International Journal of Electrical and Computer Engineering (IJECE)
Vol. 7, No. 2, April 2017, pp. 950~960
ISSN: 2088-8708, DOI: 10.11591/ijece.v7i2.pp950-960 

# An Improved Integrated Hash and Attributed based Encryption Model on High Dimensional Data in Cloud Environment

**Satheesh K S V A Kavuri[1], Gangadhara Rao Kancherla[2], Basaveswararao Bobba[3]**
[1]Dhanekula Institute Of Engineering & Technology, Departement of CSE, India
[2,3]Departement of CSE, Acharya Nagarjuna University, India

**Article Info**

***Article history:***

Received Jul 18, 2016
Revised Feb 16, 2017
Accepted Mar 4, 2017

***Keyword:***

Cloud security
CP-ABE
Hardware based encryption
Integrity and hashing
Integrity verification

**ABSTRACT**

Cloud computing is a distributed architecture where user can store their private, public or any application software components on it. Many cloud based privacy protection solutions have been implemented, however most of them only focus on limited data resources and storage format. Data confidentiality and inefficient data access methods are the major issues which block the cloud users to store their high dimensional data. With more and more cloud based applications are being available and stored on various cloud servers, a novel multi-user based privacy protection mechanism need to design and develop to improve the privacy protection on high dimensional data. In this paper, a novel integrity algorithm with attribute based encryption model was implemented to ensure confidentiality for high dimensional data security on cloud storage. The main objective of this model is to store, transmit and retrieve the high dimensional cloud data with low computational time and high security. Experimental results show that the proposed model has high data scalability, less computational time and low memory usage compared to traditional cloud based privacy protection models.



***Corresponding Author:***

Satheesh K S V A Kavuri,
Departement of Computer Science and Engineering,
Dhanekula Institute Of Engineering & Technology,
Vijayawada-521 139, Krishna (Dt.), AP. India.
Email: kns9@live.com

## 1. INTRODUCTION

Cloud refers to storing and accessing the user's private or public data in a remote server space instead of storing it in the local database of their personal computer. Cloud computing is a distributed architecture that provides on-demand, convenient network access to store high dimensional data with configuranle computing resources such as servers, storage and applications. Cloud servers share their computing resources as a service in a distributed manner to the connected clients by means of network connection. These shared resources are offered on demand or customers pay for their usage level [1].

Information security in distributed cloud computing deals with data protection using different encryption and decryption models. Data security involves securing data from being destroyed, lost or modified or corrupted. So, availability and correctness of cloud data must be assured using various encryption and decryption models. Also, data encryption with integrity verification mechanism are familiar models to solve security issue [2]. Various security frameworks with protecting data both from cloud side and client side are implemented in the literature for data security. But computational requirements and processing speed play a vital role in deployment of these cloud security models in cloud computing environment.

Encryption-based access control has several advantages over classical cloud access control. In a classical setup, as depicted in Figure 1, data is stored unencrypted on the server and the user needs to authenticate each time she wants to retrieve data from the server. The server is required to authorize the user's request before it sends the plaintext data to the user. Letting the server authorize the user's requests allows for exible and fine-grained access control. However, the server needs to be trusted and well-protected. Instead of storing the data in plaintext format on the server, one could encrypt the data and store this on the server. This has the advantage that the server is not burdened with the authorization and authentication of users. Moreover, the data can be stored on many|even untrusted|servers, as it is encrypted anyway [3].

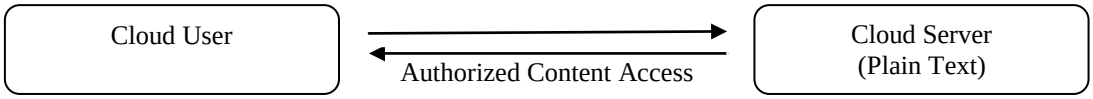


Figure 1. Classical cloud access control

### 1.1. ABE Basic Steps

Setup This algorithm is run before all other algorithms and determines the public parameters (PK) and a master key (MK) for the KGA. The PK determines the set of all possible attributes and all user keys will be derived from the MK [1].

**Key Generation:** The KGA can create new decryption keys for users using its MK. A user's private key (SK) is derived from the MK by randomizing the MK in such a way that the user cannot convert the SK back to the MK. To decide which construction is preferred, we need to estimate how many attributes will be used by a key generation authority (KGA). If each KGA uses a small set of attributes, the small universe seems more desirable, as it is in general less computationally intensive. The ciphertext consists of multiple parts. One of these parts is a randomly chosen secret number operating on the plaintext. The other parts are needed to reconstruct this secret number. Using a secret sharing scheme that splits the secret number into various parts, the access structure is enforced by using these in parts of the ciphertext [4]. To prevent user collusion, each SK is randomized by a unique, user-speci_c number, or, the key is bound to a fixed global identifier (GID) of the user.

Access structures are used to define which users have access to which resources. In the case of attribute-based authentication, attributes determine the authorization level of the user. An access structure can be regarded as a collection of sets of attributes. Each single set describes which attributes are needed to be granted access. As long as the user's attributes satisfy at least one set in the collection, the user is granted access. There are two kinds of access structures: monotonic and non-monotonic. Monotonic access structures ensure that whenever a user would be granted access based on a subset of his attributes, he will be granted access based on all his attributes. This means that no negations of attributes are possible. Nonmonotonic access structures do allow such negation of attributes. Here, the possession of an extra attribute may deny you access. Fuzzy Identity based encryption scheme in which a descriptive attributes set are considered as identity for encryption and decryption process. For the privacy or secret key $k_{AS}$ corresponds to the attribute set S. We can decrypt the data using $k_{AS'}$ corresponds to the attribute set $s'$ and satisfies the condition $|S \cap S'| < d$ ,where d is the minimum number of attributes.

## 2. LITERATURE SURVEY

Integrity and authentication of data in cloud environment are essential issues to ensure that data confidentiality and privacy preserving to the customer’s data or queries. Problem of dealing with user’s queries and encrypted data over cloud environment were discussed widely in research literature [2-5] try to memory integrity checking to address integrity issues by applying Hash tree over memory content. An integrity verification approach [7] in hybrid clouds is applied to support the data migration and scalability service is implemented on limited data. Table 1 shows the various Attribute Based Encryption Schems

Table 1. Various Attribute Based Encryption Schems

| Sl. No | Method | Parameter | Drawbacks |
|---|---|---|---|
| 1 | Provable Data Possession [7] | RSA based homomorphism linear model | It may leak user's data to the third party users |
| 2 | ABE [8] | Setup, Public and secret keys | Doesn't support secure communication in cloud environment |
| 3 | CP-ABE [9] | Attributes, Policies, Key Generators | As the number of attributes size or storage space increases, computational time also increases. |
| 4 | KP-ABE [10] | Key policies, Attributes, Key generation | Fail to construct the access policy patterns for multiple cloud storage services. |

Require a number of exponential generators for private key computation which is a significant computation ovehead.Also it requires Random oracle model which is less secure than other standard models. Both the cipher texts and private keys are labeled with an policy set and attribute set , the decryption will succeed only if there exist at least k common attributes between the cipher text and a private key.

The traditional models ensure data security by using encryption is not optimal in the cloud virtual machines of cloud providers. Although the trusted third party authorities are aware of the malicious insider, they assume that they have limited solutions to overcome these issues. A secured, cost-effective multi-cloud storage method is implemented in cloud environment which controls an economical distribution of information among the available cloud instances to provide the customers with secure storage and data availability. A high performance cloud computing service is implemented that integrates the parallel processing framework and checkpoint infrastructure such as Message passing interface for virtual machines.

In the cloud server attacks, the length of the overlapping runtime of the cloud instances and malicious virtual machines is important to find the network bandwidth. Since limiting the overlapping execution times may degrade the network performance and increase the error rate. *Jung et. al.* [8] proposed a model that encrypts cloud data with user's attributes and send it to the remote cloud server for long term access. The cloud providers not only generate access rights to users, but also compute a secret key using attributes and policies. In this case, KDC is not required. The major issue with this approach is that users can get different keys from different owners for the same attribute, which increases the total number of secret keys to the users along with storage and communication overhead.

## 3. PROPOSED METHOD

In this proposed model, each cloud uploads high dimentional documents as input to our encoded process as shown in Figure 2. In this framework, user's each document is hashed using the proposed cloud hardware based hash algorithm. Computed hash value along with the user's document data are encrypted using the *proposed ABE encryption model*. Our proposed encryption model initializes cloud server parameters for key generation, encryption and decryption process. Encrypted hash and user's data are uploaded to the cloud storage with integrity value in encoded format. Similarly, each user decrypts the encoded hash and data from the cloud storage using the decoding process as shown in Figure 3. In the decoding process, each user's encoded data from the cloud is decrypted using the proposed ABE decryption model. Integrity verification of the decrypted plain text is checked against the decrypted hash value for data modification verification. If the data is not modified then it is accessed to the user's system through the internet.

Figure 4. Shows the overview of the multi-data partition using available cloud servers. In this process, cloud user's multiple documents are partitioned and assigned to the nearest cloud servers using the data block size.

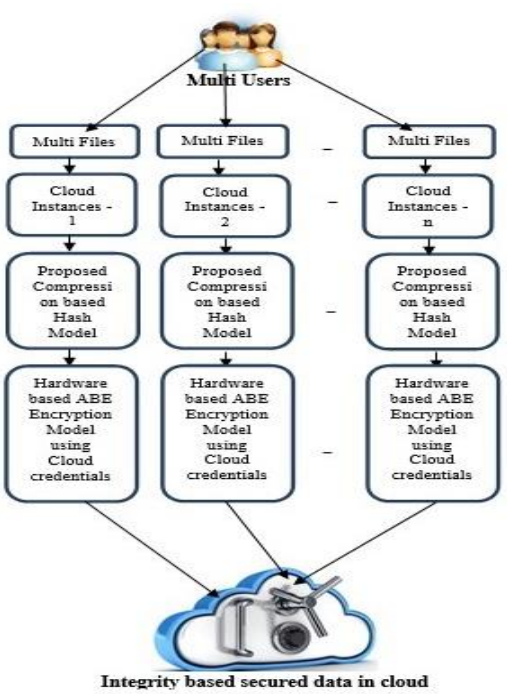


Figure 2. Parallel Multi-Doc Hash based Encryption Model

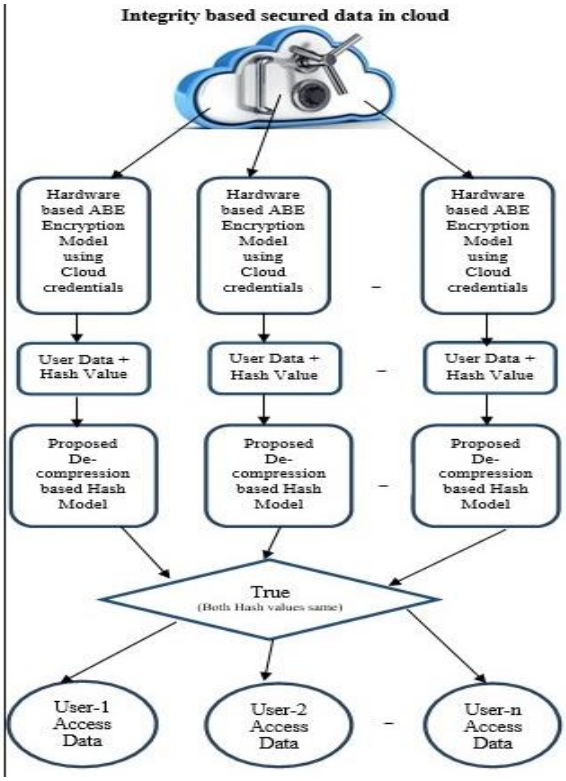


Figure 3. Parallel Multi-Doc Hash based Decryption Model

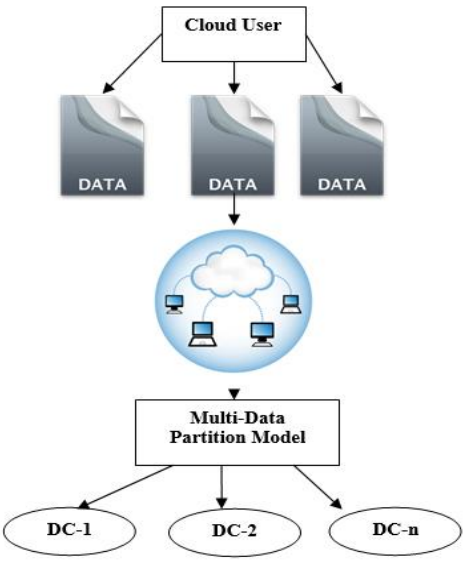


Figure 4. Multi-Data Partition based on Available Data-Centers

### 3.1. Multi-File Data Partition Algorithm

Input: User Data Files

$\delta[]$ be the number of available cloud instance servers.

$\psi[]$ be the list of data storage zones.

Output: File wise partitions.

Procedure:
For each file in the Folder
Read data file f,
Let N bet the number of partitions, each with bytes bits.
Divide data-file into N partitions each with bytes.
N=f.size/1024;
Data[] dt;
For each i=0 to N
Do
For each partition p in N
Do
Let D be the block size in KB, the minimal size can be computed using

$$\text{If}(D > \max\{\frac{\sum_{i=1}^{n}|f[i-1],p]|}{\text{MinBlockSize}}, \frac{\sum_{i=1}^{n}|f[i],p]|}{\text{MaxBlockSize}}\})$$

then
Assign file index f[p,i] to p.
$\eta$[i]=count(f[p,i], $\psi$[i]) // distributing different partition data in cloud servers.
Append data center $\eta$[i]'s instance ID .
Else
Dt[]=f[p,i];
End if
Done
For(int k=0;k< $\eta$.length;k++)
Do
If( $\eta$[k]==empty)
Then
$\eta$[k]=count(Dt[], $\psi$[i]) // distributing different partition data in cloud servers.
Append data center $\eta$[k] to the queing list.
Else
Continue;
done
Done
Done

### 3.2. Parallel multi-doc based Hash Algorithm (PMHA)

Input: Multiple File data partitions.
Output: 1024-bit hash value.
Procedure:
Proposed parallel hash algorithm use 10 rounds to generate compressed hash code. In each round 1024-bit partitioned data as an input. Each round follows three phases namely.

1. Transformed Data Conversion State Representation.
2. Cloud User Policy based Substitution Box.
3. User Access Policy Based Shift Columns in right to left and left to right.

### 3.3. Overview of each phase

**Phase 1:**

In this phase, user partitioned data is converted into sub-partitions as follows:

Let $F_iP_0, F_iP_1, F_iP_2, F_iP_3, ..F_iP_N$ are the ith file data partitions with 1024 bit size. Each partition is represented in the form of list as shown below.

$$F_0P_0 = F_0d_0^0, F_0d_1^0, F_0d_2^0 ....F_0d_{1024}^0 \cdots\cdots F_0P_{n_1} = F_0d_0^{n_1}, F_0d_1^{n_1}, F_0d_2^{n_1} ....F_0d_{1024}^{n_1}$$

$$F_1P_1 = F_1d_0^0, F_1d_1^0, F_1d_2^0 ....F_1d_{1024}^0 \ldots\ldots\ldots F_1P_{n_2} = F_1d_0^{n_2}, F_1d_1^{n_2}, F_1d_2^{n_2} ....F_1d_{1024}^{n_2}$$

$$\vdots$$

$$F_NP_M = F_Nd_0^N, F_Nd_1^N, F_Nd_2^N ....F_Nd_{1024}^N \ldots\ldots\ldots F_0P_{n_1} = F_0d_0^{n_1}, F_0d_1^{n_1}, F_0d_2^{n_1} ....F_0d_{1024}^{n_1}$$

For each file partition $F_i P_j$

Do

*Each file Partition is divided into N blocks of 64 bits' block size. Here w is 8 blocks. So* $F_0P_0$ *can be further partitioned into sub-partition as:*

$$F_0P_0 -> SP_0^0, SP_1^0, SP_2^0 ...SP_{15}^0$$
$$SP_0^0 = Sd_0^0, Sd_0^1, Sd_0^2 ....Sd_0^{63}$$
$$SP_1^0 = Sd_1^0, Sd_1^1, Sd_1^2 ....Sd_1^{63}$$
$$SP_2^0 = Sd_2^0, Sd_2^1, Sd_2^2 ....Sd_2^{63}$$
$$.$$
$$.$$
$$,$$
$$SP_n^0 = Sd_n^0, Sd_n^1, Sd_n^2 ....Sd_n^{63}$$

Similarly, $F_NP_M -> SP_0^N, SP_1^N, SP_2^N ...SP_7^N$ .

**Non-Linear Sub-Partition Substitution**

For each sub-partition

Do

$$TSP_i = [a*(SP_i)^2 + b*SP_i + c] \bmod 3$$

i=0….16

Where

a=number of AND policies.
b=number of OR policies.
c=number of ANY policies.

Done

**Phase 2**

In this phase, transformed sub-partition data is taken as source data. User access policy data is represented as state matrix. For each dynamic user access policy, random dynamic S-Box is generated to each authorized cloud user.

**Phase 3**

In this phase, input data are shifted from left to right in column wise using authorized cloud user parameters. Depends on the user access in the policy structure, three parameters are calculated. These three parameters are used to transform columns left to right and then similarly shift row wise in the reverse order.

**Key Generation Process**

**Input:** Total User Access Policies

**Procedure:**

*Step 1: Construct 1024 bits state matrix using Total User Access Policies.*
*Step 2: Select 1024 bits randomly from 1024 bits using pattern / permutation matrix.*
*Step 3: Select 1024 bits is assigned to each round function of hash algorithm.*
*Step 4: Repeat steps 1 to 3 until 10 rounds.*

**Encrytion and Decryption 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, operator's list oplist and operators position list poslist as input and generates hashcodes of three policy patterns of policy list.

**Input:**

List:=Polilist, Attlist, Oplist,Poslist. Hardware parameters

**Procedure:**

$H_1^{'}$ =*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.*

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

*Public Key=* $\{ S^{'}, g_p, g_q, g_r, G_\alpha, G_\beta, G_\gamma, H_1^{'}, H_2^{'}, H_3^{'}; \}$*; [1]*

*M*aster 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

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.

For each pair of file partition f[i] data

Do

For each character j in f[i]

$M_1$ =f[i][j];

$M_2$ = f[i+1][j];

Additive Homomorphic Encryption

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

Multiplicative Homomorphic Encryption

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

$M_1 := C_0$;

$M_2 := C_0^{'}$; // |P| is partition size

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 |P|

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 |P|

Cipher Text CT={ T.P, $H_1^{'}, H_2^{'}, H_3^{'}$,M.e($Enc(M_1 + M_2)$,Enc($M_1.M_2$)) ,{ $C_{1,i}, C_{2,j}, C_{3,k}$ },C};

Done

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.

$$e(g_p, g_p)^{S^{'}(\alpha+\beta+\gamma)} . \prod_{i=1}^{|P|} e(C_{1,i}, K_{1,i}) . \prod_{j=1}^{|P|} e(C_{2,j}, K_{1,j}) . \prod_{k=1}^{|P|} e(C_{3,k}, K_{1,k})$$

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

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

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 the 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 to M.

## 4. RESULTS AND ANALYSIS

All experiments are performed with the real time Amazon aws cloud servers and client configurations as Intel(R) CPU 2.13GHz, 4 GB RAM, and the minimum OS platform is Microsoft Windows 7 Professional (SP2). This framework requires third party libraries cpabe, abe, amazon aws, and jama.etc.

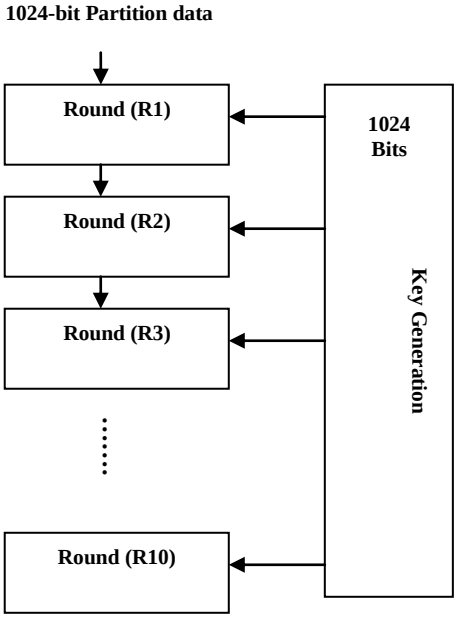


Figure 5. Proposed Parallel Hashing Process

Table 2. Comparison of Hash based Encryption and DecryPtion models

| Algorithm | DataSize (KB) | HashTime (ms) | EncryptionTime (ms) | DecryptionTIme (ms) |
|---|---|---|---|---|
| CPABE+MD5 | >1500 | 4746 | 7330 | 5766 |
| KPABE+SHA12 | >1500 | 5844 | 5866 | 5432 |
| FHEncrytion+SHA256 | >1500 | 6834 | 7955 | 7198 |
| DUPHA+HybridABE | >1500 | 2365 | 3686 | 3519 |
| ParallelHash+Hybrid ABE | >3000 | 2178 | 3286 | 3316 |

From the Table 2, it is clear that proposed parallel ABE based encryption and decryption model has less computational time compared to traditional models in cloud environment. From the Figure 5, it is clear that proposed parallel multi-doc hash model has less computational time compared to traditional models in cloud environment. From the figure 6, it is clear that proposed parallel ABE based encryption and decryption model has less computational time compared to traditional models in cloud environment. From the Table 3, it is clear that proposed parallel hash based ABE encryption and decryption model has less cloud storage computation compared to traditional models ABE on cloud environment. From the Figure 7, it is clear that proposed parallel hash based ABE encryption and decryption model has less cloud storage computation compared to traditional models ABE on cloud environment.

Table 3. Memory storage comparison with traditional ABE models encryption and decryption time

| Algorithm | DataSize (KB) | Cloud Storage (KB) |
|---|---|---|
| CPABE+MD5 | 500 | 545 |
| KPABE+SHA12 | 500 | 574 |
| FHEncrytion+SHA256 | 500 | 541 |
| DUPHA+HybridABE | 500 | 417 |
| ParallelHash+Hybrid ABE | 500 | 359 |

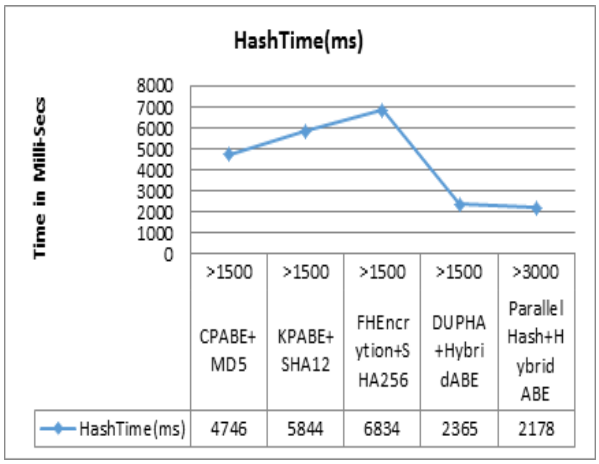


Figure 5. Proposed parallel multi-doc hash computation

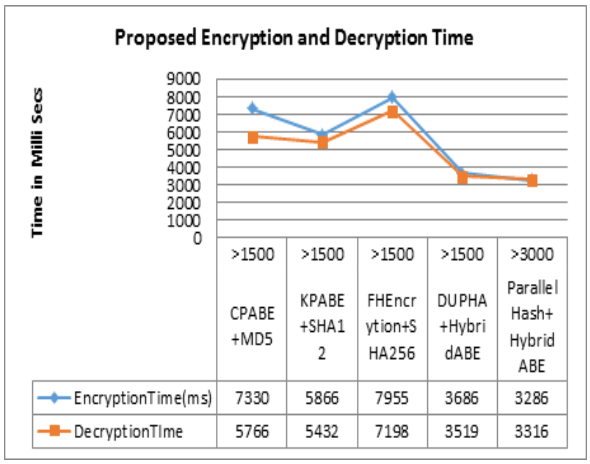


Figure 6. Proposed parallel multi-doc

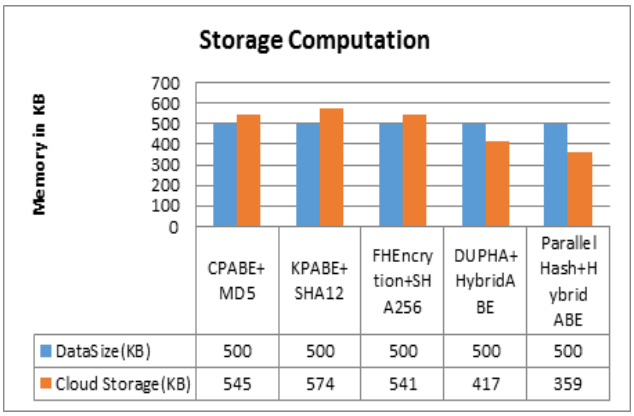


Figure 7. Proposed parallel multi-doc storage computation

## 5. CONCLUSION

Cloud data outsourcing through un-authorized clients and distributed systems are exponentially increasing cloud hardware and software resources. Cloud environment provides on demand resource allocation from a shared pool of hardware and software resources. With more and more cloud based applications are being available and stored on various cloud servers, a novel multi-user based privacy protection mechanism need to design and develop to improve the privacy protection on high dimensional data. In this paper, a novel integrity algorithm with attribute based encryption model was implemented to ensure confidentiality for high dimensional data security on cloud storage.The main objective of this model is to store, transmit and retrieve the high dimensional cloud data with low computational time and high security.Experimental results show that the proposed model has high data scalability, less computational time and low memory usage compared to traditional cloud based privacy protection models.

## REFERENCES

[1] L. Ibraimi, *et al.*, "Efficient and provable secure ciphertext-policy attribute-based encryption schemes," *Proc. 5th Int. Conf. Inf. Secur. Pract. Exper.*, vol. 5451, pp. 1-12.

[2] J. Hur, "Improving security and efficiency in attribute-based data sharing," *IEEE Trans. Knowl. Data Eng.*, vol/issue: 25(10), pp. 2271-2282, 2013.

[3] J. Lai, *et al.*, "Attribute-based encryption with verifiable outsourced decryption," *IEEE Trans. Inf. Forensics Security*, vol/issue: 8(8), pp. 1343-1354, 2013.

[4] Hur, "Attribute-based secure data sharing with hidden policies in smart grid," *IEEE Transactions on Parallel and Distributed Systems*, vol/issue: 24(11), 2013.

[5] S. Hohenberger and B. Waters, "Online/offline attribute-based encryption," *Public-Key Cryptography PKC*, vol. 8383, 2014.
[6] J. K. Liu and J. Zhou, "An efficient identity-based online/offline encryption scheme," *ACNS*, pp. 156C167, 2009.
[7] F. G. Y. Mu and Z. Chen, "Identity-based online/offline encryption," *Financial Cryptography*, pp. 247C261, 2008
[8] T. Jung, *et al.*, "Control cloud data access privilege and anonymity with fully anonymous attribute-based encryption," *IEEE Trans. on Info. Forensics and Security*, vol/issue: 10(1), pp. 190-199, 2015.
[9] J. Li, *et al*., "Securely outsourcing attribute-based encryption with checkability," *IEEE Trans. on Parallel and Distributed Systems*, vol/issue: 25(8), pp. 2201-2210, 2014.
[10] J. Han, *et al.*, "Improving privacy and security in decentralized ciphertext-policy attribute-based encryption," *IEEE Trans. on Info. Forensics and Security*, vol/issue: 10(3), pp. 665-678, 2015.