International Journal of Engineering & Technology, 7 (1.1) (2018) 675-678

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Security enhanced using honey encryption for private data sharing in cloud

N. Srinivasu [1]*, Masood Sahil [2], Jeevan Francis [2],Sure.Pravallika [2]

[1] *Professor, Koneru Lakshmiah Educational Foundation,Department of CSE,India*
[2] *Student, Koneru Lakshmiah Educational Foundation,Department of CSE,India*
**Corresponding author E-mail: masoodsahil.sk16@gmail.com*

Abstract

In today's modern age technology as there is a production of vast amount of data, it is getting very difficult to store such a vast amount of information. The best way to store this huge data is on cloud. As nowadays business organisations are moving towards cloud to store their data, security remains the primary concern. Is the data securing enough on the cloud or not? One of the ways to secure data on cloud is by providing security on cloud through Honey Encryption. Juels&Ristenrpart introduced honey encryption and showed how to achieve message recovery security even in the face of attacks that can exhaustively try all likely keys. Honey Encryption is a new encryption scheme that ensures the messages decrypted with invalid keys yield a valid looking message. In this paper, we present our implementation of Honey Encryption and apply it to useful real-world scenarios such as providing security to files which are been saved in cloud. The files contain variety of information in it. We also provide assurance against brute force attacks.

Keywords: *Cloud Security; DTE (Distributed Transforming Encoder); Fully Homomorphic Encryption; Homomorphic Encryption; Honey Encryption.*

1. Introduction

1.1. Cloud computing

Cloud computing is an internet-based computing where in the devices and computers are provided with shared computing resources and storage based on demand. [1] These services are based upon pay per usage basis where user can access to unlimited resources but only pay for those resources which they use. With increasing in demand for computing power, storage and infrastructure,cloud computing has become more useful in today's age. [2] Many companies like Amazon, Microsoft and Google have started providing cloud services to the computers demanding for it.

1.2. Cloud security

Security in cloud computing, especially Data security is the main obstacle for spending on cloud services, and [3] moreover data security is prevalent in all three types of Cloud services models.
Data security in Different service models:

1) Software as a Service (SaaS):

In Software as a service user can 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 from cloud-based vendors for their business purposes. The data that is used by the applications for processing is usually 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 user 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.

Copyright © 2018 N. Srinivasu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

2. Solution related to data security in cloud computing

2.1. Homomorphic encryption

Many business people are using traditional cryptography algorithms to ensure that confidentiality of the data. Encrypted data can be used to send and store data in the cloud, which can ensure that the data is secure. Many private and public key encryption schemes such as RSA and AES are used for this purpose. In addition to this, 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 datacannot take place in an encrypted format. To resolve this issue,Homomorphic Encryption technique was developed by IBM in 2009.[4] Homomorphic encryption provides the facility of processing the data without being decrypted.

When Vendors transfer the data to the cloud, they use Different encryption techniques to make sure, they 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 don't have to decrypt the data for processing every time. Any operation 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 only the encrypted data without knowing the private key and performs operations on the. The user doesn't 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.

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.

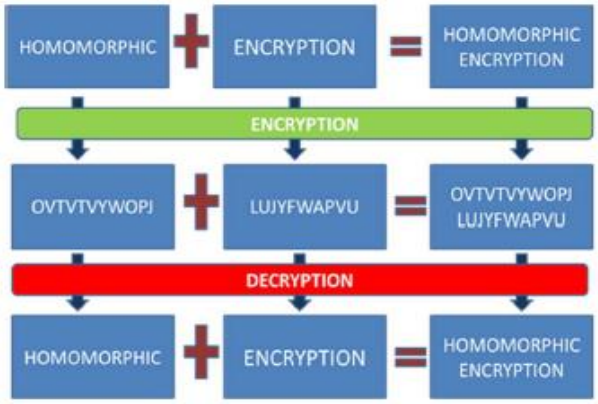

Fig. 2.1: Homomorphic Encryption.

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

2.1.1. 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 cipher text (NOT, AND, OR AND XOR), is known as fully homomorphic encryption. [5]

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

Additive Homomorphic:

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

AND

Multiplicative Homomorphic:

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

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. [6] Because of the inefficiency of the Homomorphic encryption, it is not used for implementation of cloud security.

2.2. Honey encryption

Honey Encryption scheme was developed by Ari Juels, Former chief scientist of the RSA, and Thomas Ristenpart from the University of Wisconsin. Honey Encryption is best suited in the situations where the encrypted data is obtained from the passwords. [7] This Encryption technique provides highly resilient against Bruteforce attacks. If an attacker tries to carryout brute force attack in the situations where the encrypted data is obtained by using the Honey Encryption security makes it complicated for an attacker to know if he has correctly guessed a password (or) Encryption key. For an example, if an attacker tries to get a credit card number by making 100 attempts, then for all 100 attempts they will be getting 100 false credit card numbers. Each Decryption is going to look as plausible as others. The attacker has no way to distinguish which is correct.

2.2.1. Distribution transforming encoder

We will describe below how the original honey Encryption works, which Juels and Risternpart developed. [8] 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 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".

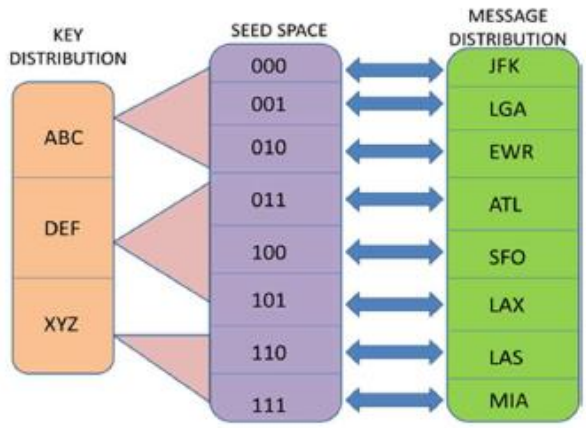

Fig 2.2: Message Mapping on Seed Space.

A good Distribution Transforming Encoder is one that models the message distribution well, in the sense that if u pick a seed uniformly at random and then apply 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 message of the function F inverse.

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

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.

$C = S_k \text{ XOR } S_m$

For example, consider the encryption of Airport codes in Figure4. Our message space M consists of different Airport codes, M = {JFK, LGA, EWR, ATL, SFO, IAX, LAS, MIA}. We have taken the seed space of 3 bits in the example. Now, consider the process of encrypting an airport code, say "MIA" under the password "DEF". Using the DTE, we find the seed corresponding to our airport code "MIA" to get the seed value as 001. After retrieving the seed value for both the message and the password, DTE XOR them together, i.e, 111 XOR 011 to get the 100 as the cipher text.

$G(k) \text{ XOR } F^{-1}(m) = S_k \text{ XOR } S_m = 111 \text{ XOR } 011 = 100 = C$

To decrypt now, DTE finds 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 cipher text 100 to recover the original seed as 111. DTE takes the seed after decryption, maps it to the message distribution and recover the correct airport code as "MIA".
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 100 using another password "ABC". DTE will find the seed value for the password "ABC" to retrieve 111. After retrieving the seed value of the password, DTE XOR the seed value 001 with the cipher text 100 to get the seed value as 111. DTE takes the seed to get the airport code as the "LAX". In this case, we get a perfectly valid looking airport code, but the airport code was incorrect. We get the airport code as "LAX" instead of the airport code as "MIA". The decryption under any password yields a valid message.
Algorithm:
File Uploading and Downloading-:

a) The admin will upload a file in the cloud with attributes like File_ID, File_Name, Caption for that file.
b) The File has been secured with a special key.
c) When the User want to download that file, he will login first and choose the file to download.
d) Then here a Verification Key will be sent to his personal mail, the key will be of 24-bits key.
e) Then the User must enter that key here at the Verification Key module, then the file will be Download.
f) If any hacker tries to download the file illegally and by mistake he enters the wrong verification key then it won't be showing your inserted key is wrong, but the file will be available for him, but the file will be fake which will not be the original file.
g) And the attacker thinks that the file is same, and he won't be trying again for that file.

Honey Encryption proves out to be the best amongst the two solutions discussed in this paper. 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 to be legit, they can even brute-force the data encrypted by the Honey Encryption.

3. Results



Fig 3.1: Wrong Password Is Given.



Fig 3.2: A Fake File Is Created.

4. Proposed solution for future work

The common problem we are facing with data security in cloud computing using Honey Encryption is the Known plain text attack. But it can be prevented using symmetric key algorithm standard such as AES, and also if we traced out the attacker after entering the wrong key by his MAC address then we shall block him after tracing his MAC address.

5. Conclusion

Security in Cloud Computing is still a challenging topic as more and more people and organizations have already moved to Cloud. To make the users convinced about the security of Cloud Computing, a lot of work still need to be done. Honey Encryption pro-

vides better security as compared to the other solutions available for data security in cloud computing. Homomorphic encryption has few limitations in the form of high storage and computation cost, as well as, complexity. The current solution 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 Cloud security. Hence, we support future developments and the usage of Honey encryption for Cloud Security.

References

[1] P. Mell and T. Grance, "The NIST Definition of Cloud," September 2011. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspec ialpublication800-145.pdf.

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

[3] K. Hashizume, D. G. Rosado, E. FernándezMedina and E. B. Fernandez, "An analysis of security issues for cloud computing," Journal of Internet Services and Applications, 2013.

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

[5] 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.

[6] 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.

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

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