority upon each revocation event, and of course the authority is able to freely revoke any attribute of users whenever possible. The ultimate requirement is that proxy servers should stay online and perform honestly[1].

In both larger or smaller scale content distribution systems it can be often crucial to ensure data helpful to simply a select multitude of users. In commercial content distribution, just for instance, an organization might need because of its digital media to be available simply to paying customers. Throughout the smaller scale, suppose a department's faculty need to access the academic transcripts of graduate applicants. If electronic copies of one's transcripts were stored toward the department's file server, if they simply be accessible from the faculty and students. In fact its's often incredibly important to guard the identities of this users who are capable of access protected content. Students receiving an email from an instructor notifying all students failing the class would likely like to keep their identities private. Commercial sites often not wish to disclose identities of consumers because competitors might use this info for targeted advertising. When an employee is up for promotion, a business might wish to downplay that is on his promotion committee and hence that is ready to read his performance evaluation.

Traditional CP-ABE scheme, once users obtain the credentials typically from a trusted authority at the beginning of setup phase, the access ability is often valid for individuals that will probably even break the confidential rules by abusing these private information. Upon detecting those malicious activities, without any revocation mechanism embedded, the operating trusted authority has to rebuild in the whole system. Therefore, revocation mechanism really should be designed straight into the system right from the start rather than just being added following the other issues are addressed, simply because it requires meticulous planning on where functionality should be placed and how to reduce the computational and communication costs.

ABE thus is envisioned as a possible important tool for addressing the challenge of secure and fine-grained data sharing and access control. In ABE, the encryption keys and/or ciphertexts are labeled with sets of descriptive attributes defined for your system users. As well as a particular user private key can decrypt a particular ciphertext only if them match. A celebration could encrypt a document to every one users who tend to have a certain set of attributes drawn given by a predefined attribute universe. ABE, on the other hand, is often being criticized for its high scheme overhead as extensive pairing operations are usually required.

Besides fine-grained access policy, there is an increasing need to protect user privacy in today's access control systems. In some critical circumstances, the access policy itself could be sensitive information. Therefore, we propose an attribute – based encryption scheme where encryptor specified access policies are hidden. Even the legitimate decrypted user cannot obtain the details about the access policy associated with the encrypted data [2].

## II. Related Work

[3] Proposed traceability for predicate encryption schemes to broadcast encryption. Traceability allows a group manager to predict malicious insiders who leak their personal secret keys to an adversary, or to determine which authorized users' keys have been compromised.

Cryptographic access control over untrusted storage is investigated in both cryptography community and networking community. In

cryptography community, Broadcast Encryption (BE) was introduced by Fiat and Naor . Compared with traditional one-to-one encryption schemes, BE is very efficient. Based on tradeoffs between key storage and ciphertext storage overhead, existing BE schemes can be generally categorized into the following classes: (i) constant ciphertext, linear public and/or private key on number of total receivers ; (ii) linear ciphertext on number of revoked receivers, constant (or logarithm) public and/or private key, (iii) sub-linear ciphertext, sub-linear public and/or private key.

Cheung and Newport [4] raised a provably secure CP-ABE scheme which is proved to be secure under the standard model. Further on, their scheme supports AND-Gates policies which deals with negative attributes explicitly and uses wildcards in the ciphertext policies.

Goyal et al. [5] put forward a bounded ciphertext policy attribute-based encryption in which a general transformation method was proposed to transform a KP-ABE system into a CP-ABE one by using "universal" access tree. However, the parameters of ciphertext and private key sizes will grow up in the worst case.

[6].CCP-ABE: We construct an efficient Constant Ciphertext Policy Attribute Based Encryption (CCP-ABE) scheme that can encrypt a message with an AND-gate access policy with wildcards. Moreover, CCP-ABE supports non-monotonic data access control policy. ABBE: Based on CCP-ABE, we present an Attribute Based Broadcast Encryption (ABBE) scheme. Compared with existing BE schemes,

ABBE is flexible as it uses both descriptive and non-descriptive attributes, which enables a user to specify the decryptors based on different abstraction levels, with or without exact information of intended receivers. Moreover, ABBE demands less storage overhead compared to existing BE schemes. Both schemes uses Tree based structures for policy verification and retrieval. As the number of policies increases tree structure increases which is very difficult to check the policy.This approach has limitation when the message size increases or doesn't supportable to media files

## III. Proposed Work

**Proposed approach follows four phases:**

1. **Setup**
2. **Key Generation**
3. **Encryption Process**
4. **Decryption Process**

**Setup:**

Setup algorithm takes $\alpha, \beta, \gamma$ ,G,e with G= $G_\alpha \times G_\beta \times G_\gamma$ .p,q,r are the elements in Zp. First compute $g_p, g_q, g_r$ are the generators of $G_\alpha, G_\beta, G_\gamma$ respectively. Following algorithm generates setup parameters for the given Total policy pattern(T.P). Given Total policy pattern is divided into three patterns with AND($\wedge$),OR($\vee$),*. Algorithm takes Attribute list Attlist, Policy list polilist, operators list oplist and operators position list poslist as input and generates hashcodes of three policy patterns of policy list.

**Setup Algorithm:**

**Input:**
List:=Polilist, Attlist, Oplist,Poslist.
**Procedure:**
Oplist[]:={$\wedge$,$\vee$,*};
**Step1:** Read Polilist,Attlist,Oplist and Total Policy Pattern(Tp).
**Step2:** Identifying the operators position in Tp.

```
for(i=0;i<Oplist.length;i++)
do
  for(j=0;j<TP.length;j++)
  do
  if(Oplist[i]==Tp[j])
  then
    pos[i]:=j;
  else
    continue;
  end if
  done
done
```

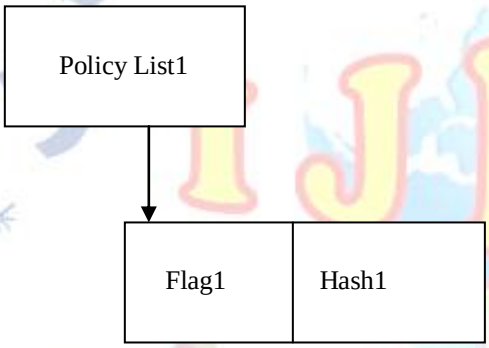
**Step3:**

```
Len1:=pos[1]-pos[0];
Len2:=pos[2]-pos[1];
Len3:=Tp.length-pos[2];
Copy three pattern policy values in three lists as
List pat1=null;
List pat2=null;
List pat3=null;
pat1.add(substring(1,len1));
pat2.add(substring(len1+1,len2));
  pat3.add(substring(len2+1,Tp.length));
```

**Step4:**
Process AND policy pattern for Hash calculation.

```
String temp1:=pat1.get(substring(1,len1));
```

```
Boolean[] flag1=false;
String[] hash1=null;
Tokenizing pattern1 policies with space delimiter
as
String[] att=StringTokens(Attlist," ");
String[] poli1=StringTokens(temp1," ");
for(i=0;i<poli1.length;i++)
do
  for(j=0;j<att.length;j++)
  do
    if(poli1[i].equals(att[j]))
    then
      flag1[i]=true;
      hash1[i]=MD5(poli1[i]);
    else
    exit;
    end if
  done
done
```




List Mapping of Pattern1.

```
count:=0;
for(i=0;i<poli1.length;i++)
do
  if(flag1[i]==true)
  count++;
  end if
done
if(count==poli1.length)
then
  hash(pat1):={concat(hash1[i]," ")};
i=0.....poli1.length.
end if
```

**Step5:**

Process OR policy pattern for Hash calculation.

```
String temp2:=pat2get(substring(len1+1,len2));
Boolean[] flag2=false;
String[] hash2=null;
Tokenizing pattern2 policies with space delimiter
as
String[] att=StringTokens(Attlist," ");
String[] poli2=StringTokens(temp2," ");
for(i=0;i<poli2.length;i++)
do
  for(j=0;j<att.length;j++)
  do
    if(poli2[i].equals(att[j]))
    then
      flag2[i]=true;
      hash2[i]=MD5(poli2[i]);
    else
    continue;
    end if
  done
done
for(i=0;i<poli2.length;i++)
do
  if(flag2[i]==true)
  hash(pat2):={concat(hash2[i]," ")};
i=0.....poli2.length
  end if
done
```

**Step6:**

Process * policy pattern for Hash calculation.

```
String
temp3:=pat1.get(substring(len2+1,Tp.length));
Boolean[] flag3=false;
String[] hash3=null;
Tokenizing pattern3 policies with space delimiter
as
String[] att=StringTokens(Attlist," ");
String[] poli3=StringTokens(temp3," ");
for(i=0;i<poli3.length;i++)
do
  for(j=0;j<att.length;j++)
  do
    int check:=0;int p:=0,q:=0;
    int n=att[j].length;
    int m=poli3[i].length;
          while(p<n)
          do
            if(poli3[i].charAt(q)==att.charAt(p))
            then
              if(q=m-1)
              then
                check=p-m+2;
                p++;
                q++;
              else if(q>0)
              then
                q=0;
              else
                p++;
```

```
                end if
              done
            if(check!=-1)
            then
              flag3[i]:=true;
              hash3[i]=MD5(poli3[i]);
            else
              continue;
            end if
      done

done

for(i=0;i<poli3.length;i++)
do
  if(flag3[i]==true)
        hash(pat3):={concat(hash3[i],” “)};
       i=0.....poli3.length
  end if
done
```

$H_1^{'}$=HextoDecimal(Hash(pat1)); i=0…pat1.length.

$H_2^{'}$=HextoDecimal(Hash(pat2)); i=0…pat2.length.

$H_3^{'}$=HextoDecimal(Hash(pat2)); i=0…pat2.length.

$S^{'} = H_1^{'} + H_2^{'} + H_3^{'}$;

Public Key:={$S^{'}, g_p, g_q, g_r, G_\alpha, G_\beta, G_\gamma, H_1^{'}, H_2^{'}, H_3^{'}$; };

Master key:={$\alpha, \beta, \gamma$}; known to T.A

**Key Generation :**

Key Generation algorithm will take set of attributes , Policy pattern hash values as input and returns Secret key as output.

Each user is associated with secret key and it will be generated using three pattern keys as

$K_{1,i} = g_p^{1/(S^{'}+\alpha)}$; i=0.....pat1.length;

$K_{1,j} = g_q^{1/(S^{'}+\beta)}$;j=0.....pat2.length;

$K_{1,k} = g_r^{1/(S^{'}+\gamma)}$;k=0....pat3.lenght;

Secret key:={Tp,Hash(pat1), Hash(pat2), Hash(pat3), $K_{1,i}, K_{1,j}, K_{1,k}$};

**Encryption Process:**

Input: Public key, Policy Patterns, Message;
Procedure:

Public Key:={$S^{'}, g_p, g_q, g_r, G_\alpha, G_\beta, G_\gamma, H_1^{'}, H_2^{'}, H_3^{'}$; };

Calculations:

$C_0 = g_p^{S^{'}}$;

$C_0^{'} = g_p^{(\alpha+\beta+\gamma)}$; where $\alpha, \beta, \gamma \in G_\alpha, G_\beta, G_\gamma$;

$C_{1,i} = g_p^{H_2^{'}+H_3^{'}}.g_p^{H_1^{'}+\alpha}$ i:=0.....pat1.length;

$C_{2,j} = g_p^{H_1^{'}+H_3^{'}}.g_p^{H_2^{'}+\beta}$ j:=0.....pat2.length;

$C_{3,k} = g_p^{H_1^{'}+H_2^{'}}.g_p^{H_3^{'}+\gamma}$ k:=0.....pat3.length;

Encryption algorithm encrypts the message using policy pattern structures. Algorithm uses three patterns with homomorphic encryption and decryption process. Additive and Multiplicative homomorphism takes two inputs and generate secure encrypted values as output. Homomorphic encryption and decryption uses $C_0, C_0^{'}$ as input.

Additive Homomorphic Encryption

$Enc(M_1 + M_2) = Enc(M_1) + Enc(M_2)$;

Multiplicative Homomorphic Encryption

$Enc(M_1.M_2) = Enc(M_1).Enc(M_2)$;

$M_1 := C_0$;

$M_2 := C_0^{'}$;

Enc($M_1$):=Enc($C_0$)=($C_0 + \gamma * \beta$)mod n where n=$\alpha * \beta$;

Enc($M_2$):=Enc($C_0^{'}$)=($C_0^{'} + \gamma * \beta$)mod n where n=$\alpha * \beta$;

$Enc(M_1 + M_2)$:=Enc($C_0 + C_0^{'}$)=Enc($C_0$)+Enc($C_0^{'}$);

:=($C_0 + \gamma * \beta$)mod n+($C_0^{'} + \gamma * \beta$)mod n

Enc($M_1.M_2$):=Enc($C_0.C_0^{'}$):=Enc($C_0$).Enc($C_0^{'}$);

:=($C_0 + \gamma * \beta$)mod n. +($C_0^{'} + \gamma * \beta$)mod n;

Cipher Text CT={Tp, $H_1^{'}, H_2^{'}, H_3^{'}$ ,M.e($Enc(M_1 + M_2)$ ,Enc($M_1.M_2$))

,{$C_{1,i}, C_{2,j}, C_{3,k}$},C};

Cipher Text CT is publicly available to all the attribute policy holders. This CT will be decrypted only those users who has exact policy matching patterns.

**Decryption Process:**

Input(CT,Secret Key):

Secret key:{ Tp,Hash(pat1), Hash(pat2), Hash(pat3), $K_{1,i}, K_{1,j}, K_{1,k}$};

CT:= {Tp, $H_1^{'}, H_2^{'}, H_3^{'}$ ,M.e( $Enc(M_1+M_2)$ ,Enc( $M_1.M_2$ ))

,{ $C_{1,i}, C_{2,j}, C_{3,k}$ },C};

Decryption Process:

Consider M.e( $Enc(M_1+M_2)$ ,Enc( $M_1.M_2$ )).e( $C_{1,i}$ , $K_{1,i}$ ).e( $C_{2,j}$ $K_{1,j}$ ).e( $C_{3,k}$ , $K_{1,k}$ ) to extract added pattern policy.

$Enc(M_1+M_2)$= Enc( $C_0+C_0^{'}$ )=Enc( $C_0$ )+Enc( $C_0^{'}$ );

:=( $C_0+\gamma*\beta$ )mod n+( $C_0^{'}+\gamma*\beta$ )mod n

Enc( $M_1.M_2$ ):=Enc( $C_0.C_0^{'}$ ):=Enc( $C_0$ ).Enc( $C_0^{'}$ );

:=( $C_0+\gamma*\beta$ )mod n. +( $C_0^{'}+\gamma*\beta$ )mod n;

Dec( $Enc(M_1+M_2)$ ):=(Enc( $C_0+C_0^{'}$ )) mod $\alpha$

:= (( $C_0+\gamma*\beta$ )mod n+( $C_0^{'}+\gamma*\beta$ )mod n) mod $\alpha$

:= $C_0+C_0^{'}$ ----- > (1)

Dec(Enc( $M_1.M_2$ )):=(Enc( $C_0.C_0^{'}$ ):=Enc( $C_0$ ).Enc( $C_0^{'}$ ))mod $\alpha$ ;

:=(( $C_0+\gamma*\beta$ )mod n. +( $C_0^{'}+\gamma*\beta$ )mod n) mod $\alpha$ ;

:= $C_0.C_0^{'}$ -------- > (2)

Solving eq-(1) and eq (2) we will get $C_0$ and $C_0^{'}$.

Now e( $Enc(M_1+M_2)$ ,Enc( $M_1.M_2$ ))=e( $C_0$ , $C_0^{'}$ )

:=e( $g_p^{S^{'}}$ , $g_p^{(\alpha+\beta+\gamma)}$ )

:= $e(g_p,g_p)^{S^{'}(\alpha+\beta+\gamma)}$

Consider

$$\prod_{i=1}^{n} e(C_{1,i},K_{1,i}) = e(g_p^{H_2^{'}+H_3^{'}}.g_p^{H_1^{'}+\alpha}, g_p^{1/(S^{'}+\alpha)})$$

$$= e(g_p^{H_2^{'}+H_3^{'}+H_1^{'}+\alpha}, g_p^{1/(S^{'}+\alpha)})$$

$$= e(g_p^{S^{'}+\alpha}, g_p^{1/(S^{'}+\alpha)})$$

$$= e(g_p,g_p)^{(S^{'}+\alpha)/(S^{'}+\alpha)}$$

$$= e(g_p,g_p) = 1$$

$$\prod_{j=1}^{n} e(C_{2,j},K_{1,j}) = e(g_p^{H_2^{'}+H_3^{'}}.g_p^{H_1^{'}+\beta}, g_p^{1/(S^{'}+\beta)})$$

$$= e(g_p^{H_2^{'}+H_3^{'}+H_1^{'}+\beta}, g_p^{1/(S^{'}+\beta)})$$

$$= e(g_p^{S^{'}+\beta}, g_p^{1/(S^{'}+\beta)})$$

$$= e(g_p,g_p)^{(S^{'}+\beta)/(S^{'}+\beta)}$$

$$= e(g_p,g_p) = 1$$

$$\prod_{k=1}^{n} e(C_{3,k},K_{1,k}) = e(g_p^{H_1^{'}+H_2^{'}}.g_p^{H_3^{'}+\gamma}, g_p^{1/(S^{'}+\gamma)})$$

$$= e(g_p^{H_1^{'}+H_2^{'}+H_3^{'}+\gamma}, g_p^{1/(S^{'}+\gamma)})$$

$$= e(g_p^{S^{'}+\gamma}, g_p^{1/(S^{'}+\gamma)})$$

$$= e(g_p,g_p)^{(S^{'}+\gamma)/(S^{'}+\gamma)}$$

$$= e(g_p,g_p) = 1$$

Substituting in CT we get

$\Rightarrow$ M.e( $Enc(M_1+M_2)$ ,Enc( $M_1.M_2$ )).e( $C_{1,i}$ , $K_{1,i}$ ).e( $C_{2,j}$ $K_{1,j}$ ).e( $C_{3,k}$ , $K_{1,k}$ )

$\Rightarrow$ M. e( $C_0$ , $C_0^{'}$ ).e( $C_{1,i}$ , $K_{1,i}$ ).e( $C_{2,j}$ $K_{1,j}$ ).e( $C_{3,k}$ , $K_{1,k}$ )

$\Rightarrow$ M. $e(g_p,g_p)^{S^{'}(\alpha+\beta+\gamma)}.\prod_{i=1}^{n} e(C_{1,i},K_{1,i})$ .

$\prod_{j=1}^{n} e(C_{2,j},K_{1,j}).\prod_{k=1}^{n} e(C_{3,k},K_{1,k})$

$\Rightarrow$ M. $e(g_p,g_p)^{S^{'}(\alpha+\beta+\gamma)}$ .1.1.1

$\Rightarrow$ M. $e(g_p,g_p)^{S^{'}(\alpha+\beta+\gamma)}$ ----- > (3)

Now Based on the user entered policy A and D parameters may vary as

If user entered policy is in pattern1 then

$D_{1,i} = g_p^{\alpha}$

$A_{1,i} = g_p^{\beta+\gamma}$

If user entered policy is in pattern2 then

$D_{2,j} = g_p^{\beta}$

$A_{2,j} = g_p^{\alpha+\gamma}$

If user entered policy is in pattern3 then

$D_{3,k} = g_p^{\gamma}$

$A_{3,k} = g_p^{\alpha+\beta}$

If user entered policy is in patter1 then decryption follows:

Decryption:= M. $e(g_p, g_p)^{S'(\alpha+\beta+\gamma)}$ /e(C,D*A)

:= M. $e(g_p, g_p)^{S'(\alpha+\beta+\gamma)}$ /e($g_p^{S'}$, $g_p^{\alpha} . g_p^{\beta+\gamma}$ )

:= M. $e(g_p, g_p)^{S'(\alpha+\beta+\gamma)}$ / e($g_p^{S'}$, $g_p^{\alpha+\beta+\gamma}$)

:= M. $e(g_p, g_p)^{S'(\alpha+\beta+\gamma)}$ / e$(g_p, g_p)^{S'(\alpha+\beta+\gamma)}$

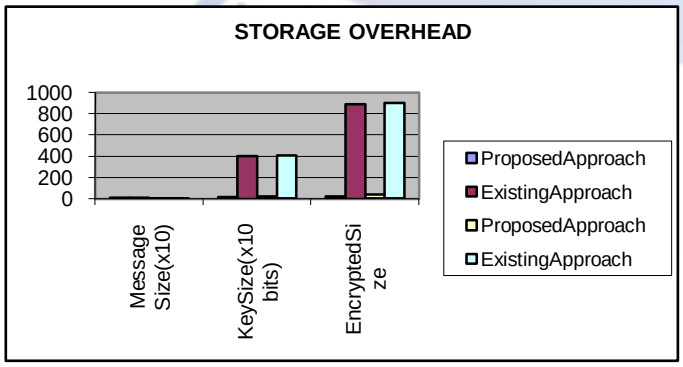
:= M

Similarly other policies can decrypt the original message M.

## IV. Experimental Results

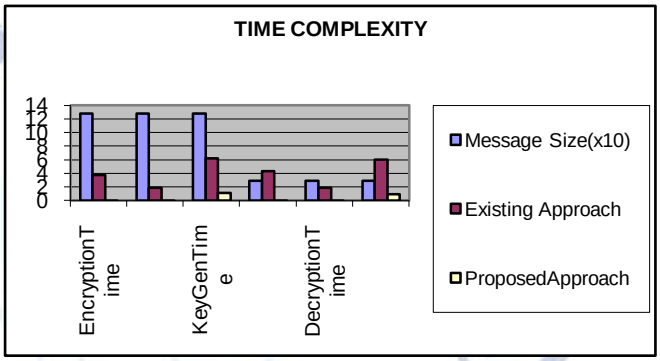
**Storage overhead:**

| | Message Size(x10) | KeySize(x10bits ) | EncryptedSize |
|---|---|---|---|
| **ProposedApproach** | 12.8 | 18.7 | 24 |
| **ExistingApproach** | 12.8 | 406.6 | 890 |
| **ProposedApproach** | 2.88 | 18.7 | 40 |
| **ExistingApproach** | 2.88 | 406.6 | 906 |




**TIME COMPLEXITY:**

| | Message Size(x10) | Existing Approach | Proposed Approach |
|---|---|---|---|
| **EncryptionTime** | 12.8 | 3.77 | 0.008 |
| **DecryptionTime** | 12.8 | 1.909 | 0.012 |
| **KeyGenTime** | 12.8 | 6.193 | 1.12 |
| **EncryptionTime** | 2.88 | 4.334 | 0.008 |
| **DecryptionTime** | 2.88 | 1.907 | 0.014 |
| **KeyGenTime** | 2.88 | 5.99 | 0.964 |




## V. Conclusion

In this proposed work , a robust attribute based algorithm is designed for secure sharing of data between users. This proposed approach performs well for small as well as large datasizes. This system takes constant time for encryption and decryption process as well as key generation process. For encryption and decryption this system uses homomorphic schemes for policy verification process. This system effectively tested on different text messages. This framework effectively works well to other formats like doc,pdf, other datafiles. This system takes less time to access the document information even when large number of users accessing the same shared data through online.

## References

[1] Bit Level Encryption Standard (BLES): Version-I Neeraj Khanna ,International Journal of Computer Applications (0975 – 8887) Volume 52– No.2, August 2012.

[2] A DRM Framework Towards Preventing Digital Piracy,Ravi Sankar Veerubhotla and Ashutosh Saxena, Senior Member, IEEE, 978-1-4577-2155-7/11 2011 IEEE.

[3] Tracing Insider Attacks in the Context of Predicate Encryption Schemes,Jonathan Katz and Dominique Schroder.

[4] Cheung, L.,Newport, C., "Provably secure ciphertext policy ABE", In: Proc. of the ACM Conf. on Computer and Communications Security, NewYork: ACM , 2007.456-465.

[5] Goyal, V., Jain, A., Pandey, O., Sahai, A., "Bounded ciphertext policy attribute based encryption", In: Aceto, L., Damgard, I., Goldberg, L.A., Halld′orsson, M.M., Ing′olfsd′ottir, A., Walukiewicz, I. (eds.) ICALP 2008, PartII.LNCS, vol.5126,pp.579–591.Springer.

[6] On Efficient Ciphertext-Policy Attribute Based Encryption and Broadcast Encryption Zhibin Zhou and Dijiang Huang.