INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 6.078
(Volume 7, Issue 1)
Available online at: https://www.ijariit.com

Security in cloud computing using honey encryption

Navneet Singh Khurana
navneetsingh9009@gmail.com
SAM College of Engineering and Technology, Bhopal, Madhya Pradesh

ABSTRACT

As more and more data creation is happening in today's age, it is getting very difficult to store such a vast amount of information. The best way to store these vast amounts of data is to store it on the cloud. As people and big organizations are moving towards cloud to store their data, security remains the primary concern. Is the data securing enough on the cloud? One of the ways to provide security in cloud computing is with Honey Encryption. Honey encryption generates ciphertext, which if provided with an incorrect decryption key, produces a believable plaintext. Hence, by providing false plaintext Honey Encryption provides assurance against Brute force attack. In addition, after the data encryption, SRM (Secure Repository Manager) divides the data into chunks of small data and uploads it to cloud servers.

Keywords: *Cloud Computing, Homomorphic Encryption, Elliptic Curve Cryptography, Honey Encryption, Distribution Transforming Encoder, Secure Repository Manager*

1. INTRODUCTION

1.1 What is Cloud Computing?

Cloud Computing is an internet based computing wherein the devices and computers are provided with shared computing resources and storage based on their demand. [1] These services are available on a metered basis (i.e. pay per usage) where users have access to unlimited resources but only pay for those, which theyuse. With increasing demand for computing power, storage and infrastructure, cloud computing have become more prevalent in today's age. Many companies like Amazon, Microsoft and Google have started providing cloud services to the computers demanding for it. With the help of Cloud computing, users can have easy access to storage, infrastructure and cloud facilities over the internet. Cloud services platform such as Amazon Web services, Microsoft Azure and Google Cloud Platform maintains the network- connected hardware required for these application services, while you can use these cloud services through a web application.

1.2 History of Cloud Computing:

A group of technology executives coined the term "Cloud Computing" at the office of Compaq in 1996. [2] They believed that all Business Software should move to the web and file storage for users should become common. Just after that, Salesforce.com revolutionized the concept of Cloud computing by providing delivery of enterprise applications via the web. [3] Slowly, other big players in the market provided other cloud services platform. In 2002, Amazon launched cloud services in the name of Amazon web services followed by Google Apps in 2006(started by Google) and Microsoft Azure (launched by Microsoft) in 2009.

1.3 Data Security in Cloud Computing:

From the user's perspective, security in Cloud Computing, especially Data Security is the main obstacle for endorsing Cloud Computing services. [4] Moreover, Data Security is prevalent in all three types of Cloud Service Models. Data security in different Cloud Service Models:

1) **Software as a Service (SaaS):** In Software as a service, users are able to use the application provided by the Cloud service vendor running on the Cloud infrastructure. SaaS applications mainly include business applications such as ERP, CRM, SCM, etc. Organizations, which do not have the resources to develop their own applications, usually buy the applications from cloud-based vendors for their business purposes. The data that is used by the applications for processing is usually stored in the cloud itself. Moreover, this data is stored in the form of plaintext, which makes it more vulnerable to different types of attacks. Users have the least control over the security in this case, as both the application and the data are stored in the Cloud and it becomes the primary responsibility of the vendor to provide security in Software as a Service (SaaS) facility.
2) **Platform as a Service (PaaS):** In Platform as a service, users can deploy their own application on the cloud infrastructure without installing anything on their own machine. PaaS provides resources such as operating system, a software framework to build high-level applications. Data security in PaaS can take place in three forms,
   a. By use of third party software frameworks in the Cloud, which may or may not vulnerable to attacks?
   b. Complexity in developing a secure application by the user that is hosted on the Cloud.
   c. Data Security issue can also take place while transferring the code of the application from the Cloud to the local machine.
   d. If any of the vulnerabilities are present, then Data security can take place and the code of the application can be revealed to the attacker. Hence, in the case of Platform as a Service, responsibility of the security rests partly with the user and partly with the provider.
3) **Infrastructure as a Service (IaaS):** In Infrastructure as a Service, users are provisioned to

harness storage and computing power provided by the cloud service vendors. Users use Cloud Infrastructure to run and deploy arbitrary applications and Operating systems as per their requirement. In IaaS, users have better control over the security as there is no vulnerability in the Virtual machines provided by the service provider. However, the Cloud vendor also provides the computation power, storage facility and the network facility, which can be vulnerable to the attack. VMs are usually copied to provide flexibility to the user but it can lead to unintentional data leakage. Some important information in the form of passwords may be recorded while creating an image of the VMs.

2. SOLUTIONS RELATED TO DATA SECURITY IN CLOUD

2.1 Computing

2.1.1 Homomorphic Encryption: Many Cloud Vendors are using traditional cryptography algorithms to ensure that confidentiality of the data. Encrypted data can be used to send and store data to the cloud, which can ensure that the data is secure. Many Private and Public key encryption techniques such as RSA and AES are used for this purpose. In addition to that, Secure Socket Layer (SSL) is used to ensure the security of the data while transmitting to and from the cloud. While the security is achieved at the time of transmitting and storing the data in the cloud, processing of the data cannot take place in an encrypted format. To resolve this issue, Homomorphic Encryption technique was developed by IBM in 2009. [5] Homomorphic encryption provides the facility of processing the data without being decrypted.

When we transfer the data to the cloud, we use different encryption techniques to make sure the security of the data being transferred. However, when the data needs to be processed on the remote server, the cloud service provider needs to access the data in the plaintext format for processing it. Hence, for that purpose, we need Homomorphic Encryption so that

we do not have to decrypt the data for processing every time. Any operations on the encrypted data should provide us with the same result, which we would have otherwise received on the decrypted data. There is another important problem, which needs to be addressed. Suppose, if we do not opt for the Homomorphic encryption. Then, for decrypting the data over the cloud, the user needs to share the information of the private key with the cloud vendor. If public key encryption is used, then cloud vendor needs to save the information of many keys from different users. Hence, Homomorphic encryption provides an efficient way to solve this problem.

Homomorphic Encryption takes into consideration only the encrypted data without knowing the private key and performs operations on them. The user does not need to provide the private key to the cloud vendor for performing any action on the data. Thus, the user is the only holder of the key. When we compare the result of the operation done on the encrypted as well as the decrypted, it should come as same.

For explaining the concept of Homomorphic Encryption, let us consider a simple example of Julius Ceaser Cipher by shifting characters by 7 places, which is partially homomorphic with respect to the concatenation operation.

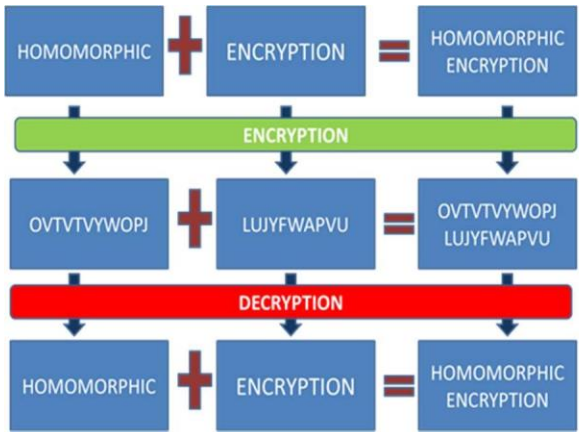


Fig. 1: Homomorphic Encryption

In the above example provided in Figure 1, we can see that it was not necessary to decrypt the cipher text before performing the concatenation operation.

Hence, we can say that Julius Ceaser Cipher (Shift - 7) is homomorphic on concatenation operation.

E (HOMOMORPHIC) = OVTVTVYWOPJ E (ENCRYPTION) = LUJYFWAPVU

E (HOMOMORPHIC) + E (ENCRYPTION)

= OVTVTVYWOPJ + LUJYFWAPVU E (HOMOMORPHICENCRYPTION)

= OVTVTVYWOPJLUJYFWAPVU

2.1.2 Fully Homomorphic Encryption: As we saw in the above example, Homomorphic concatenation arrives at the same result, as does the non-homomorphic concatenation. However, this is not always the case. Hence, we need a Homomorphic encryption, which is able to solve all the operations in the cloud. The encryption, which can perform all the operations on the ciphertext (NOT, AND, OR and XOR), is known as fully homomorphic encryption. [6]

An encryption is said to be fully homomorphic if it follows the below operations,

$$E(M1 + M2) = E(M1) * E(M2)$$

 Additive Homomorphic

AND

$$E(M1 * M2) = E(M1) * E(M2)$$

 MultiplicativeHomomorphic

For explaining the concept of Additive Homomorphic Encryption, we are taking the example of Paillier Algorithm. We have two ciphers C_1 and C_2as described below,

$$C_1 = g^{N1}.r^n mod n^2 \quad C_2 = g^{N2}.r^n mod n^2$$

$$C_1.C_2 = g^{N1}.r^n.g^{N2}.r^n mod n^2$$

$$= g^{N1+N2}.(r_1 r_2)^n mod n^2$$

Hence, with the help of above formula, we can say that Paillier Algorithm is Additive Homomorphic.

For explaining the concept of Multiplicative Homomorphic Encryption, we are taking the example of RSA Algorithm. Again, we have to ciphers C_1 and C_2as described below,

$$C_1 = m^e mod\ n \quad C_2 = m^e mod\ n$$

1

$$C_1 . C_2 = m^e m^e mod\ n = (m_1 m_2)^e mod\ n$$

Hence, with the help of above formula, we can say that RSA Algorithm is Multiplicative Homomorphic. It is generally proposed to use fully Homomorphic encryption for Cloud Security, which can perform all kinds of operation on cipher text without decrypting. However, very complicated calculations are involved in the encryption system. Moreover, the cost of computation and storage is high with Homomorphic Encryption. [7] Because of the inefficiency of the Homomorphic Encryption, it is not used for the implementation of cloud security.

2.2 Elliptic Curve Cryptography

Neal Koblitz and Victor Miller proposed elliptic Curves (EC) in 1985 for the application of Cryptography. Elliptic Curve Cryptography is a public key cryptographic algorithm. This algorithm provides the user with a public and private key. The public key is used for encryption and the private key is used for decryption.

For the Application of Data Security in Cloud Computing, Elliptic Curve Diffie-Hellman Key Exchange Algorithm is used. Strength of ECC depends on the difficulty to solve the elliptic curve discrete logarithm problem [8]. Equation of the Elliptic Curve can be defined as below,

$y^2 = x^3 + ax + b$

Therefore, if P and Q are two points on the curve, then point R can be defined as,

$R = P + Q$

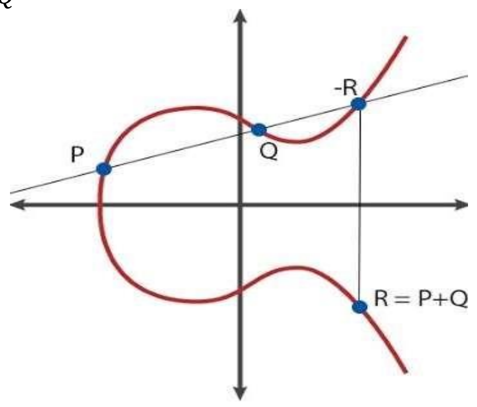


Fig. 2: Elliptic Curve

Let us see how Elliptic Curve Cryptography works for Cloud Security. Suppose, a user wants to retrieve a file from cloud securely. We will denote user as U and Cloud Vendor as V in this case. The Cloud vendor will encrypt the file with user's public key and will then transfer the file to the user. The user can then decrypt the received file using his own private key. In this manner, a secure transfer of file takes place between the Cloud and the user using Elliptic Curve Cryptography.

Algorithm for Data Security in Cloud computing using ECC:

Both the user and the cloud vendor agree to some publicly known data item.

a) The elliptic curve equation

i) values of U and V

ii) prime, p

b) The elliptic group computed from the elliptic curve equation

c) Basepoint, R, taken from the elliptic group

Key generation:

1) User (U) selects an integer dU. This is User's private key.
2) User then generates a public key PU=dU*R
3) Cloud Vendor (V) similarly selects a private key dV and computes a public key

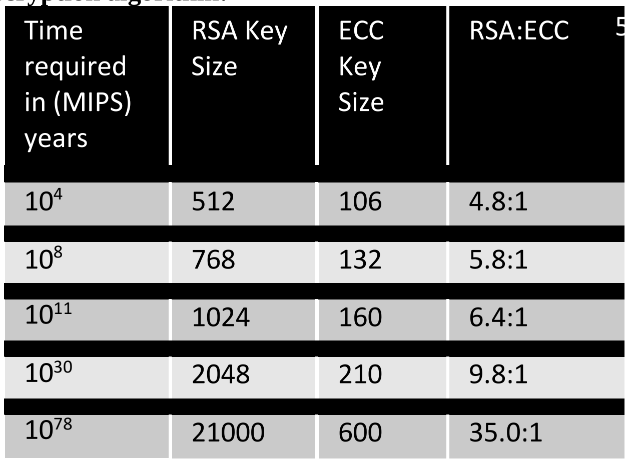
a) PV= dV *R

4) User (U) generates the secret key K= dU *PV. B generates the secret key K= dV *PU.

Encryption algorithm:

1) Suppose User U wants to retrieve an encrypted message from Vendor V Vendor V takes plaintext message M and encodes it onto a point, PM, from the elliptic group.
2) Vendor chooses another random integer, k from the interval [1, p-1]
3) The cipher text is a pair of points PC = [(kR), (PM + kPU)]
4) Send ciphertext PC to User U.

Decryption algorithm:



Time required in (MIPS) years	RSA Key Size	ECC Key Size	RSA:ECC
10^4	512	106	4.8:1
10^8	768	132	5.8:1
10^{11}	1024	160	6.4:1
10^{30}	2048	210	9.8:1
10^{78}	21000	600	35.0:1

Fig. 3: Comparison of ECC with RSA

Even though ECC has many advantages to make it compatible for Cloud security, it has some serious disadvantages, which cannot be ignored. The User U will take the following steps to decrypt cipher text PC.

1) User U computes the product of the first point from PC and his private key dU, dU * (kR)
2) User U then takes this product and subtracts it from the second point from PC, (PM + kPU) – [dU (kR)] = PM + k (dU*R) – dU (kR) = PM
3) User then decodes PM to get the message, M.

Thus, using Elliptic Curve Cryptography, Data security in Cloud Computing can be achieved using the above algorithm. [9] Elliptic Curve Cryptography is very efficient for Cloud Security as it reduces the computational power when compared to another public key encryption algorithm such as RSA. Elliptic Curve Cryptography also uses a smaller encryption key used for encoding and decoding the message. The smaller the size of the key, the less computing power it takes for encryption. A 160-bit ECC public key is claimed to be as secure as 1024-bit RSA or Diffie- Hellman key (IEEE Computer, August 2000). Biggest disadvantage for using Elliptic Curve Cryptography is that it increases the size of the cipher text much more as compared to another public key encryption such as RSA. In addition to that, the implementation to ECC is much more complex and difficult as compared to other public key encryptions. Due to its complexity, it introduces a likelihood of producing more implementation errors as compared to other algorithms. Lastly, it is difficult to explain or justify the complexity of the algorithm to the client, which makes it less likely to be used for Cloud architecture.

2.3 Honey Encryption

Ari Juels together with Thomas Ristenpart of the University of Wisconsin has developed a new encryption system known as Honey Encryption. [10]This Encryption system proves to be highly resilient against Brute force attack. With the help of this Encryption system, if ciphertext is decrypted with the incorrect key, it produces a plausible looking yet incorrect plaintext. The incorrect key will generate a fake plaintext when used while decrypting the data. The attackers think of the fake plaintext as a legal message as it looks like a plausible plaintext.

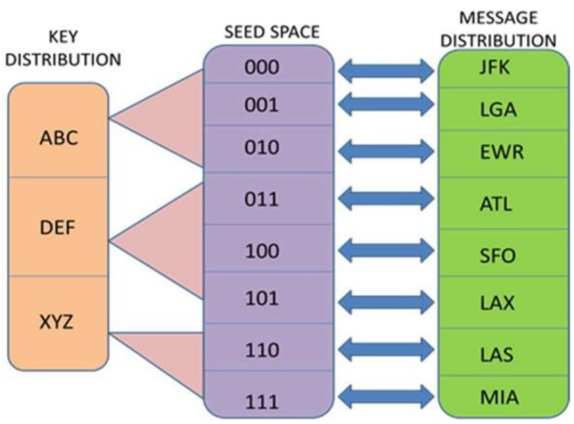


Fig. 4: Distribution Transforming Encoder

We will describe below how the original Honey Encryption works, which Juels and Ristenpart developed. [11]For implementing Honey Encryption, we need a message space M that contains all possible messages. By using Distribution Transforming Encoder (DTE), we map each message with a seed space S. DTE acts as a function F mapping seeds, which are just bit strings of a length n (n being a predetermined value), from a seed space into the message space. For example, in Figure 4, "JFK" from message space is being mapped to Seed Space "000".

A good Distribution Transforming Encoder is one that models the message distribution well, in the sense that if you pick a seed uniformly at random and then apply the DTE to it, you will get back the message distribution. Moreover, the function F of DTE is invertible which means that if you pick a message, you can find the corresponding seed. Honey encryption also makes use of a mapping from Key (Password) Distribution to Seed Space using a function G. G maps keys (passwords) randomly onto the seed space like a Hash function. For example, in Figure 4, we are assuming that password "ABC" is mapped to 000, 001 and 010.

Therefore, to encrypt a message M under a Password K, we first compute a seed corresponding to the message of the function F inverse.

$$S_N = F^{-1}(m)$$

After the seed is computed for the message, we compute a seed corresponding to the password.

$$S_k = G(k)$$

Once both the seeds are computed, encryption can be achieved by XOR both the seeds.

$$C = S_k \; XOR \; S_N$$

For example, consider the encryption of Airport codes in Figure 4. Our message space M consists of different Airport codes, M = {JFK, LGA, EWR, ATL, SFO, LAX, LAS, MIA}. We have taken the seed space of 3 bits in this example. Now, consider the process of encrypting an airport code, say "ATL" under the password "XYZ". Using the DTE, we find the seed corresponding to our airport code "ATL" to get the seed value as 011. At the same time, DTE finds the seed corresponding to our password "XYZ" to get the seed value as 110. After retrieving the seed value for both the message and the password, DTE XOR them together, i.e. 011 XOR 110 to get 101 as the cipher text.

$$G(k) \; XOR \; F^{-1}(m) = S_k \; XOR \; S_N = 011 \text{ XOR } 110 = 101 = C$$

To decrypt now, DTE finds the seed value for the password "XYZ" to retrieve 110. After retrieving the seed value of the password, DTE XOR the seed value 110 with the cipher text 101 to recover the original seed as 011. DTE takes the seed after decryption, maps it to the message distribution and recover the correct airport code as "ATL".

Now, if you try to decrypt the message with the wrong password, it will generate a plausible looking yet incorrect message. For example, let us try to decrypt the same cipher text 101 using another password "DEF". DTE will find the seed value for the password "DEF" to retrieve 011. After retrieving the seed value of the password, DTE XOR the seed value 011 with the cipher text 101 to get the seed value as 110. DTE takes the seed to get the airport code as "LAS". In this case, we get a perfectly valid looking airport code but the airport code was incorrect. We get the airport code as "LAS" instead of the airport code as "ATL". The decryption under any password yields a valid message.

The above-explained method of Honey Encryption can be used for encrypting the messages in the Cloud Architecture. After the messages are encrypted using Honey Encryption, we introduce a method of Secure Repository Manager (SRM) specifically for Cloud computing. [12] SRM can be used for securely storing the data on the Cloud Architecture. The job of SRM is to divide the data into small chunks and upload the data randomly onto the cloud server. The size of the chunk was directly related to the level of the security of the data. Division of the data into smaller chunks means highly sensitive and critical data. Division of the data into medium chunks means the data is less sensitive and critical. Division of the data into bigger chunks means the data was not at all sensitive or critical.

After the encryption process, the data chunks were stored at random locations. The information about the location and the server was stored in the Secure Repository Manager (SRM). For the data retrieval process, necessary information in the form of a password, chunk size and location is gathered by SRM at the client side. After the information gathering is done, decryption process takes place and the data is kept in order for maintaining the original form of the message. For avoiding the damage of data while transferring from cloud to the user, secure connection in the form of HTTPS and SSH is used. If any means still damage the data, a request is resent to the cloud server, for the user to have the data in the original form.

2.4 Secure Repository Manager Database (SRM DB)

The information about the file such as File ID, name, size, cloud server path while file uploading, file chunks and the password key is stored in the SRM. All this information is stored in the SRM DB as portrayed in Figure 5.

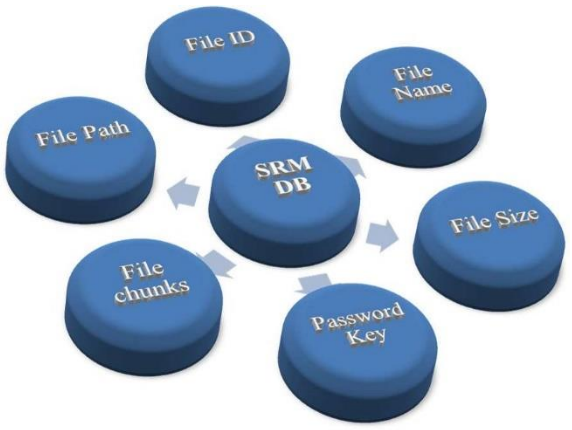


Fig. 5: SRM Database

Encryption of the data was done before uploading the file on the server and decryption of the data happens while downloading the file on any legitimate (authenticated) user. Legitimate user must have the password key to decrypt the data from the cloud. Below is the algorithm for uploading and downloading the file from the cloud server.

File Uploading (Encryption):

1) The user sends an Access Request message to the SRM for Authentication.
2) If the authentication is successful, SRM creates a secure connection HTTPS between the cloud server and the user.
3) Once the secure channel is created, the user picks up a file to upload to the cloud server.
4) The user is prompted for a password before the actual uploading starts.
5) The file is encrypted using Honey Encryption and Password as the key.
6) Once the file is encrypted, it is converted into smaller chunks before uploading the file on the cloud server.
7) After division into smaller chunks, the chunks are uploaded to the cloud server.
8) Session close if the chunks are uploaded successfully.

File Downloading (Decryption):

1) The user sends an Access Request message to the SRM for Authentication.
2) If the authentication is successful, SRM creates a secure connection HTTPS between the cloud server and the user.
3) Once the secure channel is created, the user picks up a file to download from the cloud server.
4) The user enters the password for the purpose of decryption.
5) Information in the form of a password, chunk size and location is gathered by SRM.
6) Decryption process takes place and the data is kept in order for maintaining the original form of the message.
7) After decryption, the file is downloaded to the user's machine.
8) Session close if the download is successful.

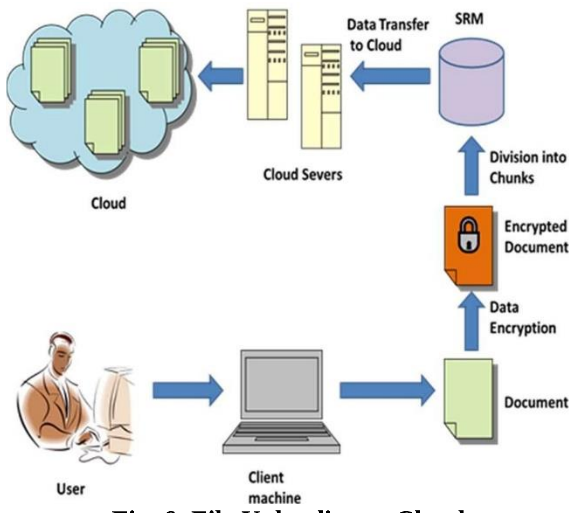


Fig. 6: File Uploading to Cloud

Honey Encryption proves out to be the best amongst the three solutions discussed in this article. However, there is some limitation in Honey Encryption which can compromise the Data security in Cloud computing. For example, Honey Encryption is vulnerable to known-plaintext attack in which the attacker already has the knowledge about the target message. If the attacker has an idea that a plaintext must match in order to be legit, they can even brute- force the data encrypted by the Honey Encryption.

3. PROPOSED SOLUTION FOR FUTURE WORK

The main problem we are facing with Data Security in Cloud Computing using Honey Encryption is the Known-plaintext attack. If the problem related to known-plaintext attack is addressed, the desired level of security can be achieved in Cloud Computing.

We know that the known-plaintext attack can be prevented using symmetric-key algorithm standard such as AES (Advanced Encryption Standard). For the implementation of AES in Honey Encryption, the seed space of DTE should be equal to the block size of the AES cipher. [13]

By the combination of Honey Encryption and AES, we can achieve the highest level of security in Cloud Computing. Honey Encryption method described in this paper uses Password based encryption for Cloud Security. Hence, it is prone to known-plaintext attack. If the password-based encryption is replaced by symmetric key encryption such as AES, the known-plaintext attack can be avoided.

4. CONCLUSION

Security in Cloud Computing is still a challenging topic as more and more people and organizations are moving towards Cloud. To make the users convinced about the security of Cloud Computing, a lot of work still needs to be done. Honey Encryption provides better security as compared to the other solutions available for Data Security in Cloud Computing. Homomorphic Encryption and Elliptic Curve Cryptography have few limitations in the form of high storage and computation cost, as well as, complexity. The current solution of Honey Encryption using Password-Based Encryption also has limitation for being prone to known-plaintext attack. In future, integration of Honey Encryption and AES can be evaluated to prevent known-plaintext attack for the purpose of Cloud security. Hence, we support future developments and the usage of Honey Encryption for the purpose of Cloud Security.

5. REFERENCES

[1] P. Mell and T. Grance, "The NIST Definition of Cloud," September 2011. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf.

[2] A. Regalado, "Who Coined 'Cloud Computing'?," MIT Technology Review, 2011.

[3] A. Mohamed, "A history of cloud computing," March 2009. [Online]. Available: http://www.computerweekly.com/feature/A- history-of-cloud-computing.

[4] K. Hashizume, D. G. Rosado, E. Fernández- Medina and E. B. Fernandez, "An analysis of security issues for cloud computing," *Journal of Internet Services and Applications*, 2013.

[5] B. Prince, "IBM Discovers Encryption Scheme That Could Improve Cloud Security, Spam Filtering," 25 June 2009. [Online]. Available: http://www.eweek.com/security/ibm-discovers-encryption-scheme-that-could-improve-cloud-security-spam-filtering.

[6] M. TEBAA, S. E. HAJJI and A. E. GHAZI, "Homomorphic Encryption Applied to the Cloud," *Proceedings of the World Congress on Engineering,* vol. 1, 2012.

[7] Y. Sun, J. Zhang, Y. Xiong and G. Zhu, "Data Security and Privacy in Cloud Computing," *International Journal of Distributed Sensor Networks,* 2014.

[8] S. Bollavarapu and B. Gupta, "Data Security in Cloud Computing," *International Journal of Advanced Research in Computer Science and Software Engineering,* vol. 4, no. 3, 2014.

[9] V. Gampala, S. Inuganti and S. Muppidi, "Data Security in Cloud Computing with Elliptic Curve Cryptography," *International Journal of Soft Computing and Engineering,* vol. 2, no. 3, 2012.

[10] T. Ristenpart and A. Juels, "Honey Encryption: Security Beyond the Brute-Force Bound," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2014.

[11] A. Juels, "The Password That Never Was," in *Center for Research on Computation and Society*, 2014.

[12] R. Masood and M. Aslam, "INNOVATIVE APPROACH ENSURING SECURITY AND PRIVACY IN CLOUD COMPUTING," *Pakistan Journal of Science,* vol. 68, no. 1, 2016.

[13] J. Jaeger, T. Ristenpart and Q. Tang, "Honey Encryption Beyond Message Recovery Security," *Annual International Conference on the Theory and Applications of Cryptographic Te chniques,* 2016.