International Journal of Recent Engineering Research and Development (IJRERD)
Volume No. 02 – Issue No. 05, ISSN: 2455-8761
www.ijrerd.com, PP. 84-89

HOMOMORPHIC ENCRYPTION IN CLOUD

M. S. Bhuvaneswari [1], K. Abinaya Vasuki [2], J. Karthik Kumar [3]

[1]*Department of CSE, Mepco Schlenk Engineering College, Sivakasi, India*

[2] *Department of CSE, Mepco Schlenk Engineering College, Sivakasi, India*

[3] *Department of CSE, Mepco Schlenk Engineering College, Sivakasi, India*

Abstract: The problem of insecurity of data in cloud has curbed the progress of cloud computing movement. In practice, the existing system considers encrypting the data before moving it to the cloud. If data is encrypted before moving it to the cloud, the data cannot be accessed or searched until it is decrypted. Using homomorphic encryption we can categorize and mine data which is still in an encrypted state and upon decryption, the search result matches the exact data and as a result the data is not rendered vulnerable. This helps in secured storage of confidential files in the cloud. Safer search is enabled by searching the encrypted files. Integrity of data is preserved without disclosing it to the provider. This helps to ensure the confidentiality of the processed data in cloud computing environment.

Keywords: Confidentiality, Data Privacy, Data Processing, Decryption, Encryption, Homomorphic Encryption.

1. Introduction

Cloud Computing offers a number of benefits and services to its customers who pay for the use of hardware and software resources on demand, which they can access via internet without the need of expensive computers or a large storage system capacity and without paying any equipment maintenance fees. Cloud Computing has emerged as an important paradigm that has attracted considerable attention in both industry and academia. Cloud storage outsourcing has become a popular application for enterprises and organizations to reduce the burden of maintaining big data in recent years. However, in reality, end users may not entirely trust the cloud storage servers and may prefer to encrypt their data before uploading them to the cloud server in order to protect the data privacy. Cloud Computing has been the most promising innovation in the computing world. Its usage is still hindered by the security concerns related with critical data. The encryption of remotely stored data has been the most widely used technique to bridge this security gap. The vast usage of Cloud Computing solutions for data storage and with Big Data Analytics gaining strong foothold, the security on cloud is still at big risk.

The existing system ensures to encrypt the data stored. Also it is very easy to have secure transmission from a local machine to a cloud data store by encrypting the data to be stored and securing the channel of data transmission with key exchanges. But actually performing computations on that data stored in the cloud requires decrypting it first which makes critical data available to the cloud provider. Data Mining and other Data Analysis onto the Encrypted Database is a far distant thing to achieve by using the encryption standards available. The proposal here is to encrypt data before sending to the cloud providers thereby to enable a cloud computing vendor to perform computations on clients' data at their request, without exposing the original data. To achieve this it is also necessary to hold the cryptosystems based on Homomorphic Encryption. An optimal solution for the problem of insecurity of data is proposed and solutions to practical problems are also formulated based on Homomorphic Encryption. The confidential files of the cloud user are encrypted and stored in the cloud. Homomorphic encryption is used to search the required file among the encrypted files without decrypting them. This will provide the same results after processing as if we have searched directly on the original files eliminating the exposure of original data to the cloud provider. The retrieved files are decrypted and the required file is provided to the user. To be specific, the contribution of this work consists of:

- Encrypting the files that the user uploads to maintain the privacy of the user data.
- Encrypting each file with a separate key available only in the portal to maintain confidentiality of the user data.
- Storing the encrypted file in the cloud.
- Encrypting the user query before providing it to the cloud.
- Retrieving the files related to the user query in an encrypted form.
- Decrypting the file required by the user.

The rest of the paper is organized as follows. Section II describes the related work. Section III depicts the proposed work. Section IV discusses the results out of the experiments. Section V concludes the paper with remarks on future work.

2. Related Work

Definition [3]: An encryption is homomorphic, if: 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. Homomorphic encryption is distinguished as additive Homomorphic encryption (only additions of the raw data) like the Pailler and Goldwasser-Micali cryptosystems and multiplicative Homomorphic encryption (only products on raw data) like the RSA and El Gamal cryptosystems based on the operations performed on the raw data. [5] An algorithm is fully homomorphic if both the properties are satisfied simultaneously. Both the types of homomorphic encryptions give security to the cloud user to do the computations on encrypted data without decrypting. But Somewhat Homomorphic Encryption scheme gives a security with low complexity and Fully Homomorphic Encryption adds to it. The performance of Somewhat Homomorphic Encryption is faster than Fully Homomorphic Encryption due to its less complexity and takes less time to execute than Fully Homomorphic Encryption. The execution time of Somewhat Homomorphic Encryption is almost half of the Fully Homomorphic Encryption. [6] Searchable encryption can be realized in either symmetric or asymmetric encryption setting.

Public Key Encryption with Keyword Search (PEKS) enables a user to search encrypted data in the asymmetric encryption setting. In a PEKS system, using the receiver's public key, the sender attaches some encrypted keywords (referred to as PEKS ciphertexts) with the encrypted data. The receiver then sends the trapdoor of a to-be-searched keyword to the server for data searching. Given the trapdoor and the PEKS ciphertext, the server can test whether the keyword underlying the PEKS ciphertxt is equal to the one selected by the receiver. If so, the server sends the matching encrypted data to the receiver.

3. Proposed Work

In this section the detailed design and methodologies of the proposed work is described. The work comprises of six modules: Manage Accounts, Upload, Encrypt, Search, Decrypt and Delete. The overall system design is shown in Figure1

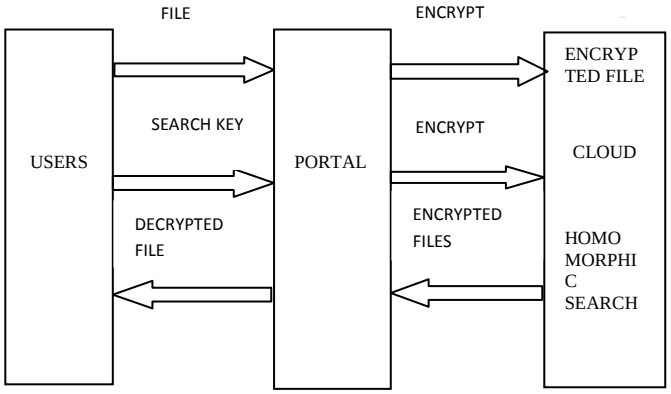

Figure 1: System Design of the proposed work

3.1 Manage Accounts

The portal has two types of users. One type of user is the administrator and the other type users are the account holders of the portal. The administrator of the portal has the permission to add users to the closed community and delete users from the closed community. The administrator adds users to the portal based on the request from the user. The user details such as name, gender, date of birth, email, preferred username, and password are stored in the account details table. When the user is added, the user can upload the files to the portal so that the file is encrypted and stored in the cloud. The administrator can delete a user account if the user quits the closed community. The details of the corresponding user are deleted from the account details table.

3.2 Upload

Once the user logs into the portal, they can upload the file to be encrypted. The file is uploaded into the portal along with the filename and the metadata about the file. The file details like file id, filename, metadata, keys to encrypt the file, userid of the user who uploaded the file are stored in the file_details table. The metadata is used to identify the file that the user wants to retrieve from the cloud among the other files being uploaded into the cloud. The user can upload any type of file like pdf, image, mp3 etc.., and each uploaded file is

encrypted using unique key values.

3.3 Encrypt

The file is encrypted before it is stored in the cloud. The file is encrypted with two different encryption algorithms (AES and Triple DES) to ensure security. Encrypting the filename and the file contents with different encryption algorithms preserves the confidentiality of the file.

3.3.1 File Name Encryption

The file name is encrypted using Triple DES algorithm. Triple DES is a symmetric-key block cipher, which applies the DES cipher algorithm three times to each data block. The availability of increasing computational power made brute-force attacks feasible on the original DES cipher because of the DES cipher's key size of 56 bits. Triple DES provides a relatively simple method of increasing the key size of DES to protect against such attacks, without the need to design a completely new block cipher algorithm. Triple DES uses a key bundle that comprises three DES keys, K_1, K_2 and K_3, each of 56 bits. The encryption algorithm is

Ciphertext = E_{K3} (D_{K2} (E_{K1} (plaintext)))

Each triple encryption encrypts one block (64 bits) of data. In each case the middle operation is the reverse of the first and last. In each case the middle operation is the reverse of the first and last. This improves the strength of the algorithm when using keying option 2, and provides backward compatibility with DES with keying option 3.The standards define three keying options:

Keying option 1 All three keys are independent.
Keying option 2 K_1 and K_2 are independent, and $K_3 = K_1$.
Keying option 3 All three keys are identical

DES is a symmetric encryption system that uses 64-bit blocks, 8 bits of which are used for parity checks. Each of the key's parity bits is used to check one of the key's octets by odd parity, that is, each of the parity bits is adjusted to have an odd number of '1's in the octet it belongs to. The key therefore has a useful length of 56 bits, which means that only 56 bits are actually used in the algorithm. The algorithm involves carrying out combinations, substitutions and permutations between the text to be encrypted and the key, while making sure the operations can be performed in both directions. The combination of substitutions and permutations is called a product cipher.

3.3.2 File Content Encryption

The file content is encrypted using AES algorithm. AES is a symmetric encryption algorithm. AES supports a block length of 128 bits and key lengths of 128, 192, and 256 bits. The number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.

a. Byte Substitution (SubBytes)

The 16 input bytes are substituted by looking up a fixed table. The result is in a matrix of four rows and four columns.

b. Shiftrows

Each of the four rows of the matrix is shifted to the left. Shift is carried out as follows:

- First row is not shifted.
- Second row is shifted one (byte) position to the left.
- Third row is shifted two positions to the left.
- Fourth row is shifted three positions to the left.
- The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other.

c. Mix Columns

Each column of four bytes is can be transformed using a special mathematical function. Four bytes of one column is given as input to the function which outputs four completely new bytes that replaces the original column. A new matrix consisting of 16 new bytes is produced as the result. This step is not performed in the last round.

d. Addroundkey

The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round

key. If this is the last round then the output is the ciphertext. Otherwise, the resulting 128 bits are interpreted as 16 bytes and begin another similar round. The encrypted file is uploaded in the cloud.

3.4 Search

Whenever the user wants to retrieve the files from the cloud, the search query given by the user is encrypted using the same key used for file content encryption and the search query is given to the cloud. The cloud searches for the file names that are related to the search query and returns the list of files with file names that are related to the search query in an encrypted form. Searching the files in the cloud that are in an encrypted form by using an encrypted search query preserves the content security in the cloud.

3.5 Decrypt

The names of the files retrieved from the cloud are decrypted using Triple DES decryption algorithm and displayed to the user. The content of the file chosen by the user is decrypted using AES decryption algorithm and the user can download the decrypted file from the portal, thus ensuring homomorphism of the data stored in the cloud.

3.5.1 File Name Decryption

The file name is decrypted using Triple DES decryption algorithm. Triple DES decryption is the same as Triple DES encryption done in reverse order. Triple DES uses a key bundle that comprises of three DES keys, K_1, K_2 and K_3, each of 56 bits. The decryption algorithm is

Plaintext = D_{K3} (E_{K2} (D_{K1} (ciphertext)))

3.5.2 File Content Decryption

The file content is decrypted using AES decryption algorithm. AES decryption is the same as AES encryption done in reverse order. A set of reverse rounds are applied to transform the ciphertext back into the original plaintext using the same encryption key.

3.6 Delete Files

The user can delete his file from the portal, by giving the file id. If the user deletes his file from the portal, the file is also deleted from the cloud.

4. Results and Discussion

In this section the results obtained in this work is described. A web portal is designed which can be used as an interface between the user and the cloud provider.

4.1 Upload

Once the user logs into the portal, the user can upload his file into the portal. The user can give the metadata about the file in the File Details textbox. The user can browse the file in the local system upon clicking the browse button and upload the desired file into the cloud.

4.2 Encrypt

The user can encrypt the file by clicking on the encrypt button. Once the file is encrypted, it is uploaded into the cloud by using the unique access token of the cloud account. The file encrypted and uploaded through the portal is to be stored in the cloud account. The cloud used is the Dropbox
cloud. The file name is encrypted with Triple DES Encryption algorithm. Figure 2. shows the file that is uploaded after encryption.Both file name and contents are encrypted.

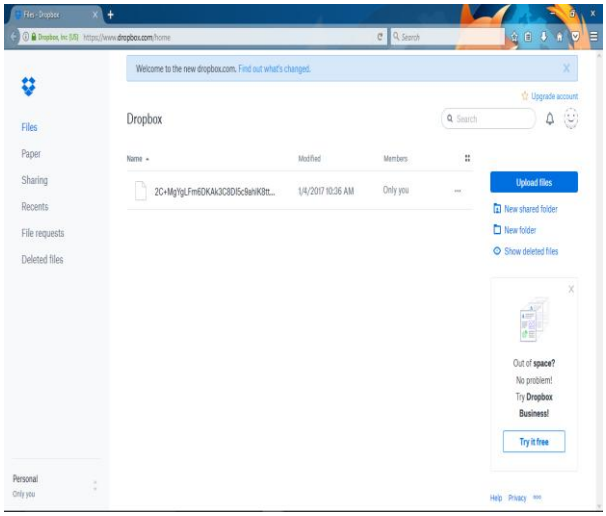

Figure 2: Cloud Account after uploading the file

4.3 Search

The user's search query is given in the File Details text box. The files related to the user's search query are displayed in a table along with a button to initiate the download of the file upon clicking the Search for Files button. The files related to the user's search query are displayed in a table along with a button to initiate the download of the file.

4.4 Decrypt

Upon clicking the download button the popup window ensuring download appears and upon clicking ok the corresponding file is being decrypted and downloaded to the user's local system. The contents of the decrypted file are the same as the original file thus ensuring full integrity.

4.5 Delete

The user can search for the file that he wants to delete. The user can give the search query in the File Details textbox. The list of related files are displayed in a table with a button to initiate deletion. Upon clicking the button the popup window that ensures deletion of the file appears on the screen. Upon clicking the ok button the file is deleted from the cloud. When the file is deleted from the cloud, the details of the file is also deleted from the portal. This eliminates the space complexity.

The comparison based on memory usage between Pailler and AES algorithm is given in the Figure 3.

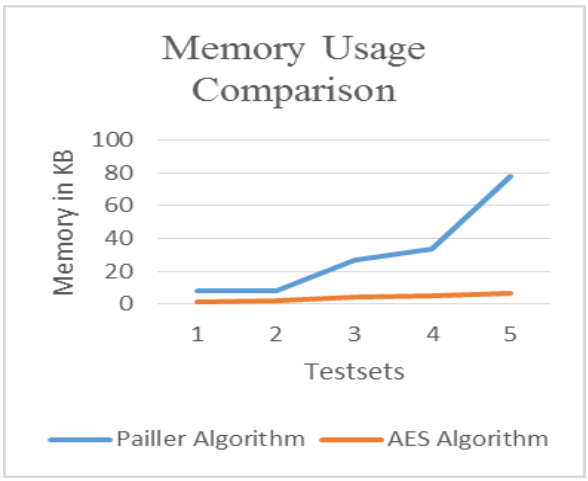

Figure 3: Comparison based on memory usage between Pailler and AES encryption algorithms

The comparison based on runtime between Pailler and AES algorithm is given in the Figure 4.

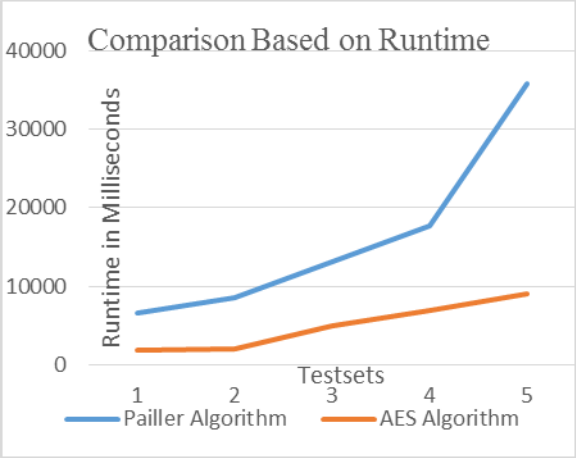

Figure 4: Comparison based on run time between Pailler and AES encryption algorithms

5. Conclusion and Future Work

A solution for the problem of insecurity of data in cloud that has curbed the progress of cloud computing movement has been proposed and implemented. The problem of insecurity of data has been resolved using the concept of Homomorphic Encryption. Computing search operation on encrypted data without compromising the privacy of data is achieved using Homomorphic Encryption. Security is enhanced by encrypting the file content and the file name with different encryption algorithms

The system considers uploading files to only one cloud provider namely DropBox. The work can be extended for multiple cloud providers. The user can be given the choice of choosing his desired cloud provider so that the files can be encrypted and uploaded.

6. Acknowledgement

The authors wish to thank the Management and Principal of Mepco Schlenk Engineering College, for their support in carrying out this research work.

7. References

[1] Sunanda Ravindran,Parsi Kalpana, 2013, 'Data Storage Security Using Partially Homomorphic Encryption in a Cloud', IJARCSSE, V3&4,V3I40357,pages 603-606.

[2] Payal V.Rarmar,Shraddha B.Padhar,Shafika N.Patel, April2014, 'Survey of various Homomorphic Encryption algorithms and schemes', IJCS(0975-8887) V91-No8, pages 26-32.

[3] MahaTebaa, SaidElHajji, AbdellatiELGhazi, 2012, 'Homomorphic Encryption Applied to the Cloud Computing Security', Proceedings of the World Congress on Engineering Vol I WCE London, U.K.

[4] Michael Brenner, Jan Wiebelitz, Gabriele von Voigt and Matthew Smith, 2011,'Secret Program Execution in the cloud applying Homomorphic Encryption' 5th IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2011)

[5] V.Biksham, D.Vasumathi, 2016, 'Querybased computations on encrypted data through homomorphic encryption in cloud computing security',International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT).

[6] Rongmao Chen, Yi Mu, Guomin Yang, Fuchun Guo, and Xiaofen Wang, April 2016,'Dual-Server Public-Key Encryption With Keyword Search for Secure Cloud Storage', IEEE Transactions on Information Forensics and Security, Vol 11, NO. 4.