TEST Engineering & Management

January - February 2020
ISSN: 0193 - 4120 Page No. 5179 - 5185

Cost Optimization Methods for Computational Time and Storage Space using Compression and Crypto-Graphical Techniques for a File in the Cloud

S. Sandhya Rani[1], Dr. Challa Narasimham[2]
[1] M. Tech, Vignan's Institute of Information Technology, India.
[2] Professor, Vignan's Institute of Information Technology, India.
sandhyaseeramsetty@gmail.com , drchallan@gmail.com

Article Info
Volume 82
Page Number: 5179 - 5185
Publication Issue:
January-February 2020

Article History
Article Received: *18 May 2019*
Revised: *14 July 2019*
Accepted: *22 December 2019*
Publication: *25 January 2020*

Abstract:
All the transactions are digitalized today and there is an extreme need for information security to be available in efficient and effective way. Security plays a vital role in this digital world. A secure connection must be established between the sender and receiver, this paper describes through Diffie-Hellman key exchange algorithm. Valuable data must be kept as safe as possible in order to stay away from attacks. This can be achieved through cryptography. Reducing the amount of space to store data we can enhance security in best and cheap way. Using various symmetric and asymmetric algorithms, data security can be enhanced. There are proven results that compared to symmetric algorithms, asymmetric algorithms are the best ones due to its advanced feature like private (secret) key. This paper narrated about how effectively the space and computational time can be obtained in a cloud. This is applicable in Business Organizations, Research centers, nuclear power plants, satellites, etc.

Keywords: Text file, Compression, Decompression, Encryption, Decryption, Cloud, time, space.

1. INTRODUCTION

Security is any action taken by an organization to avoid and monitor unauthorized access and future damage to private data, users or devices of the network. The safety objective is for all legitimate customers to maintain the network running secure. we need high security for businesses, hospitals, nuclear power plants, public agencies, satellites, etc. loss of security may lead to damage to our lives economically.

Data compression is a reduction in the number of bits required to represent data. Converting one form of data into another format definitely of reduced size is also termed as compression. Compressing data can save storage ability, speed up file transfer, and reduce storage hardware and network bandwidth expenses. Administrators spend less cash and less time on storage as a consequence of compression. It is a significant data reduction technique as data continues to evolve exponentially.

Network security can be enhanced through cryptography. cryptography is the method of converting the plain text into some random and irrelevant text and reversing the process again. Through encryption and decryption process, we can pass data from one location to the other in the network safely. Encryption is the method of converting plain text information into something seemingly random and irrelevant (cipher text). The method of turning cipher text back into plain text is decryption.

. Data can be compressed by performing two methods. They are: 1. Lossless data compression and 2. Lossy data compression. Lossless compression method is helpful for texts. Lossy compression technique is helpful for graphics, audio, video and pictures. Among these, lossless data compression method is best as there is no change of content (or)

January - February 2020
ISSN: 0193 - 4120 Page No. 5179 - 5185

original data meaning as we are considering text file. Data decompression is the technique of converting the reduced format of text to the original format of larger size. Data which is obtained after decompression and before decompression is same. In order to restore the original data, decompression method must be used along with compression.

Data compression and decompression can be achieved through various methods. They are Huffman coding, Shannon - Fano coding, LZ78, LZ77, Arithmetic coding, Sequitur, Prediction with partial match (ppm). As per the literature available, Huffman coding is playing vital role compression/decompression process.

Encryption and decryption are possible through symmetric cryptography and asymmetric cryptography. For encryption and decryption, symmetric encryption algorithms use a single key which is public key used for both encryption and decryption process. Various symmetric key encryption algorithms include Two fish, Serpent, AES(Rijndael), Blowfish, CAST5, RC4, DES, 3DES, Skipjack, Safer++, and IDEA. For encryption and decryption process, asymmetric encryption algorithms use mathematically-related key pair, one is the public key and the other is the private key. When using the public key for encryption, the associated private key is used for decryption and the associated public key is used for decryption while using the private key for encryption process. Asymmetric key cryptographic algorithms include ELGamal, RSA, Diffie-Hellman key exchange, Elliptic curve cryptography, PKCS. Among symmetric encryption and asymmetric encryption, asymmetric encryption algorithms provide more security because of its private(secret) key nature. So, in this paper we are using asymmetric encryption algorithms. How these algorithms works are explained below.

Whitfield Diffie and Martin Hellman presented the Diffie-Hellman key exchange to address the issue of safely determining a shared key between two communicating parties over an insecure network. Diffie Hellman cryptosystem, is a technique for exchanging cryptographic keys by first creating a shared secret key to be used for inter-communication purposes and not for encryption or decryption process. This key exchange method guarantees that the two sides who have no previous knowledge of each other jointly establish a shared secret key over the unsecure internet. Key transformations ate interchanged, and both end up with the same session key that looks like a secret key. Then each of them can calculate a third session key tat cannot readily be obtained from an intruder who understands both exchanged values.

RSA is most widely used asymmetric encryption algorithm in the world. RSA was first openly defined by Ron Rivest, Adi Shamir and Leonard Adleman of the Technology Institute of Massachusetts in 1977. It can be used to encrypt a message without having to exchange a secret key. Its security is based on the difficulty of factoring large integers that are result of two large prime numbers. Multiplying these two numbers is simple, but determining the initial prime numbers from the total (or) factoring is usually impractical relative to the time it would take even today's supercomputers.

Elliptical Cryptography (ECC) was found by Whitfield Diffie and Martin Hellman in 1976 and utilizes the issue known as the Discrete Logarithm Problem (DLP) as its asymmetrical procedure. It is a public key encryption method based on elliptical curve theory that can be used to produce quicker, lower, and more efficient cryptographic keys and is evolved from Diffie Hellman. ECC generates keys as the product of very large prime numbers, instead of traditional technique of generation, through the characteristics of the elliptic curve equation.

Cloud computing is the delivery of different services over the internet. These resources include tools and applications such as data storage servers, databases, networking, and software. In addition to maintaining files on a proprietary hard drive or local storage device, cloud – based storage makes it possible to save them to a remote database. To operate it, as long as an electronic device has an access to the internet, it has access to information and software programs. A file can be uploaded into the cloud by following these sequences of steps. In Google Cloud Platform Console, open the **Cloud Storage Browser**. In the list of buckets, click on the name of the bucket to upload an object. In the **Objects tab** for the bucket, either: Drag and drop the desired files from your desktop in the Google Cloud Platform console. Click on the **Upload Files** button, select the files you want to upload in the dialog that appears, and clock Open.

2.EXISTING SYSTEM

Working with cloud-based data storage, the end user data is first encrypted by using any cryptographic

January - February 2020
ISSN: 0193 - 4120 Page No. 5179 - 5185

algorithm and then stored on the cloud. Whenever the data is required, the end user simply places a request to the cloud service provider for accessing the data. The cloud service supplier authenticates the user first as the genuine user and then provides the data to the requester using any asymmetric algorithm.

The above process occurs over an insecure network. Attackers might attack and can theft (or) modify the data travelling over the network. There is a loss of confidentiality and integrity to the data. RSA, ECC asymmetric cryptographic algorithms are more secure as it was very difficult to find the secret key [13-18]. Attacker might unable to derive the plain text from the obtained cipher text without the secret key. As of now, third party cloud service providers are concentrating more on data safety rather than low storage space of data that would gradually decrease the installation cost for end users. The main key issues are: Time and Space. The amount of time required to store a file and the space occupied by the file onto the cloud should be as small as possible. Not all compression techniques are best. Based on the type of data, various compression techniques are used. To avoid data loses, Huffman coding has been considered for compression and decompression process. Based on the level of security of data, various cloud types are chosen. To overcome these limitations, the paper proposed to describe a new m3odel to design cost effective computational time and storage space.

3. DESIGN METHODOLOGY FOR COST EFFECTIVE TIME AND SPACE FOR CLOUD-BASED DATA

Secure and reliable connection establishment between the sender and receiver would result in safe data transfer over insecure network. Diffie-Hellman key exchange algorithm is widely used to establish a secure and reliable connection between the two authenticating parties. data compression will definitely decrease the storage space. Cloud service providers does not concentrate much on choosing of efficient algorithm. Choosing the best asymmetric algorithm for storing data on cloud will definitely reduce the amount paying to the cloud service provider by the end user.

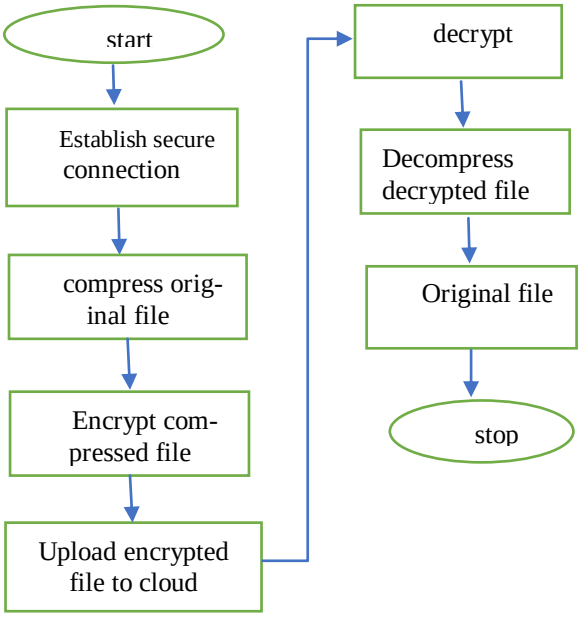

Fig 3.1: Design Flow chart

Design method 1:

A text file of size 128KB is chosen, which is compressed using Huffman compression technique. The compressed file size is obviously smaller than the original file size. Then the compressed file is encrypted using public key cryptographic algorithm i.e., RSA. The encrypted file of same size is then

January - February 2020
ISSN: 0193 - 4120 Page No. 5179 - 5185

uploaded to the cloud. As per the user request, the compressed encrypted file is retrieved back to the user system. On the user machine, the file is then decrypted using RSA decryption algorithm. Again, the decrypted file is decompressed using Huffman decompression technique. The resultant file then obtained is same as the file that was uploaded at the beginning. This is how, the space required to store a file onto the cloud is reduced. So, pay-as-you go policy is more efficient for customers who are wishing to store data safely on cloud with low cost.

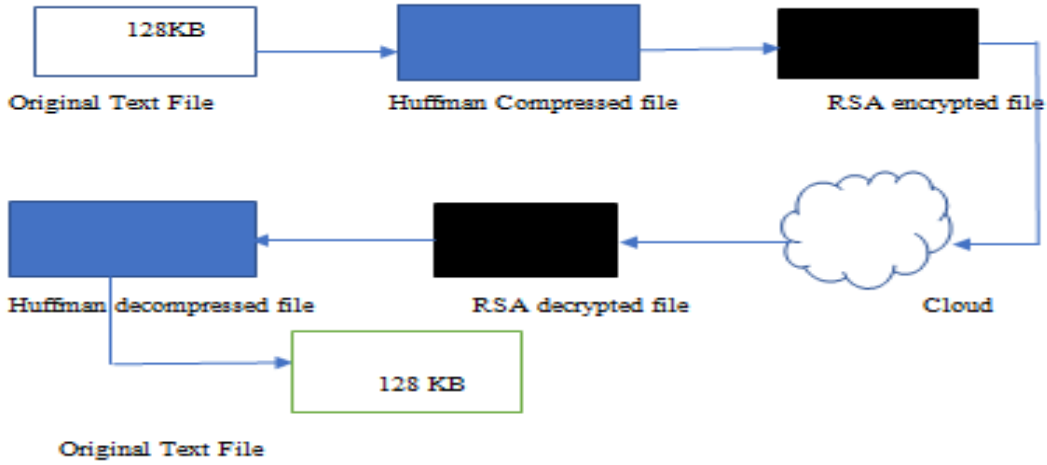

Fig 3.2 Design Model for RSA

Design Method 2:

A text file of size 128KB is chosen, which is compressed using Huffman compression technique. The compressed file size is obviously smaller than the original file size. Then the compressed file is encrypted using public key cryptographic algorithm i.e., ECC(Elliptic Curve Cryptography). The encrypted file of same size is then uploaded to the cloud. As per the user request, the compressed encrypted file is retrieved back to the user system. On the user machine, the file is then decrypted using ECC decryption algorithm. Again, the decrypted file is decompressed using Huffman decompression technique. The resultant file then obtained is same as the file that was uploaded at the beginning. This is how, the space required to store a file onto the cloud is reduced. So, pay-as-you go policy is more efficient for customers who are wishing to store data safely on cloud with low cost.

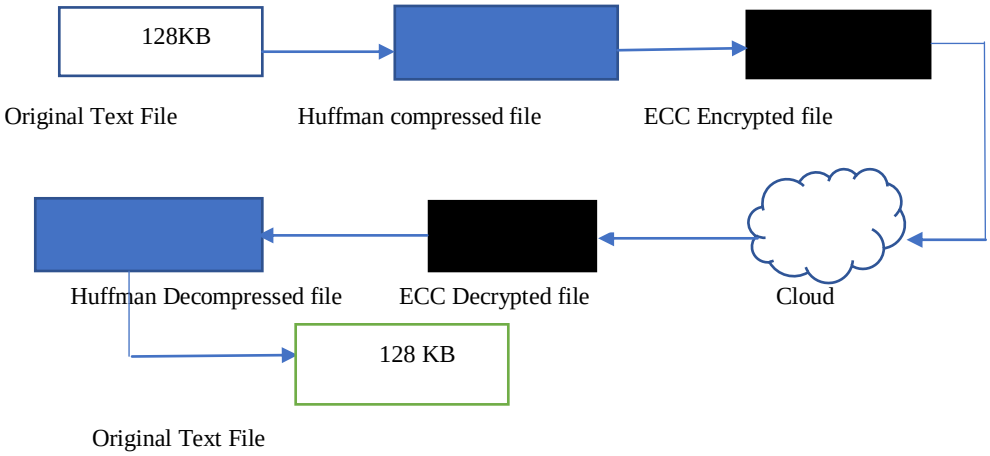

Fig. 3.3 Design model for ECC

In the above three cases, various compression and compression techniques were used for the same file of the same size. Performance metrics were calculated in each case and then compared to decide the best public-key cryptographic algorithm to store data on cloud efficiently and effectively.

TEST Engineering & Management

January - February 2020
ISSN: 0193 - 4120 Page No. 5179 - 5185

4.DESIGN METHODOLOGY FOR EFFECTIVE COMPUTATIONAL TIMINGS AND SPACE

In this paper, it is going to narrate and compare the compression and decompression times as well as encryption and decryption times for various asymmetric algorithms. Storage space of each outcoming file is also compared with the original file. The main key issues in the cryptography are time and space. To propose a best and effective technique, it must be time -effective and space-efficient.

Storage space before compression: The original file size is calculated. Using Huffman compression algorithm the storage space before compression is noted down.

$$\text{sts-bef-comp}_{\text{Huff}}$$

Storage space after compression: The file size after compression is calculated. Using Huffman compression algorithm the storage space before compression is noted down.

$$\text{sts-aftr-copm}_{\text{Huff}}$$

a. Using RSA cryptosystem - computational time

Encryption and decryption time: The amount of time required to encrypt a particular text file and the amount of time required to decrypt a particular text file using an RSA algorithm are calculated individually. encryption and decryption times and the total time for both the processes are noted down.

$$Tct_{RSA} = Et_{RSA} + Dt_{RSA}$$

Tct_{RSA} = total computational time for a file storing on cloud using RSA algorithm.

Et_{RSA} = encryption time after compression.

Dt_{RSA} = decryption time before compression.

b. Using RSA cryptosystem - Storage space

c. Using ECC cryptosystem - computational time

Encryption and decryption time: The amount of time required to encrypt a particular text file and the amount of time required to decrypt a particular text file using an ECC algorithm are calculated individually. Encryption and decryption times and the total time for both the processes are noted down.

$$Tct_{ECC} = Et_{ECC} + Dt_{ECC}$$

Tct_{ECC} = total computational time for a file storing on cloud using ECC algorithm.

Et_{ECC} = encryption time after compression.

Dt_{ECC} = decryption time before compression.

Based on the type of encryption algorithm used to encrypt the file, the encryption spaces may vary for various algorithms of same file size. So, for various algorithms, the total computational time for storing a file on cloud is calculated and noted down.

5.IMPLEMENTATION AND ANALYSIS

By computing encryption and decryption times, compression and decompression times and comparing those results would define which asymmetric algorithm is efficient for storing data on the cloud.

Huffman procedure:

a. Procedure to compute Ct_{Huf}:

```
long startTime = System.currentTimeMillis();
 if (choice.startsWith("c")) {
 startTime = compress();
  System.out.println("Compressing ...");
  long startTime = System.currentTimeMillis();
   tree.compress(charListFromFile(input), bitOut);
   long elapsed = System.currentTimeMillis() - startTime;
   System.out.println("\nDone. (" + elapsed + "ms)");
```

b. Procedure to compute Dct_{Huf}:

```
long startTime = System.currentTimeMillis();
else if (choice.startsWith("d"));
 startTime = decompress();
System.out.println("Decompressing " + inputFileName + " ...");
tree.decompress(bitIn, bitOut);
if (outputFileName.length() > 0) {
output.close();
long elapsed = System.currentTimeMillis() - startTime;
 System.out.println("\nDone. (" + elapsed + "ms)");
```

RSA procedure:

a. Procedure to compute Et_{RSA}:

```
   static private void doEncrypt(String[] args)
   long time1 = System.nanoTime();
```

TEST Engineering & Management

January - February 2020
ISSN: 0193 - 4120 Page No. 5179 - 5185

```
Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
cipher.init(Cipher.ENCRYPT_MODE, pvt);
processFile(cipher, inputFile, inputFile + ".enc");
long time2 = System.nanoTime();
double enctime = time2 - time1;
System.out.println("encryption time is "+enctime);
```

b. Procedure to compute Dt_{RSA}:

```
private static void doDecrypt(String[] args)
long time1 = System.nanoTime();
Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
cipher.init(Cipher.DECRYPT_MODE, pub);
processFile(cipher, inputFile, inputFile + ".ver");
long time2 = System.nanoTime();
double enctime = time2 - time1;
System.out.println("decryption time is "+dectime);
```

ECC procedure:

a. Procedure to compute Et_{ECC}:

```
Cipher aCipher = Cipher.getInstance(aSecretKey.getAlgorithm());
aCipher.init(Cipher.ENCRYPT_MODE, aSecretKey);
byte[] encText = aCipher.doFinal(getBytes());
System.out.println(Base64.encodeBase64String(encText));
System.out.println(encText);
```

b. Procedure to compute Dt_{ECC}:

```
Cipher aCipher = Cipher.getInstance(aSecretKey.getAlgorithm());
aCipher.init(Cipher.DECRYPT_MODE, aSecretKey);
byte[] decText = aCipher.doFinal(Base64.decodeBase64("0wwerdjkHbVhYI+YPxUnmw==".getBytes())); String text = new String(decText);
System.out.println("Decoded="+text);
```

Table 1: Time Characteristic Table (File Size :128 KB)

Algorithm	Encryption Time	Decryption Time
RSA	**4.132675E8**	**7.82966E7**
ECC	**11.26453E8**	**2.34528E8**

OBSERVATIONS : As compared to the above encryption and decryption times, RSA algorithm gives best results while performing encryption when compared to ECC algorithm. ECC algorithm gives best results while performing decryption when compared to RSA algorithm. So, RSA algorithm works well when the total computational time is calculated.

CONCLUSION:

This paper described how effectively the secure connection could be established between the two parties. The PKCS like RSA and ECC implementation for encryption and decryption process has been implemented and the computational timings for the corresponding techniques has been measured. Further, this paper utilized the best compression/decompression process implementation through Diffie Hellman and uploaded to the cloud.

REFERENCES:

[1] Leena Khanna, Anant Jaiswal, "Cloud Computing: Security Issues and Description of Encryption Based Algorithms to Overcome Them", IJARCSSE 2013

[2] G Devi, Pramod Kumar "Cloud Computing: A CRM service Based on a Separate Encryption and Decryption using Blowfish Algorithm" IJCTT 2012

[3] Simarjeet Kaur "Cryptography and Encryption in Cloud Computing", VSRD International Journal of CS and IT,2012

[4] Nelson Gonzalez, Charles Miers, Fernando Redigolo Marcos Simplicio, Tereza Carval-

January - February 2020
ISSN: 0193 - 4120 Page No. 5179 - 5185

ho, Mats Naslund, Makan Pourzandi "A quantitative analysis of current security concerns and solutions for cloud computing", Springer 2012.

[5] Wayne Jansen, Timothy Grance "Guidelines on Security and Privacy in public Cloud Computing", National Institute of Standards and Technology 2011

[6] Ayan Mahalanobis, "Diffie-Hellman Key Exchange Protocol, Its Generalization and Nilpotent Groups." 2005

[7] Ansah Jeelani Zargar, Mehreen Manzoor, Taha Mukhtar "Encryption/Decryption using Elliptical Curve Cryptography", IJARCSSE 2017

[8] Dr. E. Laxmi Lydia, K. Vijaya Kumar, P. Amaranatha Reddy, D. Ramya, "Text Mining with Hadoop: Document Clustering with TF-IDF and Measuring Distance Using Euclidean", Journal of advanced research in dynamical & control systems, Vol.10,14-Special Issue, 2018.

[9] Dr. E. Laxmi Lydia, B. Prasanna Kumar, D. Ramya, "Generation of dynamic energy management using data mining techniques basing on big data analytics issues in smart grids", International Journal of Engineering & Technology, 7(2.26), 2018, 85-89.

[10] E. Laxmi Lydia, P. Govindaswamy, SK. Lakshmanaprabu, D. Ramya, "Document Clustering based on Text mining K-means algorithm using Euclidean distance similarity", Journal of advanced research in dynamical & control systems, vol. 10, 02-special issue, 2018.

[11] E. Laxmi Lydia, D. Ramya, "Text Mining with Lucene and Hadoop: Document clustering with updated rules of NMF Non-Negative Matrix Factorization", International Journal of Pure and Applied Mathematics, vol 118(7), 191-198, 2018.

[12] E. Laxmi Lydia, P. Krishna Kumar, K. Shankar, S.K. Lakshmanaprabu, R. M. Vidhyavathi, Andino Maseleno, "Charismatic Document Clustering through novel K-means non-Negative Matrix Factorization (KNMF) algorithm using Key Phrase extraction", International Journal of parallel programming, 2018.

[13] Shankar, K. (2018). An optimal RSA encryption algorithm for secret images. International Journal of Pure and Applied Mathematics, 118(20), 2491-2500.

[14] Shankar, K., Devika, G., & Ilayaraja, M. (2017). Secure and efficient multi-secret image sharing scheme based on boolean operations and elliptic curve cryptography. International Journal of Pure and Applied Mathematics, 116(10), 293-300.

[15] Shankar, K., & Eswaran, P. (2016). RGB-based secure share creation in visual cryptography using optimal elliptic curve cryptography technique. Journal of Circuits, Systems and Computers, 25(11), 1650138.

[16] Elhoseny, M., & Shankar, K. (2019). Reliable data transmission model for mobile ad hoc network using signcryption technique. IEEE Transactions on Reliability.

[17] Shankar, K., Lakshmanaprabu, S. K., Gupta, D., Khanna, A., & de Albuquerque, V. H. C. (2018). Adaptive optimal multi key based encryption for digital image security. Concurrency and Computation: Practice and Experience, e5122.

[18] Shankar, K., & Ilayaraja, M. (2018, January). Secure Optimal k-NN on Encrypted Cloud Data using Homomorphic Encryption with Query Users. In 2018 International Conference on Computer Communication and Informatics (ICCCI) (pp. 1-7). IEEE.