IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-5, May 2018 E-ISSN: 2347-2693

Implementation of TVES for Enhancing Performance of Attribute Based Encryption and Collaboration in the Cloud Storage

Salim Y. Amdani.[1*], Sohel A. Bhura [2], Akshada D. Yevatkar [3]

[1*]Dept. of CSE, B.N. College of Engineering, Pusad, India
[2] Dept. of CSE, B.N. College of Engineering, Pusad, India
[3] ME student, Dept. of CSE, B.N. College of Engineering, Pusad, India

[*]*Corresponding Author: a_yevatkar@outlook.com,*

Available online at: www.ijcseonline.org

Accepted: 15/May/2018, Published: 31/May/2018

Abstract— Cloud computing is a set of different types of hardware and software activities that work collectively to deliver many aspects of computing to the end user as an online services. It provides on demand scalable and cost effective services for storing documents. However with all of these features it enters with number of challenges associated in utilizing the cloud securely. Achieving unbreakable document or outsourced data security is an important issue to consider. For this purpose encryption has been suggested, with the popularity of using cryptography to store data in the cloud Attribute Based Encryption receives widespread attention of the researchers. Also to provide security to the outsourced documents, Attribute based encryption is an excellent solution. On the other side of cloud environment, to increase productivity of the cloud, collaboration with more users is important. But the encryption limits the admission of newly authorized user to acquire the data because, when the ciphertext is need to be share with others legally there will be necessity of extending an original access policy specified in ABE. In this process, at the time of extending policy, data uploader had to download ciphertext, modify the access policy as per current requirement and then reupload the data with modified access permissions. This significantly increases the processing time to extend access policy and consumes server space unnecessarily. To address above problem we have introduce modified attribute based encryption with time varying encryption scheme for document encryption which is based on the research of Time Varying Camellia Encryption Algorithm for the cloud storage environment and a novel solution for access permission management in which we will use metadata file to store access permission. Finally we discuss the evaluation result made in between existing work and proposed system.

Keywords— Time varying encryption, Extended Access Control, Collaboration in cloud, Attribute Based Encryption.

I. INTRODUCTION

The cloud data security and privacy are major concern for cloud users nowadays. Cloud computing itself has a number of issues standing in the way of largely adoption of this technology. Several common issues can be seen in the considerable literature work [1-3]. Confidentiality can be managed by encrypting a data at client side before outsourcing it to the cloud. But once the data is encrypted, it will restrict the users to collaborate in the cloud hence the notion of extending access control [4]. In the context of providing data or information security in cloud data storage to achieve it perfectly the algorithm opted for data encryption should be able to withstand against any type of vulnerability. Symmetric encryption and Asymmetric encryption are basic types of cryptography. In symmetric encryption, encryption and decryption keys are same whereas in case of Asymmetric encryption, keys are different for encryption and decryption purpose. AES, 3DES are examples of symmetric encryption and the RSA is an example of asymmetric encryption. Attribute based encryption(ABE) proposed by Amit Sahai and Brent Waters [5] is imposed for "one to many" encryption, so that only intended users will get an access to the ciphertext. The attributes select how the data is encrypted and the users with corresponding keys should be able to access the ciphertext. Most of the ABE system uses symmetric encryption for the encryption phase of data. The most important part of symmetric encryption is block cipher generation, many of transport protocols use the block encryption as the underlying security, since it encrypts the plaintext into block ciphertext with highspeed, however the fact that we cannot deny is repetition of plaintext. In case of some part of plaintext might get repeated and the attackers make the analysis of ciphertext to be repeatedly generated, then it would be a dangerous threat to the data security and attackers will get the entry point to access the whole system. Normally what happens in leaking the data security is, hackers or attackers would perform some researches over the attack objects in order to learn basic framework of the

encrypted information, which has been send by the data uploader. Therefore hackers may guess the repeated information's plaintext messages by making tentative attacks on trial and error basis. This can led to vulnerability of data.
To ensure data security, privacy, extended and still efficient access control in cloud storage environment is challenging, because of different security requirements of various cloud application. Different types of cloud applications are classified on the basis of private, public, hybrid cloud deployment models but here we are going to classifying cloud applications based on the risk involved in it.

Categorization Of Cloud Applications

- High Critical Application (HCA)
  In this category, applications are extremely bound to timing constraints, accuracy of system decides the system's outcome whether it is successful or failure.
  Basically these are real time systems that need to fetch the data with high accuracy.
  IRCTC's Tatkal reservation system is an example of high critical cloud application. In this scenario while doing payment of ticket, if the entered banking details are not submitted to the irctc server within the span of few seconds then ticket wouldn't book successfully. And hence system will considered as failure for that specific transaction.
- Medium Critical Application (MCA)
  In case of MCA, the failure of the system have less impact. To consider this category we can discuss the example of Drop Box for cloud storage. Drop Box had been failed number of times but still it doesn't have much impact, billions of people share the data in drop box. Also the Amazon web service outage happened in 2008 is other example of medium critical application of cloud. Simple cloud application that does not affected by the outages as the cloud has backup in multiple datacenters.
- Low Critical Application (LCA)
  These applications have no impact of failure on the system. Example of this cloud application includes Zomato, Call Fire etc. which does not need the encryption for this type of application. Although these application have no impact of failure, they needed to be tested in terms of program code, intended application should not communicate with other application or read the user data. The proposed work is intended for category of medium critical cloud application to ensure data security and to extend access control in the cloud storage efficiently [6].

Now in case of accessing the stored data in the cloud, access permissions are set by the data uploader at the time of ciphertext creation, once the ciphertext is generated with access permission associated with it, data will be stored to the cloud. Now suppose if new user enters in the system who needs to access the ciphertext, but user is unauthorized by an original access permissions, as he/she had not the part of the system at a time of document upload. To motivate this scenario, consider the example –
Alice uploads a copy of the data to the cloud securely, with an access policy which states that software developers from company X can decrypt its contents. Bob and Charlie, who are software developers in company X, can hence decrypt the data. Suppose now there is a regulation that any code that is written in the company must be verifiable, therefore accessible by the new group "senior software engineers" in the company. Suppose, Dexter who is one of the member of the senior software engineers group in the company should be able to access and decrypt the encrypted contents. Hence, there is a need for Bob to "extend" the access permission to allow Dexter to access the data. This scenario is depicted in Fig. 1.

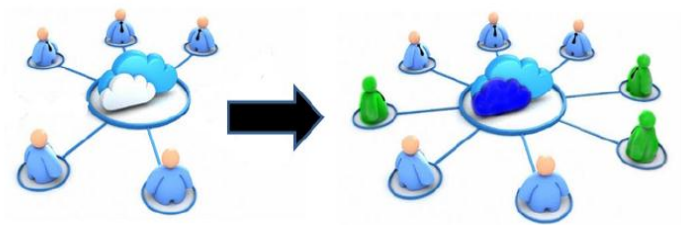

Fig. 1: Collaboration Scenario.

To solve above problem following are possible solutions, first is, Bob asks Alice to re-upload the encrypted plaintext, which can be accessed by the original access policy or the added access policy. This solution only allows Alice to extend the policy and it also requires Alice to verify that Bob fulfils her access policy. The extension cannot be done if Alice is unavailable. Another solution is asking Bob to download the ciphertext, decrypt it, and then re-upload it with the added policy that the decryption can be conducted successfully by regulators from company X. This solution seems work although it is not efficient. However, we note that downloading the ciphertext, decrypt it and reupload it with new access permissions once again, consumes more processing time and server space unnecessarily.
The proposed work aims combinedly for enhancing data security and extendable access control in the cloud environment. To enhance the security of outsourced document encrypted using symmetric encryption, we introduced time varying encryption scheme to avoid the possibility of hacking the data to be repeatedly generated in block cipher encryption. And for managing access

permission, we have split generated data body and data prefix part, which has proposed in existing system [4], we separate data prefix that contains access policy and stored it in metadata file, which will be interlinked with encrypted document. So that whenever the access permission needs to be modify, we have to deal with metadata file only, hence the processing time for extending access policy or permission will be reduced, there is no need to download ciphertext everytime.

Rest of the paper is organized as follows: Section II discusses related work done in ABE and access control schemes. Section III presents the Proposed Work including flow diagram for it. Section IV focuses on the evaluation done by implementing proposed work and finally we will conclude the proposed system work in Section V.

II. RELATED WORK

In this section, we review some existing work in the literature that is closely related to our proposed work.

Attribute Based Encryption

The notion of ABE was first proposed by Amit Sahai and Brent Waters [5], In ABE, a user will wish to encrypt a document to other users that have a certain set of attributes. For example, in a computer science department, the chairperson might want to encrypt a document to all of its systems faculty on a hiring committee. In this scenario it would encrypt to the identity {*"hiring-committee","faculty","systems"*}. Any user who has an identity that contains all of these attributes could have access to the document. The main advantage of ABE is that user could simply store his data on untrusted server, there is no need to rely on trusted server to perform authentication process before delivering the decrypted data. After innovation of ABE, two variants were introduced named as Key Policy - ABE and Ciphertext Policy - ABE by Vipul Goyal, Omkant Pandey et.al [7], In KP-ABE, access policies are associated with the key and ciphertext is generated with set of attributes, so that users who has valid key will get the ciphertext. The decisional bilinear DiffieHellman assumption was used for the technique. Their scheme supports for fine grained access control (allows providing different access rights to the set of users and specifying the access rights of individual user) by storing data on the server in encrypted form while different users are still able to decrypt data as specified by the security policy. The drawback of kp-abe scheme is that the encrypted data cannot choose who can decrypt the file shared. Furthermore to intend this work for dealing with the complex scenario of attribute management by multiple authorities M. Li et al. [8] proposed a multiauthority KP-ABE data access control scheme for securing personal health records in public cloud storage.

In contrast to the KP-ABE, CP-ABE has the ciphertext enclosed with access policy whereas, the key generated by authority is associated with set of attributes. The decryption is successful if attribute set follows the access policy specified in the ciphertext. The CP-ABE is more appropriate in terms of access control for cloud environment as it facilitates the data uploader or encryptor to select the access policy to decide who is valid for downloading the ciphertext. Lots of research has been done in ABE related to their further classification such as HABE, Distributed-ABE etc. but that is out of the scope of proposed work hence we mainly focused on ABE and their main variants.

Data Security and Access Control

Issa M. Khalil et.al. [9] introduces the various security techniques and data storage for cloud. They have identified vulnerabilities associated with the clouds and classified security threats and attacks, presenting state of the art tried to control vulnerabilities, calibrate the attacks, and proposed cloud security framework presenting lines of Defense identifying the dependency levels and identified various cloud security threats. Different cloud security issues like malicious insiders, multi tenancy, side channels, weak browser security, mobility etc. are analyzed thereby classifying as categories of security standards, network, access, cloud infrastructure and data.

The data is stored on shared cloud it's privacy should be protected for an effective data utilization service. issues related with data privacy in searchable symmetric encryption (SSE) addressed by P. Shanmuga Priya et.al.[10] they proposed that server-side ranking based on order-preserving encryption (OPE) causes leakage in privacy which is resolved by two round searchable encryption, supporting top-k multi keyword retrieval. TRSE is employed with vector space model and fully homomorphic encryption (FHE).

The FHE is an encryption system providing arbitrarily complex computation on encrypted data.

Cong Wang, Ning Cao et.al[11], with the help of traditional techniques of searchable symmetric encryption users are able to securely search the encrypted data with the help of keywords, but this process is insufficient for data utilization in huge number of data files in cloud because the approaches provide support to Boolean search only. Their work contributed to outsourcing the data that needs to focus upon cloud data security by encrypting it before storage into public cloud.

Prithi D.and Priya J. et.al. Proposed a technique for the encrypted cloud data using keyword based search and retrieval [12]. With the help of cloud computing an application can be executed on multiple machines simultaneously. They explained Searchable symmetric encryption (SSE) which helps to retrieve encrypted data over cloud and for preventing the leakage two round searchable encryption (TRSE) scheme is visualized. TRSE supports top-k multi keyword retrieval and generate a log file module. This enhances the efficient retrieval of data providing search accuracy. Though security issues have been considered to address, we can see that from existing work, most of the data

security providing and retrieval systems uses symmetric encryption, the problem of symmetric encryption that had been discussed in introduction section is yet to solve.

The issue of repeated plaintext in symmetric encryption is first addressed by Yan Wang and Nihong Wang [13]. They proposed a working model for symmetric encryption, in that key generation is manipulated in number of rounds so that respective ciphertext will be generate differently after the repetitive plaintext enters.

Access Control technique suggested by Kapil Raghuvanshi et.al [14], includes the user attributes for key generation and for encryption, that are changes with time, such as current date and time combinedly forms whatever the string is used as the key for both encryption and decryption purpose. Encryption is quite secure but in case of decryption, key needs to be stored separately in the database in order to preform decryption, and if key is leaked by any insider the data will be compromised at much more cost.

Extendable Access Control System with Integrity Protection (EACSIP) [4], is proposed by W. Susilo, Goumin Yang et.al, proposed, to enhance the collaboration in cloud storage environment. To increase the productivity of cloud, collaboration with more users is necessary, authors achieve this concept by introducing new method for checking integrity of user known as FKE-ET (functional key encapsulation with equality test). In this system, the creation of ciphertext is done using symmetric key algorithm (AES), generated ciphertexts then divided into two parts first is data body and another is prefix for further process. Access policies of key will be added to the prefix along with data body, if the FKEET runs successfully then only, the user will be considered as authorized and thereby user can extend to other user, Any user who acquire the plain data would be able to extend access policy. the existing work on enabling access policy modification (or merely extending the access policy) fail to provide integrity check and hence, cannot be used in the extended access control scenario. In this system the file encryption attributes are attached in the file itself that are to be regularly modified when permissions are updated from one user to another. Every time when user, having write permission, forwards a file to another with new permission new attributes are added to the file description. In this process file is regularly decrypted and then again encrypted with new details. This increases time required for managing file in terms of encryption and decryption. This also adds to more consumption of space as unnecessary replicas are created of the same file.

III. PROPOSED WORK

The proposed work implemented here focusing mainly on the basic requirements of cloud system i.e. data security, confidentiality, integrity and authenticity.

Following entities will be participating in the proposed system.

1. The cloud server – In collaboration with the cloud admin, new user registration with user id and password creation will be perform. Cloud server will be responsible for data storage, allows data uploader to encrypt and share his/her data. Also prepares user attributes for key generation such as server date and time of data or document to be upload.
2. Key Management – Performs mathematical calculations over the user attributes to make key more complex. The intuition behind key generation by user attributes for encryption is, to make ciphertext more complex to analyse even if there is any repeated plaintext. Hence it will generates such a key for encryption purpose, so that generated ciphertext will be different, every time. Key generation combinedly done by the data uploader and cloud server, since the date and time we will be using is network or server provided.
3. The data uploader – The one who owns the data and wanted to provide security to store it to the cloud. The plaintext is encrypted using proposed scheme i.e. TVES with generated key. the encrypted data can only be decrypted by those users who have valid key.
4. The User – In order to accessing the data, the user should register into the system and should posses the access permission to acquire the ciphertext. Decryption key will be generated for valid user if he/she does hold valid attributes for accessing the ciphertext.
5. Access Permission Management – Access permissions will be separately stored in the metadata file using which the user who has write permission of data or document will be able to forward the access permission to others in the system. And this will be interlinked with ciphertext, with user id and document id.

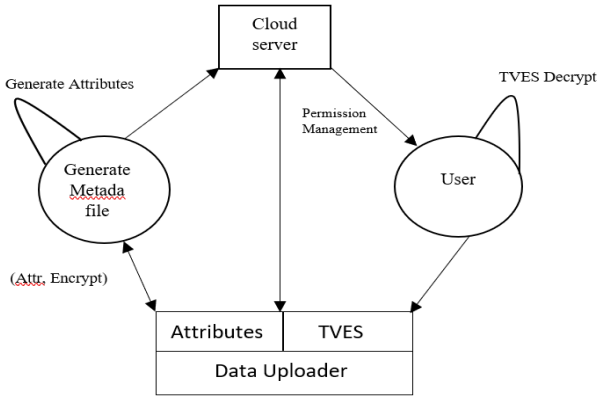

Fig. 2 : Working Flow of Proposed Scheme

- **TVES Encryption**

TVES consist of two phases for encryption purpose, first is Key Generation and another is actual encryption with generated key.

Notations Used in TVES -

k - Secrete key
rnd - Unique Random Number
dd - Date of month
m - Month of year
yy - current year
date - Composition of d,m,y, of data to be upload
t - Upload Time of document.

r

Key Generation (*Rnd,t,dd,m,yy*)

I. Obtain a random unique number Rnd.
II. Calculate date = dd+m+yy
III. Calculate t = hh+mm // hours and minute
IV. Now, finally the secrete key k as,

$$k = Rnd + (date - t)$$

After key generation, generated secrete key will be given as input to the encryption algorithm, the further process is describe below.

Algorithm 1 : TVES encryption

```
1.  Procedure TVES (k)
2.  Select document to Upload
3.  Allot document id
4.  Read document into bytes[ ]
5.  Fetch the key k
6.  for j=1 to n        // n=bytes.len()
7.      ki = concat(k,i)
8.      for i=0 to byte[].n
9.      long data=(long)byte[i]
10.     EncByte[i] =data + ki
11.     i++;
12. End For
13. End For
14. Store encrypted document on the server using encrypted byte array
```

First data is converted into bytes. To make the secrete key more secure we add '1 to n' of numbers to the end of each key, where n is the length of byte array , for example if the generated key is 301824 then we concatenate 1 to the end of key, so the key will becomes 3018241 and it will increase accordingly up to the length of bytes array, which we have converted from document firstly. Since the data generated during processing becomes large we have converted it to the 'long' from 'integer' datatype. As we are dealing with data security, to make data more secure, we combine the newly generated secrete key with each byte of array, So if the data will compromise in future no one will be able trace the actual data and information loss will be limited. Finally store the encrypted data in byte array and store document on the server with its concerned metadata file. The entry will be maintained in database, and database performs log monitoring which maintains all the records of users.

- **TVES Decryption**

Decryption Algorithm takes ciphertext an input and returns its plaintext.

Algorithm 2 : TVES decryption

```
1.  Get Encrypted document from server
2.  Read document into bytes[]
3.  Get secrete key k from database
4.  For j = 0 to bytes.len
5.  Generate ki = k + ( Date – Time)
6.  DecByte[i]=Decrypt(bytes[i],ki)
7.  Convert str into byte[]
8.    Initialize counter cnt=1;
9.              For i = 0 to byte[].len
10.             k=concat(ki,cnt)
11.             long data=(int)byte[i]
12.             data =data – ki
13.             i++
14.   End For
15. Convert DecByte[] to file
16. Deliver decrypted file to user
```

When the file is stored in the cloud storage, it is enclosed with its metadata file that specifies access permission for encrypted document. If user doesn't hold valid key then decryption will not be performed.

Access Permission Management - At the time of file upload, System will automatically create the metadata file containing all the access permissions and document attributes to generate document encryption key, hence encrypted document will be interlinked . The metadata file is an xml file. Every document will have one metadata file. The

metadata file will be divided into two sections as Document Attribute and Access Permission attributes.
Document Attribute contains Doc id, Metadata file id, Upload date, Upload time and Access Permission Attributes Access permission attributes will vary depending on the access permission details of the document, they are read and write.

IV. RESULT AND DISCUSSION

We have implemented the proposed work to address the repeated plaintext problem in symmetric encryption. Since most of the ABE system uses symmetric encryption, hence this can improve the existing performance of ABE in terms of security. The system is implemented in JAVA, The interfaces of system designed with AJAX, HTML, Javascript and the web services are hosted in Apache Tomcat. The cloud uses MySQL database which can be replaced by other database for server side data storage. results are analyzed on windows environment with Intel 2.40GHz processor, 4GB RAM.
We evaluate the proposed work with Extended Access Control with integrity protection [4], in which file encryption attributes are attached in the file itself that are to be regularly modified when permissions are updated from one user to another. Every time when user, having write permission, forwards a file to another with new permission new attributes are added to the file description. In this process file is regularly decrypted and then again encrypted with new details. This increases time required for managing file in terms of encryption and decryption as well as access permission modification. Hence the values calculated for existing system as –
Modification time = (attributes modification) + (document decryption) + (document re-encryption with new access permission & upload)
In *proposed system* we have separated the access permissions and file encryption attributes from the file. We only have used these file attributes for key generation and file encryption. Access permissions data is stored separately in an xml file along with file details, in encrypted form. Now in this case whenever there is any modification in file access permission attributes only xml file is to be decrypted and accessed. That directly subtracts file re-encryption time and also there are no replicas created of the file on server that saves the storage space on server. User doesn't ever have to deal with direct decryption of the file. In case of permission modification or document forward only xml file is updated every time.
Modification time = (attributes modification) + (xml file access & encryption)
Screenshots shows the permission forward time for image, doc and pdf files. Evaluation report shows size for each file in kb and time in millisecond.

Evaluation Report

Document Title	Document Size	Existing System	Proposed System
Branch Home Image	100 KB	142 MS	6 MS
flex Image	291 KB	421 MS	51 MS
Class Graphics	2585 KB	1493 MS	20 MS
Apointment Letter	2585 KB	1559 MS	1 MS
Database Queries	12 KB	906 MS	60 MS

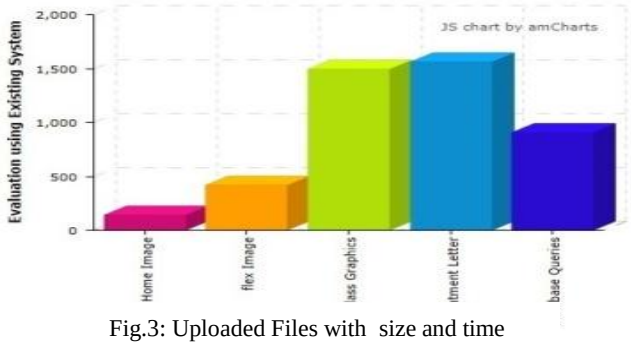

Fig.3: Uploaded Files with size and time

Fig.4: Permission Modification Time required in existing system

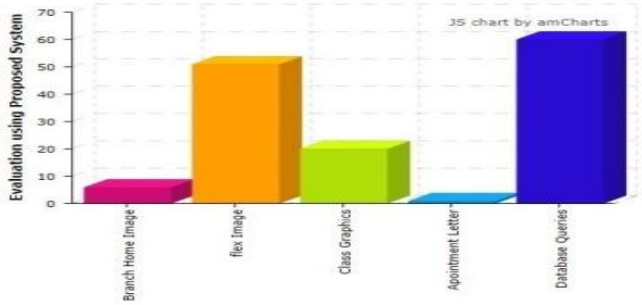

Fig.5: Permission Modification Time required in proposed system

The results are analyse with image, text and doc format files and we have observed that time required in proposed system linearly increases with number of times the permission being forwarded to

new user. And also with proposed system images are require more time for processing as compared to text or doc files.

V. CONCLUSION

The implemented work is aims to provide enhanced security to outsourced document and efficient extendable access control for medium critical cloud applications. It is intended for medium critical cloud applications, proposed scheme for key generation and encryption can be used in any security intelligence system and will be more beneficial in terms of providing extra security in symmetric encryption. Results are compared in terms of access permission forward time required in existing work and proposed scheme of permission management, and clearly we can analyse that time difference in both system is very high. Thereby proposed work is more efficient than existing system. In future the system should be tested on larger file sizes, proposed system currently supports images, pdf, doc 97-2003 file formats, further work can be extended for encrypt the files in docx, xls and video format.

REFERENCES

[1] Hamlen, K., Kantarcioglu, M., Khan, L., Thuraisingham, B.,"Security issues for cloud computing", International Journal of Information Security and Privacy, 2010

[2] Chen, Y., Paxson, V., Katz, R. ,"What's new about cloud computing security", Tech. Rep. UCB/EECS-2010-5, Electrical Engineering and Computer Sciences, University of California at Berkeley 2010.

[3] Kandukuri, B., Paturi V, R., Rakshit, ,"A.: Cloud security issues", *In: Proceedings of the IEEE International Conference on Services Computing, SCC'09, pp. 517–520. IEEE Computer Society, Washington*, DC, USA DOI 10.1109/SCC.2009.84,2009.

[4] Willy Susilo, Peng Jiang, Fuchun Guo, Guomin Yang, , Yong Yu and Yi Mu, Extendable Access Control System with IntegrityProtection,IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY DOI 10.1109/TIFS.2017.2737960.

[5] Amit Sahai and Brent Waters, "Fuzzy identity-based encryption," in EUROCRYPT 2005, ser. LNCS, vol. 3494, 2005, pp. 457–473.

[6] Saravana Kumar, Rajya Lakshmi, Balamurugun. Enhanced Attribute Based Encryption For Cloud Computing , International Conference on Information and Communacation Technologies (ICICT 2014)

[7] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine- grained access control of encrypted data," in Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, 2006, pp. 89–98.

[8] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute based encryption," IEEE Trans. Parallel Distrib. Syst., vol. 24, no. 1,pp. 131–143, 2013.

[9] Issa M. Khalil, Abdallah Khreishah and Muhammad Azeem "Cloud Computing Security: A Survey", Computers, 2014

[10] P. Shanmuga Priya and R. Sugumar, "MultiKeyword Searching echniques over Encrypted Cloud Data", IJSR,2014.

[11] Cong Wang, Ning Cao, Kui Ren and Wenjing Lou "Enabling Secure and Efficient Ranked Keyword Search over Outsourced Cloud Data", *IEEE Transactions on Parallel and Distributed Systems*, 2012.jk

[12] Preethi.D, Priya.J and shanthini.B, "Retrieval of Encrypted Data Using Multi Keyword Top –K Algorithm", *International Journal of Scientific and Research Publications*, 2014.

[13] Yan Wang and Nihong Wang , "Research on Time-Varying Camellia Encryption Algorithm", https://link.springer.com/chapter/10.1007/978-3-642-29455-6_110.

[14] Kapil Dev Raghuwanshi, Prof. Sitendra Tamrakar, An effective Access from Cloud Data using AttributeBased Encryption,2015 1st International conference on futuristic trend in computational analysis and knowledge management (ABLAZE 2015).

Authors Profile

Dr. S.Y. Amdani pursed B.E., M.E., and Ph.D. in Computer Science & Engineering from SGBAU, Amravati, in 1989, 2008 & 2014. He is currently working as Professor in Department of Computer Science & Engineering, Babasaheb Naik College of Engineering Pusad. since 1992. He is a Life member of ISTE. He had reviewed many papers of IEEE conferences and reputed journal. He is registered supervisor in CSE board of SGBAU, Amravati University and had supervised 4 Ph.D students. He has published more than 40 research papers in reputed international journals including IEEE and it's also available online. His main research work focuses on Real Time Systems, Networking, and Database etc. He has 26 years of teaching experience and 8 years of research experience.

Dr. S. A. Bhura pursed B.E., M.E., and Ph.D. in Computer Science & Engineering from SGBAU, Amravati, in 2001, 2010 & 2018. He is currently working as Assistant Professor in Department of Information Technology, Babasaheb Naik College of Engineering Pusad. since 2003. He is a Life member of ISTE. He had reviewed many papers of IEEE conferences and reputed journal. He has published more than 25 research papers in reputed international journals including IEEE, ACM, Springer and it's also available online. His main research work focuses on Real Time Database Systems, Computer Architecture, and System Software etc. He has 15 years of teaching experience and 4 years of research experience.

Ms. A. D. Yevatkar pursed B.E in Computer Science & Engineering from SGBAU, Amravati, in 2015 and is PG student at Babasaheb Naik College of Engineering Pusad. currently pursuing ME in CSE from SGBAU, Amravati. Her main areas of Interest are cryptography and cloud computing.

