ASIAN JOURNAL OF PHARMACEUTICAL AND CLINICAL RESEARCH

INNOVARE ACADEMIC SCIENCES
Knowledge to Innovation

Special Issue (April)

Online - 2455-3891
Print - 0974-2441

Full Proceeding Paper

SECURE AND RECOVERABLE SPLIT KEY MANAGEMENT TECHNIQUE FOR CLOUD STORAGE

THEOPHILUS RAKESH S, PRADEEP KV

Department of ???, VIT University, Chennai Campus, Chennai, Tamil Nadu, India. Email:asha.s@vit.ac.in

Received: 23 January 2017, Revised and Accepted: 03 March 2017

ABSTRACT

Establishing mutual trust between a cloud service provider (CSP) and a client has always been a challenge. Managing the key as a whole on either of these sites poses a security risk and also questions the integrity and availability of the data itself. In this paper, we propose an effective solution to manage key at the client's location, while the CSP still manages a portion of the key. The proposed technique secures the key itself and also provides a fail-safe mechanism to retrieve the key if lost.

Keywords: Cloud service provider, Certificate authority, Key management problem, Plain text, Ciphertext.

DOI: http://dx.doi.org/10.22159/ajpcr.2017.v10s1.19586

INTRODUCTION

With the growing demand and usage of cloud services, it is imperative to provide security to the data at a prime level. To achieve optimal security, the data are encrypted using algorithms such as RSA and Diffie-Hellman. These algorithms generate a "key" to actually perform the encryption. Managing this "key" is a part of the key management problem. If this key is revealed or compromised, it poses a direct threat to the integrity of the data.

Managing this key at the clients' site or at the cloud service providers' (CSP) site poses a serious security threat to the data itself. The existing security scheme does not allow the client or the CSP to save or choose this key. A third-party termed as a certificate authority (CA) is involved. The CA chooses the key that the client and the CSP require for communication. This again exposes a security threat. What if, the CA is compromised? Hence, this is not really an optimal solution for this problem. Effective synchronization is also a problem in this case. To overcome synchronization delay, some of the public keys come pre-installed in the operating system; we use these days. Although the time one requires to crack the private key generated by the CA is exponential, the very fact that it can be cracked makes the existing system less attractive. To overcome this, the existing system uses some key discarding techniques. Where a key is discarded or destroyed after a period. This again raises another question: How many times should one update their local machines with new keys? The major drawback of the existing system is that it does not provide an effective solution for key-recovery if lost.

Securing the key and proving a key recovery mechanism ensures trust between a client and a CSP. In this paper, we propose a technique to manage the key at the client and the service providers' site while the integrity of the key remains protected. We encrypt the key and split it into equal halves. We term them sub-keys. One-half is stored at the client's location and the other at CSPs location. The hashing technique is not revealed to the CSP. If the actual key is lost at the clients' site, we take the sub-key from the CSP and merge it with the sub-key at client to regenerate the actual key.

LITERATURE SURVEY

Assorted key-management techniques were studied from the paper [1]. Symmetric key-management scheme, group key management, and various other new key-management techniques were understood from this paper [3]. This journal was greatly helpful in understanding various encryption algorithm and its role in key management. The symmetric and asymmetric key algorithms and various other hashing algorithms were understood from this journal. The mathematics behind them and how they are used in real life was also understood. The necessity of securing the data was understood from the paper [4]. How aggregated data can be hashed without any hassle and the various drawbacks in handling large amount of data for hashing to achieve security was also understood. How key management is used in this scenario was also understood. The concept of group key. Splitting key into n parts and installing them at various client location was understood from the paper [5]. The concept of symmetric key usage was deeply studied with the help of research paper [7]. The classification of the symmetric key, how they are used in wireless sensor networks and its management schemes were understood. Key distribution techniques were also understood from this paper. This paper was useful in identifying how the split key technique can be implemented in symmetric key. The concept of split key-management was derived from the paper [8]. The various methods of key-management were understood from this paper. How key-management applies to a cloud storage platform was also understood from this paper. The use of hashing algorithms was studied from the paper [10]: Data encryption standard, advanced encryption standard, and Blowfish: Symmetric key cryptography algorithms simulation based performance analysis. A brief study on group key management was studied from: Group key management technique based on logic-key tree in the field of wireless sensor network. Possible attack methods and recovery attack methods were studied from Key recovery attacks on recent authenticated ciphers.

TWO PHASE PROTOCOL

Phase I - Encrypting the data

This is a straightforward mechanism to encrypt the data where we use a symmetric or asymmetric algorithm to encrypt the data. The research focuses on managing" the key" that is used for encryption by the algorithm. We do not really pay attention to the algorithm we use. This encrypted data are then stored at the CSP's end, while the actual encryption is done at the client's site.

Phase II – "The split key technique"

Once the encryption algorithm leaves a "key" as a residue. We apply a hashing function to the key and split the actual key into two equal "sub-keys." This can be done vice-versa as well. This is done at the client's site.

THE SPLIT KEY TECHNIQUE

The plain text or the hashed primary key is given as an input to an application which then splits it into equal halves. The application's output

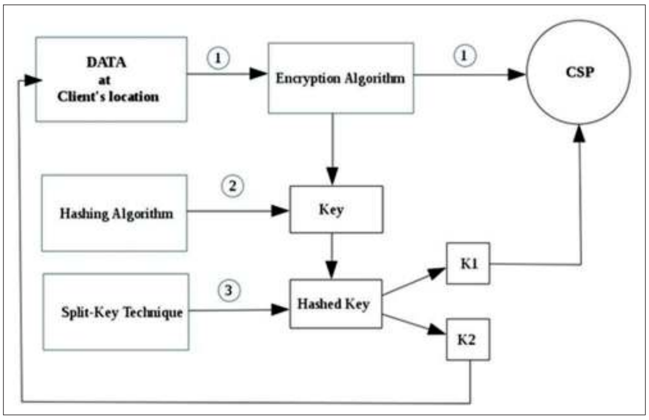

Fig. 1: Flow diagram

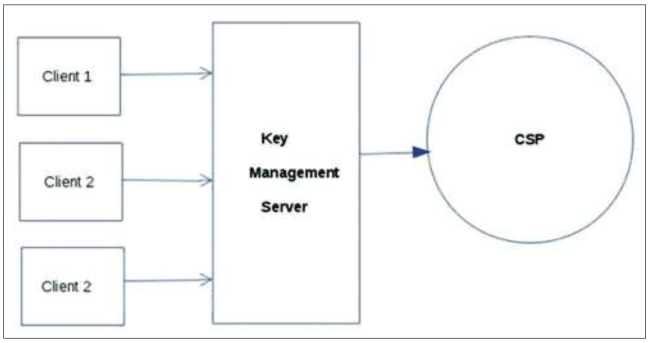

Fig. 2: With a key management server at client's location

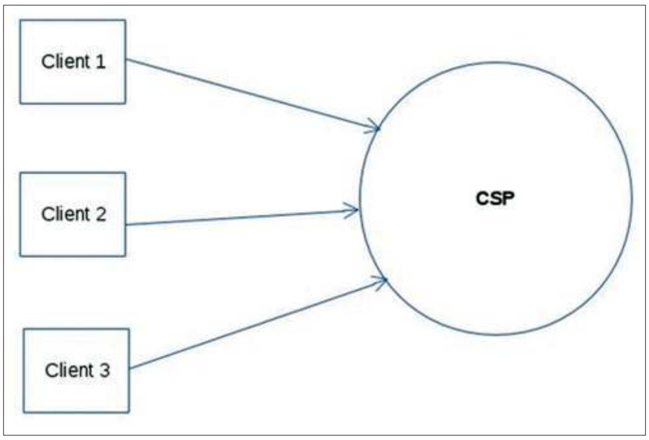

Fig. 3. Without key management server at client's location

is two files. If the key is split first then, the two halves are given as input to the hashing algorithm. This hashing technique can be kept secret.

Once the sub-keys are hashed, we randomly chose one to stay at the client's location and the other to be stored at the CSP's location. Now that the key is split into parts, effective synchronization and authentication mechanisms should be in place to generate the primary key from the sub-keys and to maintain trust between the client and the CSP.

Synchronization
When a client needs to access the data at the CSP's location, the two sub-keys should be combined to generate the actual key. The same application that was used to split the key into two equal halves is used to merge the two sub-keys to form the primary key in cipher text.

This merged output file is taken as the input to the hashing algorithm. Here, the ciphertext file is converted into a plain text file.

Authentication
No additional authentication is required because the primary key cannot be regenerated with out using the right subkeys. The subkey present at the CSP's location could be the authentication for the client and vice-versa.

DESIGN

As the owner of the data, it gives all the right for a client to generate and manage their own key. We assume, security at all levels in the client's premises is optimal.

The algorithms and the hashing techniques used are not our primary concern. The key that is used for the encryption by the algorithm is our main focus. Moreover, as this is done at the client's location, we trust all the other clients as well. Should the client be a huge enterprise? The proposed idea can be implemented with or without a local key management server (KMS).

With a local KMS
Introducing a KMS locally lessens the burden on the client's machine. This, however, increases administrative work and also increase security risk of KMS being compromised. But here, as the keys at KMS are already hashed and split the intruder cannot get any useful information. This method is suitable if the number of client in a LAN network is large.

Here, the KMS can also implement the splitting of the key, deploying them on the client machines based on requirement and necessity.

Without a KMS
Without a KMS, each of the client presents locally is responsible for generating their own key and performing the split-key technique. The clients then can freely request and access the data needed at the CSP at their will.

ACCESSING THE DATA AT CSP

Accessing the data at the CSP requires the actual key. In this case, the combination of the subkeys. The application that was used to split the key is used to generate the primary key by simply joining the two subkeys. As discussed earlier, there is no authentication required. The subkeys themselves act as an authentication factor.

TRANSFER OF KEYS OVER LAN

Once the key is split into halves and stored at the consumer's and provider's location, we need an effective mechanism to transfer one-half of the key at the provider's location to the consumer's location. To achieve this, separate consumer and provider programing methods are used.

This transfer of key is practically achieved using python socket programing.

The consumer method
This method is capable of receiving the key in the form of an array of bytes. As socket programming is used, byte ordering is taken care by default.

This method also comprises of the key join method so that, once both halves of the keys are gathered they are joined together.

Algorithm

1. A consumer socket is created
2. The socket is binded with the local IP and a Port no.
3. The socket is now made to listen passively at the given port no.
4. The connection received from the CSP's end is accepted
5. The contents received is stored in a local variable
6. The contents are then written to an output file.

The contents of the output file are equivalent to the file that was transmitted from the CSP

The provider method

This method, basically transfers the one-half of the key that it has to the consumer. This key can be stored or put in the provider's place by various means. It can be as simple as just copying the half of the key and physically dumping it at the provider's end. As the split keys are already hashed, there is no real security threat here.

Algorithm

1. A provider socket is created
2. The created socket is connected to the local IP address and the port no. the consumer is listening
3. The file to be transferred is referenced using file object
4. Then, the contents of the file are then read using the file object and stored in a local variable
5. The contents of this variable are then transmitted through the port no. to the given IP address byte-by-byte.

The number of bytes received at the receiver's end is in the same order in which it was transmitted.

CONCLUSION AND FUTURE WORK

As discussed throughout, splitting the key into equal halves and encrypting them provides security to the "key" itself. If the primary key is lost at the client's side, we can still recover or regenerate the primary key by merging the two subkeys. Hence, providing a fail-safe mechanism to retrieve key if lost.

Future research can include simplifying the process of splitting the key and synchronization process. An additional authentication mechanism can be in place at the CSP's location to validate if the requester is authorized.

ACKNOWLEDGMENT

Author sincere thanks Prof. Pradeep KV, SCSE in guiding his to achieve this feat. Author appreciate all the help received from his peers in the due to course of this research and also thanks Dr. B. Rajesh Kanna, Programme Chair, SCSE Cloud computing, VIT University Chennai Campus.

REFERENCES

1. Shnaikat KN, Al-Qudah AA. Assortment of key management techniques for wireless sensor networks. Int J Adv Technol Eng Res 2014;6(6):49-63.
2. Bala S, Sharma G, Verma AK. Classification of symmetric key management schemes for wireless sensor networks. Int J Secur Appl 2013;7(2):117-38.
3. European Payments Council. Guidelines on Algorithms Usage and Key Management, Version 4.0. : EPC (European Payments Council); 2014.
4. Jayaraj V, Indhumathi M, Mathimalar V, Hemalatha S, Durai U. Secure data aggregation using efficient key management technique in wireless sensor network. Int J Comput Appl 2014;89(9):6-11.
5. Metan J, Murthy KN. Group key management technique based on logic-key tree in the field of wireless sensor network. Int J Comput Appl 2015;117(12):9-15.
6. Huang D, Mehta M, Medhi D, Harn L. A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge. Hong Kong: IEEE INFOCOM; 2004.
7. Bala S, Sharma G, Verma AK. Classification of symmetric key management schemes for wireless sensor networks. Int J Secur Appl 2013;7(2):117-38.
8. Key Management for Cloud Data Storage: Methods and Comparisons. 2014 Fourth International Conference on Advanced Computing & Communication Technologies; 2014.
9. Trappe W, Song J, Poovendran R, Liu KJ. Key management and distribution for secure multimedia multicast. IEEE Trans Multimed 2003;5(4):54457.
10. Thakur J, Kumar N. DES, AES and blowfish: Symmetric key cryptography algorithms simulation based performance analysis. Int J Emerging Technol Adv Eng 2011;1(2):6-12.
11. Renjith PR, Sojan A, Gopinadhan PK. A novel method for symmetric encryption using split plaintext key pair (Pi,Ki) algorithm. Netw Secur Cryptogr NSC 2011;2:36-41.
12. Buyya R. Introduction to the IEEE transactions on cloud computing. IEEE Trans Cloud Comput 2013;1(1):1-9.
13. Liu H. Study of authentication with IoT testbed. Dartmouth MA, USA: Department of Electrical and Computer Engineering, University of Massachusetts. IEEE 14-16 April; 2015.
14. Cloud computing security challenges and threats: A systematic map. Int J Adv Eng Technol 2015;.
15. Bogdanov A, Dobraunig C, Eichlseder M, Lauridsen MM, Mendel F, Schläffer M, *et al*. Key Recovery Attacks on Recent Authenticated Ciphers. Switzerland: Springer International Publishing; 2015.