International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 09 | Sep -2016 www.irjet.net p-ISSN: 2395-0072

Homomorphic Encryption Approach For Cloud Data Security

Suraj Sheshrao Gaikwad[1], Amar R. Buchade[2]

[1]Department of Computer Engineering, Pune Institute of Computer Technology, Savitribai Phule Pune University, Pune, India. Email: surajgaikwad2410@gmail.com
[2] Department of Computer Engineering, Pune Institute of Computer Technology, Savitribai Phule Pune University, Pune, India. Email: amar.buchade@gmail.com

---

Abstract - *Cloud computing is technique which has become today's hottest research area due to its ability to reduce the costs associated with computing. In today's internet world, it is most interesting and enticing technology which is offering the services to various users on demand over network from different location and using range of devices. Since cloud computing stores the data and propagate resources in the open environment, security has become the main obstacle which is hampering the deployment of cloud environments. So to ensure the security of data, we proposed a method which uses multilayer security for securing data using Homomorphic Encryption in cloud computing. This system helps to secure data for various cloud users.*

Key Words: Cloud Computing, Privacy Preservation, Homomorphic Encryption.

1. INTRODUCTION

Need of cloud to manipulate and manage data is increasing rapidly for sharing resources [4, 18]. It is financially beneficial to store data with a third party, the cloud provider. However, storing data on third party infrastructure poses risks of data disclosure during retrieval. Therefore, the data is stored in encrypted form [9]. Encryption alone is not sufficient, as it provides security but reduces usability. Major advantage to be drawn from cloud computing is due to delegation of computation, but encrypting data would require sharing of keys with the third party performing computation on it, thereby increasing vulnerability. Hence, there is a need of feasible homomorphic schemes that allow user to compute on encrypted data, to verify a computation done by third party, to search an encrypted database, and so on [16, 18]. Fully Homomorphic Encryption allows a third party to evaluate arbitrary functions over encrypted data without decrypting it [1, 2].

Cloud provides large shared resources where users can enjoy the facility of storing data or executing applications. instead of gaining benefit of large resources, storing critical data in cloud is not secured [3, 4]. Hence, cloud security is one of the important topic to make cloud useful at the enterprise level. Data encryption is a primary solution for providing discreetness to sensitive data. However, processing of encrypted data requires extra overhead, since repeated encryption-decryption need to be performed for every simple processing on encrypted data [18]. Hence, direct processing on encrypted cloud data is advantageous, which is possible due to the use of technique called Homomorphic encryption schemes [4, 5]. Homomorphic Encryption provides a method to perform operations directly on encrypted data.

1.1 Data Storage

One of the major concern while storing data at third party data center is that user is unaware about the location of data where exactly it is stored [13]. Users rely on services that are non-transparent to them, and no information about the servers operation is broadcast. Although this can improve security by insignificance, it also undermines user trust [15, 18]. How the security of data is ensure at the server side might not be clear to clients. Data retention is also a concern for users. The cloud service provider can be able to keep deleted data in backups or for some unpublished reason. For example, Facebook is able to kept deleted data but removed it from view.

1.2 Access Control

Most of the cloud systems include basic access control [10, 11]. Almost every system has privileged users, such as system administrators who is able to access user data and all application over the machine. When data or processes are outsourced to the third party data centers, possibly sensitive data or processes are handed over for safe keeping [13, 18]. In a local setting, user know to whom they trust for their data, but in a cloud environment users almost never know the position of the cloud server, the people managing it at the server side, and who has access to it.

1.3 Identity Protection

Data transmitted over the internet provides valuable information about people. Search keywords, credit card usage, and mobility patterns are some examples of information that can be used to identify and track individuals from apparently unsigned data, attackers can easily gain this information [10]. This same data is available practically without restrictions to cloud service providers. For example, some cloud service providers business models

include targeted advertising that is based on monitoring account traffic or data stored on user accounts [13, 18].

1.4 Privacy Preservation

One of the problem in public clouds is how to share documents based on fine-grained attribute-based access control policies (acps) [7, 10]. Solution is to encrypt documents which satisfy different policies with different keys using a public key cryptosystem such as attribute-based encryption, and/or proxy re-encryption [18].

2. RELATED WORK

To provide security to the data is always having a importance issue because of the critical nature of the cloud and very large amount of complicated data it carries, the need is even important. Therefore, data security and privacy issues that need to be solved have they are acting as a major obstacle in adopting cloud computing services.
The main security issues of cloud are:

2.1 Privacy and Confidentiality

Once user outsources data over the cloud there should be having some guarantee that data is accessible only to the authorized user and unauthorized access are prevented as well as data is protected from cloud service provider also [10, 12]. The cloud user should be confident that data stored on the cloud will be confidential [18].

2.2 Security and Data Integrity

Data security can be provided using different encryption and decryption techniques. With providing security to data, cloud service provider should also implement technique to monitor integrity of data on the cloud [18].

2.3 Basics about Encryption

Encryption schemes are, first and foremost, designed to preserve confidentiality. In cryptography, encryption is the art of changing or converting messages or information in such a way that only legitimate user can access and read the original message [6, 18]. Encryption itself is not enough prevent interference with data, but denies the message content to the interceptor.

2.4 Symmetric Encryption Scheme

In symmetric crypto system encryption as well as decryption both can be performed with the single key. Hence, the sender and the receiver have to use same key before performing any private communication. Therefore, it is not possible for two different people who never met each other to use such schemes directly [6, 18].

2.5 Asymmetric Encryption Scheme

In relation with previous scheme, asymmetric schemes introduce a fundamental difference between the abilities to encrypt and to decrypt. The encryption key is public, as the decryption key remains private. When Bob wants to send an encrypted message to Alice, she uses her public key to encrypt the message. Alice will then use her private key to decrypt it. Such schemes are more functional than symmetric ones since there is no need for the sender and the receiver to agree on anything before the transaction [6, 18].

2.6 Data Security with Homomorphic Encryption

Homomorphic encryption is technique to perform operation on encrypted data which is also known as ciphertext, which can generate an encrypted result, which, when decrypted, produce the result which is exactly same as like operations are performed on plaintext data. This can be a major advantage for applications that outsource encrypted data to the cloud [15, 18].
One Such example of concatenation of two string using homomorphic encryption is as shown in following figure [1], this show the owner encrypt two different strings and upload it to data collection server (DCS) and upon user request DCS perform the string concatenation, here operations are performed on encrypted data so, no information disclosure and result will be same as like performing operation on plaintext data.

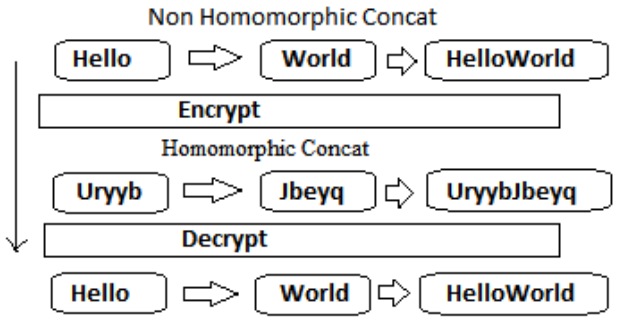

Fig -1: Homomorphic concatenation of two strings.

Homomorphic encryption technique is striking for many applications, but it has some serious limitation: the homomorphic property is typically restricted to only one operation, which is usually addition or multiplication [14]. Methods which have the homomorphic property for both addition and multiplication operation concurrently lead us a step closer to real-life applications.

Ronald Rivest and his associate introduced the concept of fully homomorphic encryption which they called as privacy homomorphism in 1978, but in 2009 Craig Gentry proposed a first fully homomorphic encryption (FHE) scheme. Gentrys scheme allowed random number of additions and multiplications on encrypted data while providing the

assurance that the outcomes were precisely indicate in the decrypted data [18].

3. PROPOSED FRAMEWORK

After the survey it is found that even if data owner encrypt the data before uploading to the cloud in order to ensure data confidentiality from that of the cloud. Data uploaded is not safe at cloud because once data owner upload the data item, it drops the command over the data and cloud service provider becomes the ultimate data handler. Hence to prevent from various attacks at cloud which includes insider attack also it very important to encrypt the data item before uploading.

Now once owner encrypt the data owner's data is secure but what if user want to perform certain operation over that data, traditionally when user request to perform certain operation, data owner needs to download the data perform the user requested operation on the data, again encrypt the data and upload it to the cloud, but this put computational overhead on data owner and computational facility of cloud remains underutilized.

So to take the advantage of computation facility owner should have some mechanism that allow CSP to perform operation of encrypted data without data owners intervention. And this is possible with homomorphic encryption technique. Hence using these technique we proposed a model for securing data at third party that cloud service provider. Under the proposed framework we have four entity and that are Data Owner, Cloud, User and Identity Provider as shown in following figure [2].

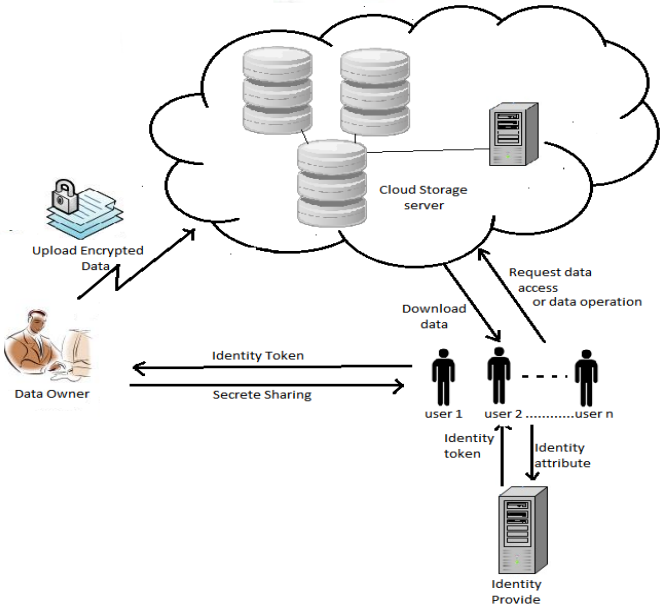

Fig -2: Proposed Framework.

- Data owner perform Homomorphic encryption on data and upload it to the cloud.
- User can send data access request or operation request to the CSP.
- CSP perform user requested operation and send the encrypted data.
- To decrypt the data user need secrete key that user can access from data owner through secure communication channel like SSL. To obtain the secrete key user need to have identity token that it can get from identity provider which is trusted third party which takes user attribute and return identity token.

4. METHODOLOGY

1)Owner: Perform KeyGen function of homomorphic encryption to obtain (Pk, Sk)
KeyGen: Downer→ KeyGen (Pk, Sk)
encrypt the data and upload it to data collection server with public information as,
Enc: *ForAll* di(Downer(E(Sk(di))) → DCS)
hence, Ct = E(Sk(di))
The upload function is used to upload the encrypted data to the cloud as,
fs (upload()): *ForAll* Cti(Downer(Cti)) →DCS

2) Cloud: Evaluate Function (Eval) allow DCS to perform operations on encrypted data when user request by performOP function receive. Server has a function *f* for doing evaluation of cipher text Ct and performed this as per the required function using Pt.
EvalPk (*f*, ESk(a), ESk(b))
EvalPk (f, Ct1, Ct2)

3) User: Request to perform operation on data eg. To concatenate two strings and download the encrypted data from DCS, request Data owner a secret key to decrypt it. Secret sharing requires identity token that should be taken from identity provider which is trusted third party.
fs(performOP()): Eval(f, Cti, Ctj)
fs(download()):user →DCS(ESk(Cti, Ctj))
fs(keyreq()): keyshare=keyreq→Downer
hence, keyshare=Sk ←Downer
fs(download()):
Result= Dec(Sk(EvalPk(f, ESk(a), ESk(b))))
hence, Result=Dec(Sk(EvalPk(f, Ct1, Ct2)))
PT=Dec(Sk(Pk, Ctij))

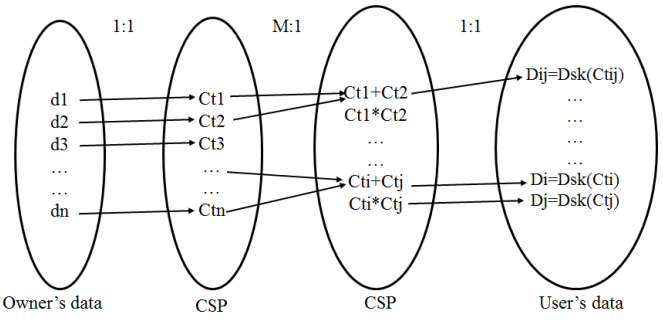

Fig -3: Set Mapping between Data Owner, DCS and User.

Figure [3] shows the mapping between data owner and Cloud Service Provider (CSP) and CSP to data user. Following figure [4] explain the detain flow of homomorphic encryption function.

Properties of Homomorphic Encryption

An encryption is called as homomorphic, if [14]: from Enc (a) and Enc(b) it is possible to compute Enc(f(a, b)), where f can be: +, ×, ⊕, ⊗ and without using the private key. Among

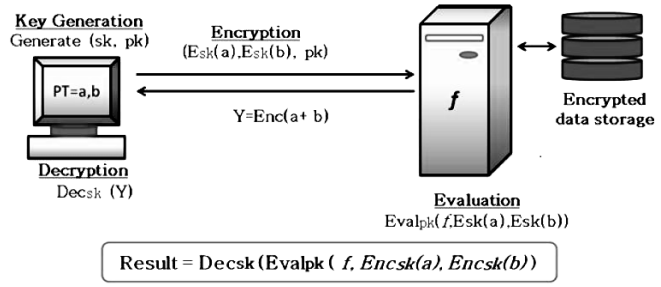

Fig -4: Homomorphic Encryption Function

the Homomorphic encryption we distinguish, according to the operations that allows to assess on raw data, the additive Homomorphic encryption (only additions of the raw data) is the Paillier and Goldwasser Micalli cryptosystems, and the multiplicative Homomorphic encryption (only products on raw data) is the RSA and El Gamal cryptosystems.
There are two main properties of Homomorphic Encryption:

- Additive Homomorphic Encryption: A Homomorphic encryption is additive, if [14]:
  Ek (Pt1 ⊕ Pt2) = Ek(Pt1) ⊕ Ek(Pt2) where Pt1 and Pt2 are plaintext messages, ⊕ denotes the additive homomorphic operation.
- Multiplicative Homomorphic Encryption: Homomorphic encryption is multiplicative, if:
  Ek (Pt1⊗ Pt2) = Ek(Pt1) ⊗ Ek(Pt2) where Pt1 and Pt2 are plaintext messages, ⊗ denotes the multiplicative homomorphic operation.

The encryption function is semantically secure this tells the fact that an opponent/data hacker cannot get any auxiliary information about the plaintext from corresponding ciphertexts and given public key [15].

5. DATASETS AND IMPLEMENTATION

5.1 Dataset

This system requires the dataset which consist numerical data and text data. As system Perform three homomorphic operations. Additive and multiplicative homomorphic encryption is perform on encrypted numerical data. Concatenation of encrypted strings is perform on encrypted text data. Numerical data is stored in Numerical Data. text file and text data is TextData.txt files, both contains encrypted data and both are stored on cloud in appropriate owner directory.
Sample text dataset as shown in following table 1 consist plaintext data which is normal data that any one can access and read, encrypted data which is different from plaintext.
Sample numerical dataset is as shown in table 2 consist plain numerical data and encrypted numerical data.

Table -1: Sample Text Dataset

Sr.No	Plaintext Data	Encrypted Data
1	Hello	w6 == @
2	World	(@C = 5
3	GoodMorning	v@@5j@C? :?8
4	GoodAfternoon	v@@5p7E6C?@@?
5	PICT	!xr%
6	Pune	!F?6

Table -2: Sample Numerical Dataset

Sr.No	Plaintext Data	Encrypted Data
1	123456	462534621953675707612819276243252737734824373410338535850949183567287269185 7
2	654321	39799493075552536865210515029007805581836350788959790231041605019004031462 31
3	777777	14698713641075435598803108954280455335624681772077995652308343514476852797 43
4	10000	20090303524261705707839709461571302355494562482327523177011034929412323699 0

5.2 Homomorphic Addition using Paillier Cryptosystem

1) Key Generation

- Choose two large prime numbers p and q randomly and independently of each other such that gcd (pq, (p-1)*(q-1)) = 1.
- Compute n = p*q and lambda = lcm ((p-1), (q-1)).
- Select random integer g in Zn^2 where gcd (L (g^lambda mod n^2), n) = 1. with μ =(L(g^lambda mod n^2)) mod inverse n, where function L (μ)=μ -1/n.
  Hence,
  The public (encryption) key is (n, g).
  The private (decryption) key is (lambda, μ).

2) Encryption

- Let Pt be a plaintext to be encrypted, where Pt ∈ Zn.
- Select random r, where r ∈ Zn.
- Compute ciphertext as: C = g^m.r^n mod n^2.

3) Decryption

- Let C be the ciphertext to decrypt, where C ∈ Zn^2.
- Compute plaintext Pt = L(C^lambda mod n2).μ mod n.

5.3 Homomorphic Multiplication RSA Cryptosystem

1) Key Generation

- Select two prime numbers p and q.
- Compute n=p*q.
- Compute ϕ (n) = (p-1)* (q-1).
- Compute e such that, gcd (ϕ (n), e) =1.
- Compute d such that, d.e mod ϕ (n) =1.

Hence,
The public key is (e, n)
The private key is (d, n)

2) Encryption

- Encryption of plaintext is done as, Ct = (Pt)^e mod n. Where, Pt is plaintext and Ct is ciphertext.

3) Decryption

- Decryption of ciphertext is done as, Pt = (Ct)^d mod n. Where, Pt is plaintext and Ct is ciphertext.

5.4 Homomorphic Concatenation using rotate47 Caesar Cipher

Rotate47 Caesar Cipher algorithm is used for string encryption, upload and decryption. Rotate47 uses a larger set of characters from the common character encoding known as ASCII. Specifically, the 7-bit printable characters, excluding space, from decimal 33 '!' through 126 '~', 94 in total, taken in the order of the numerical values of their ASCII codes, are rotated by 47 positions, without special consideration of case.

```
public String rotate (String value)
{
int length = value.length();
StringBuilder result = new StringBuilder();
for (int i = 0; i < length; i++)
{
char c = value.charAt(i);
if (c != ' ') f
c += 47;
if (c > '~')
c -= 94;
}
result.append(c);
}
```

6. RESULTS

1) Operation without homomorphic encryption

While performing operation without homomorphic encryption, upon each user request owner needs to download the data, decrypt it, perform user requested operation on it again encrypt it and upload it to the cloud. This involves repeated encryption and uploads action. Hence it is time consuming process for owner and it is also a computational overhead. Hence, total time required for single operation can be calculated as:

Required time = (Enc_time*2) + (Up_time*2) +
(Exe_time) + (Down_time) + (Dec_time)

Where,
Enc_time = Encryption time
Up_time = Upload time
Exe_time = Execution time
Down_time = Download time
Dec_time = Decryption time

Following figure 5 shows time required to perform operation on normal data without homomorphic encryption.

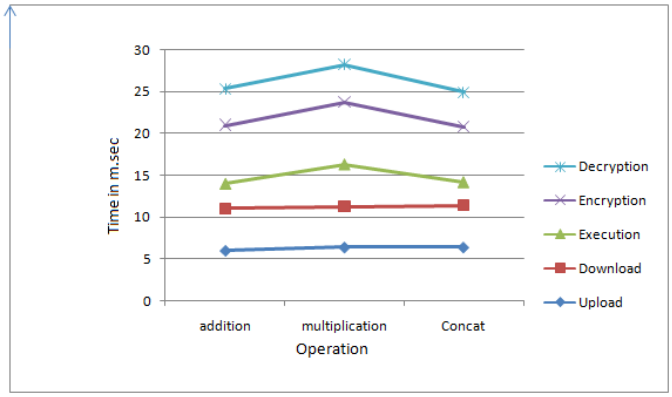

Fig -5: Without Homomorphic Encryption

2) Operation with homomorphic encryption

With homomorphic encryption time required for single operation can be calculated as:

Required time = (Enc_time) + (Up_time) +
(Exe_time) + (Down_time) + (Dec_time)

Following figure 6 shows time required to perform operation on encrypted data using homomorphic encryption. Total times required by each operation in both cases are given in figure 7 as shown below. The unit of measurement for time is millisecond (msec).

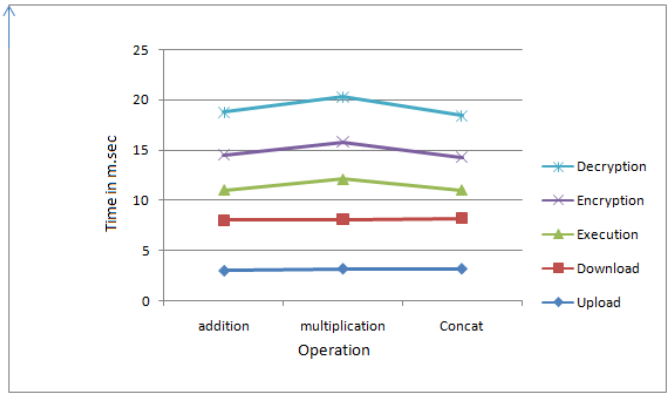

Fig -6: With Homomorphic Encryption

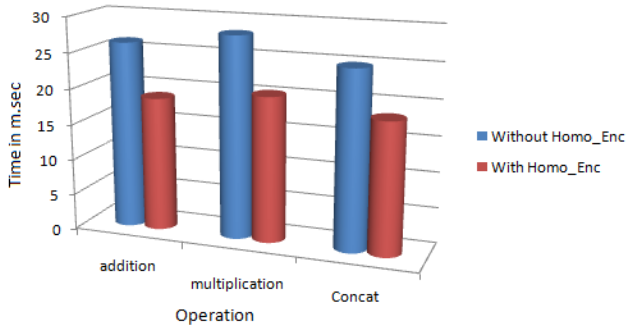

Fig -7: Total time in both cases

7. EXPERIMENTAL SETUP

The experimental setup for the proposed system will consider data collection server (DCS), Data owner, User who will access the information from data collection server and request DCS to perform certain operation on encrypted data.

The particulars about platform and technology are as follows which are used to build proposed system :

Base Operating System: Windows 7 and above, Ubuntu 14.04.
Web Server: Apache-tomcat-7.0.67 Server.
Languages: Java Servlet, JSP, JavaScript, HTML, CSS.
Database: Mysql Distribution 5.5.46 and Mysql workbench development IDE
Browser: Google Chrome, Opera, IE8 & above, Mozilla Firefox Etc.
This system is tested on Amazon EC2 cloud platform with t2.micro instance with 64-bit Microsoft Windows Server, 1 vCPU, High Frequency Intel Xeon Processors with Turbo up to 3.3GHz and 1GB of RAM. This instance is balance of compute, memory, and network resources.

8. CONCLUSION

The concept of cloud computing security root on Homomorphic encryption, which is a new concept of security that allow to provide results of operations on encrypted data without disclosing the plaintext data upon which the operations was carried out, with respect of the data confidentiality. In this paper we show a state of the art on homomorphic encryption schemes discuss its parameters, its additive, multiplicative property and performances also discuss the data security issues in cloud computing.
The application of Homomorphic encryption for data security in Cloud Computing is key concept, more generally, we the user outsource the calculations on confidential data to the Cloud server and keep the secret key that can used to decrypt the result of calculations. So, data and computation security problem is overcome through the application of Homomorphic algorithm. Data security as well as personal information privacy is achieved by this algorithm. Benefits with this approach is enhanced information privacy, enhanced data security, utilization of computational power of server and removing the burden of calculation from data owner.

REFERENCES

[1] Ayantika Chatterjee and Indranil Sengupta, "Translating Algorithms to handle Fully Homomorphic Encrypted Data on the Cloud," IEEE Transactions on Cloud Computing, Volume: pp, Issue: 99, pp: 1, September 2015.
[2] David W. Archer and Kurt Rohloff, "Computing with Data Privacy: Steps toward Realization," IEEE Computer and Reliability Societies, Volume: 13, Issue: 1, pp:22-29, February 2015.
[3] Zahir Tari, Xun Yi, Uthpala S. Premarathne, "Security and Privacy in Cloud Computing: Vision, Trends, and Challenges," IEEE Cloud Computing, Volume: 2, Issue: 2, pp: 30-38, June 2015.
[4] Chao FENG and Yang XIN, "Fast key generation for Gentry-style homomorphic encryption," The Journal of China Universities of Posts and Telecommunications, Volume: 21, Issue: 6, pp: 37-44, December 2014.
[5] Payal V. Parmar and Shraddha B. Padhar, "Survey of Various Homomorphic Encryption algorithms and Schemes," International Journal of Computer Applications, Volume: 91, Issue: 8, pp: 26-32, April 2014.
[6] Shashank Bajpai and Padmija Srivastava, "A Fully Homomorphic Encryption Implementation on Cloud Computing," International Journal of Information & Computation Technology, Volume: 4, Issue: 8, pp: 811-816, December 2014.
[7] Dr C. P. Gupta and Nitesh Aggarwal, "Fully Homomorphic Symmetric Scheme Without Bootstrapping," IEEE International Conference on Cloud Computing and Internet of Things, pp: 14-17, December 2014.
[8] Bharath K. Samanthula and Yousef Elmehdwi, "A secure data sharing and query processing framework

via federation of cloud computing," Information Systems, Volume: 48, pp:196-212, August 2013.

[9] Mohamed Nabeel and Elisa Bertino, "Privacy Preserving Delegated Access Control in Public Clouds," IEEE Transactions on Knowledge and Data Engineering, Volume: 26, Issue: 9, pp: 2268-2280, April 2013.

[10] Mohamed Nabeel and Ning Shang, "Privacy Preserving Policy-Based Content Sharing in Public Clouds," IEEE Transactions on Knowledge and Data Engineering, Volume: 25, Issue: 11, pp: 2602-2614, November 2013.

[11] Jiadi Yu and Peng Lu, "Toward Secure Multikeyword Top-kk Retrieval over Encrypted Cloud Data," IEEE transactions on dependable and secure computing, Volume: 10, Issue: 4, pp: 239-250, August 2013.

[12] Monique Ogburn and Claude Turner, "Homomorphic Encryption," Procedia Computer Science, Volume: 20, pp: 501-509, November 2013.

[13] Jian Li, Danjie Song and Sicong Chen, "A simple fully Homomorphic encryption scheme available in cloud computing," IEEE 2nd International Conference on Cloud Computing and Intelligence Systems, Volume: 1, pp: 214-217, October 2012.

[14] Craig Gentry and Shai Halevi, "Implementing Gentrys Fully-Homomorphic Encryption Scheme," Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp: 129-148, February 2011.

[15] Craig Gentry, "A Fully Homomorphic Encryption Scheme," Ph.D. dissertation, Stanford University, September 2009.

[16] Craig Gentry, "Computing Arbitrary Functions of Encrypted Data," Communications of the ACM, Volume: 53, Issue: 3, pp: 97-105, September 2008.

[17] Murat Kantarcioglu and Wei Jiang, "A Cryptographic Approach to Securely Share and Query Genomic Sequences," IEEE Transactions on information technology in biomedicine, Volume: 12, Issue: 5, pp: 606-617, September 2008.

[18] Suraj S. Gaikwad and Amar R. Buchade, "Survey on Securing Data using Homomorphic Encryption in Cloud Computing," International Journal of Computer Sciences and Engineering, Volume: 4, Issue: 1, pp: 17-21, January 2016.