Homomorphic Image Encryption in Cloud Computing

R.Kanagavalli^{#1}, Dr.Vagdevi S^{#2}

#1 Global Academy of Technology ,Bangalore,Karnataka,India,
kanagavalli2003.bali@ gmail.com

#2 DayanandSagar Academy of Technologyand
Management,Bangalore,vagdevi04@gmail.com

ABSTRACT

Cloud Computing has been envisioned as the next generation architecture of IT enterprise. In contrast to the traditional solutions where the solutions are under physical, logical and personnel controls, cloud computing moves the application of softwares and databases to the large data centers where the management of data and the services may not be fully trustworthy. In cloud storage not only text files are stored but also multimedia data like image, audio and video files are stored. More images stored means enhancing the security of the images from unauthorized use also becomes important. In this paper we have proposed a homomorphic encryption method for images stored in cloud. The method is based on mixed homomorphic operations applied to image files. The results obtained shows that the proposed work is efficient in terms of enhancing the security of the data stored during computations also.

Key words: Homomorphic Encryption; Image Security; Cloud Data Storage; Homomorphic Operations

Corresponding Author: R.Kanagavalli

INTRODUCTION

In cloud storage not only text files are stored but also multimedia data like image, audio and video files are stored. More images stored means enhancing the security of the images from unauthorized use also becomes important. Image cryptography aims at providing security to images stored based on three parameters namely diffusion, confusion and dependence on keys. An efficient image cryptosystem must have the following desirable properties 1. Computationally secure encryption algorithm. 2. Fast encryption and decryption process. 3. Flexible method. It is known that an image can be modeled as a product of an illumination component which is constant and a variable reflectance component. The image details are available in the reflectance component. Applying homomorphic encryption in the reflectance component is appropriate since it the most significant part of the image and encrypting illumination component will lead to redundancy. The approach is different from optical encryption since it can work directly on digital images. Also it very difficult to implement the digital operations like XOR with optical cryptosystems. The flow of the paper is as follows: Section II introduces Image homomorphic encryption and the existing methods followed by proposed scheme in section III and results in section IV. In section V, we have presented our conclusion of the study and its future scope.

RELATED WORK

The image processing without homomorphic encryption in cloud storage is as shown in Figure 1 and consists of the following steps:

1. The client uploads the image to the cloud.
2. When there is a processing request, the CSP will be using the original image as input for the processing and hence there is privacy disclosure.
3. The required processing will be done on the original image and the processed image will be sent to the user.
4. The user will be at the mercy of the CSP and will not be able to verify the result.

Fig 1: Image Processing without homomorphic encryption

The scenario in which the Image Processing with homomorphic encryption followed is as shown in Figure 2 and the steps are:

1. The client first encrypts the image using a homomorphic encryption method.
2. Then the image is uploaded to the cloud.
3. During processing, the CSP uses the evaluation function to do the required operation.
4. When queried the encrypted result image will be returned to the user.
5. The user gets the required processed image using decryption of the encrypted image.

Fig 2: Image Processing with homomorphic encryption

A general framework for the application of chaotic maps to provide image security was proposed by Fridich[1] in which chaotic map is discretized and then generalized. In [2] for image encryption, chaotic maps and symmetric encryption technique are used. The cipher map follows baker map principle. A generalized 3D chaotic map for image encryption was proposed by the authors in [3]. In this method, the position of the pixels is shuffled using 3D

cat maps. Another confusion chaotic map is created to enhance the security. In [4], the encryption was proposed using an external secret key and two chaotic logistic maps. The length of the secret key is 80 bits and eight operations are used to encrypt a pixel. In [5], the authors have used chaotic signals for image hiding. In [6], an image encryption scheme using Elliptic Curve Cryptography was proposed. Image is assumed to be in spatial domain and comparative linear congruential generator was used. In [7], ECC was used with magic matrix operations suitable for both compressed and uncompressed images. In [8] a method using probabilistic theory and pixel by pixel encryption technique was proposed. An homomorphic encryption technique was proposed in [9] for image scaling operation. The images are encoded and decoded using SEAL library. In [10] the authors have used secret key homomorphic encryption for encryption of image and the image processing techniques are translated to operate on floating point pixels of encrypted images. From the study of existing works it is clear that homomorphic encrypted images are a better choice of cloud operations. The study on existing literature also provides an insight that even though there are methods available for image encryption, only few methods are available using homomorphic encryption and this paved the way for the proposed work.

WORK DONE

An image can be modeled as a product of two components namely illumination and reflectance. The illumination component of the image is constant whereas the reflectance component is variable in nature. It is in reflectance component that all the image details are available. Therefore it is suitable to apply homomorphic encryption to reflectance component of an image rather than to illumination component. The approach is different from optical encryption since it can work directly on digital images. Also it very difficult to implement the digital operations like XOR with optical cryptosystems.

Any RGB image can be considered as a combination of three different images i.e. a Red scale image, a Green scale image and a Blue scale image which are stacked on each other as shown in Figure 3.

Fig 3: RGB Stack

An RGB image is represented as an array of color pixels. The array represents the three components of the image in the spatial domain at a particular instance. Therefore each pixel in the array has three constituent values. Hence the color of any pixel value is the combination of intensities of the RGB colors at the pixel's location in the array. If an image is represented by its size (width and height) say if m is the width of the image and n is the height of the image then the number of pixels in the image is given by $m \times n$. As stated earlier, a pixel can be seen as a combination of RGB color. Let I denote the image under study. Let C_{ij} denote the pixel in the i^{th} row and j^{th} column of the pixel matrix. Therefore each entry in the matrix can be written as

$$C_{ij} \leftrightarrow C_{ij}(c_0, c_1, c_2) \quad (1)$$

where c_0, c_1, c_2 represents the value of the Red, Green and Blue colors respectively.

Encryption Process:

The image encryption process consists of the following steps:

1. The image file F is read and let C be its corresponding pixel array .i.e. $C=\{C_{ij}\}$.
2. For each pixel in the array C , $C_{ij}(c_0, c_1, c_2)$, the values c_0, c_1, c_2 are sequentially encrypted to get the corresponding cipher values. Let d_0, d_1, d_2 represents the cipher pixel values of c_0, c_1 and c_2 respectively. Therefore in general

$$d_i \leftarrow En(K, w, c_i) \quad (2)$$

Where K is the key, w is the size of the column matrix and c_i is the color component.

3. After calculating all the pixel values, the cipher array $D=\{D_{ij}\}$ is generated.

Since each pixel is encrypted separately and each pixel having 3 integer values, the time complexity of the algorithm is $O(m.n.w.log w)$. The size of the encrypted array is $O(m.n.w)$. Each cipher array is w -dimensional column vector and hence D_{ij} is $w \times 3$ matrix.

Decryption Process:

The image decryption process is carried out as follows:

1. For each cipher text pixel D_{ij} in D , d_0, d_1, d_2 is decrypted to get c_0, c_1 and c_2 .

$$c_i \leftarrow Dec(K, w, d_i) \quad (3)$$

2. If $c_i < 0$, then set $c_i \leftarrow 0$.
3. If $c_i > 255$, then set $c_i \leftarrow 255$
4. Otherwise set $c_i \leftarrow c_i$
5. After calculating all C_{ij} output $C \leftarrow \{C_{ij}\}$ as the pixels array of the plain image.

The time complexity of the algorithm is $O(m.n.w.log w)$. Since the color of a pixel should be an integer in $[0, 255]$, each c_i will rounded into the nearest integer.

The methodology is tested with two commonly used image processing operations namely inserting watermark and resizing the image.

Inserting Watermark:

Inserting watermark to an image is the image processing operation which is done in order to enhance the security of the image from unauthorized access and use. The watermark is also an image which may be of same size or of different size as the parent image. The watermark image is embodied to the parent with a transparency level desired by the image's owner. The addition takes place in such a way that the watermark image's pixels are superimposed on the parent image's pixels with transparency criterion satisfied so that if any combatant tries to access the image, he will not be able to do so.

Let the parent image be I_1 and the watermark image be I_2 . Let t_l denote the desired transparency level. If C_{1ij} denote the pixel array of I_1 , C_{2ij} denote the pixel array of I_2 and C'_{ij} denotes the resultant image's pixel array, then

$$C'_{ij} = f(t_l, C_{1ij}, C_{2ij}) \cdot t_l \cdot C_{1ij} + (1 - t_l) \cdot C_{2ij} \quad (4)$$

When the client wants to add watermark which is also in encrypted form, the same can be achieved as follows:

$$\begin{aligned} C'_{ij} &= f(t_l, C_{1ij}, C_{2ij}) \\ &= En(w, t_l) * {}^{(w)}C_{1ij} + En(w, 1 - t_l) * {}^{(w)}C_{2ij} \end{aligned} \quad (5)$$

Image Resizing:

The resizing operation is done by using the quadratic linear interpolation method. Let I be the image of size $m \times n$ where m is the width and n is the height. If the resizing parameters are α and β where α represents width and β represents height, and if I' is the resized image then size of I' can be represented as $\alpha \cdot m \times \beta \cdot n$. It is also known that the image is represented by its pixel array of size $m \times n$ and each pixel C_{ij} in the array is composed of RGB components c_0, c_1 and c_2 . Therefore if C'_{ij} represents the pixel of I' then C'_{ij} is defined as

$$C'_{ij} = (c'_0, c'_1, c'_2) \text{ where } i \leftarrow 0, 1, 2, \dots, \alpha \cdot m - 1 \text{ and } j \leftarrow 0, 1, 2, \dots, \beta \cdot n - 1$$

Each C'_{ij} is dependent on four adjacent pixels namely $C_{p,q}, C_{p,q+1}, C_{p+1,q}, C_{p+1,q+1}$ where $p=i/\alpha$ and $q=j/\beta$. The operation can be written as

$$C'_{ij} = f(C_{p,q}, C_{p,q+1}, C_{p+1,q}, C_{p+1,q+1}, s, t) \quad (6)$$

$$= C_{p,q}(1-s) \cdot (1-t) + C_{p,q+1} \cdot s(1-t) + C_{p+1,q}(1-s) \cdot t + C_{p+1,q+1} \cdot s \cdot t \quad (7)$$

where $s = i/(\alpha \cdot m - p)$ and $t = j/(\beta \cdot n - q)$

When the client wants to resize the image that he has stored he can send the query to the server by providing the resize parameters α and β which can be achieved as :

$$C'_{ij} = f(C_{p,q}, C_{p,q+1}, C_{p+1,q}, C_{p+1,q+1}) \quad (8)$$

$$= C_{p,q} * {}^{(w)}u_0 + C_{p,q+1} * {}^{(w)}u_1 + C_{p+1,q} * {}^{(w)}u_2 + C_{p+1,q+1} * {}^{(w)}u_3 \quad (9)$$

where $u_0 = En(w, (1-s) \cdot (1-t))$

$$u_1 = En(w, s \cdot (1-t))$$

$$u_2 = En(w, (1-s) \cdot t)$$

$$u_3 = En(w, s \cdot t)$$

EXPERIMENTAL RESULTS

The proposed work is run on a system with 8GB RAM and 1 TB HDD with two processors. The method was compared with IGHE [11] method. The experiment was conducted by varying the floating point precision size n from 8, 16 and 32. The results of the experiment are as given below.

Run Time of Key Generation Algorithm:

An input image of size 4.4 KB(137 X 127 pixels) is considered and by varying the size of n , the running time is noted. The Run time is measured in microseconds and the results are tabulated as shown in Table 1 and the same is graphically shown in Figure 4.

Table1. Run Time of Key Generation Algorithm

Value of n	Run Time(in μs)
8	10
16	15
32	19

Fig 4: Run Time of Key Generation Algorithm

Run Time of Encryption Algorithm

An image of size 4.4 KB(137 X127 pixels) is given as input to the encryption algorithm and the value of n is varied as 8,16 and 32 and the time taken is noted . The obtained results are compared with the IGHE algorithm and the comparative results are tabulated in Table 2 and depicted as graph in Figure 5.

Table 2. Run Time of Encryption Algorithm

Value of n	Run Time(in μ s)	
	IGHE	IMHE
8	4	4
16	6	6
32	10	9

Fig 5: Run Time of Encryption Algorithm

Run Time of Decryption Algorithm

An encrypted image is given as input to the decryption algorithm and the value of n is varied as 8,16 and 32 and the time taken is noted . The obtained results are compared with the IGHE algorithm and the comparative results are tabulated in Table 3 and depicted as graph in Figure 6.

Table 3. Run Time of Decryption Algorithm

Value of n	Run Time(in μ s)	
	IGHE	IMHE
8	10	7
16	17	15
32	30	22

Fig 6: Run Time of Decryption Algorithm

Run Time of Image Addition Algorithm

The water mark insertion algorithm was run with input image of size 956 KB(882 X 638 pixels) and an encrypted watermark image of size 6.5 KB(97 X 40 pixels). The value of n is varied as 8,16 and 32 and the time taken is noted . The obtained results are compared with the IGHE algorithm and the comparative results are tabulated in Table 4 and depicted as graph in Figure 7.

Table 4. Run Time of Addition Algorithm

Value of n	Run Time(in μ s)	
	IGHE	IMHE
8	15	12
16	25	20
32	35	28

Fig 7: Run Time of Image Addition Algorithm

Run Time of Image Resizing Algorithm

The image resizing algorithm was run with input image of size 4.4KB(137 X 127 pixels) with a scaling ratio of 3:2. The value of n is varied as 8,16 and 32 and the time taken is noted . The obtained results are compared with the IGHE algorithm and the comparative results are tabulated in Table 5 and depicted as graph in Figure 8.

Table 5. Run Time of Resizing Algorithm

Value of n	Run Time(in μ s)	
	IGHE	IMHE
8	12	10
16	20	17
32	40	30

Fig 8: Run Time of Resizing Algorithm

From the results discussed above it is clear that the proposed method provides byte level homomorphic security for the image data stored and also ensures the confidentiality and integrity of data.

CONCLUSION

With the increase in use of cloud computing for personal and enterprise and with the advent in the technology, storage of data has enormously grown and providing security of those data is of paramount importance. With the digital era, more images are stored in the cloud and providing security to them is an interesting area. In this paper, we have proposed a method which uses homomorphic encryption operations. The common image processing operations like adding watermark and image resizing are applied to the images encrypted using the proposed method and the results are compared with the existing methods. The results show that the proposed method is more efficient and maintains the confidentiality and integrity of data.

REFERENCES

- I. J. Fridrich, Symmetric ciphers based on two-dimensional chaotic maps, *International Journal of Bifurcation and Chaos*, vol. 8, no. 6, pp. 1259–1284, 1998.
- II. G.Alvarez and S.Li.,Some basic cryptographic requirements for chaos based cryptosystems. *International Journal of Chaos Applications*, pp.2129-2151,2006.
- III. Guanrong Chen, Yaobin Mao and Charles K. Chui ,A symmetric image encryption scheme based on 3D chaotic cat maps, *Chaos,Solitons Fractals* 21(3),Elsevier,pp.749-761,2004.
- IV. N.K.Pareek, V.Patidar and K.K. Sud,Image Encryption using chaotic logistic map, *Image and Vision Computing*,Vol.24,no.9,pp.926-934,2006.
- V. Kamal Jadidy Aval, Morteza Sabery Kamarposhty,and Masumeh Damrudi , A Simple Method for Image Encryption Using Chaotic Logistic Map, *Journal of Computer Science and Computational Mathematics* ,Vol.3 ,no.3,pp.25-30,2013.
- VI. S. Maria Celestin Vigila and K.Muneeswaran , Nonce Based Elliptic Curve Cryptosystem for Text and Image Applications, *International Journal of Network Security*, Vol.14,no.4,pp.236-242,2012.
- VII. Srinivasan Nagaraj,K.Koteswara Rao, and G.S.V.P Raju, Image Encryption Using Elliptic Curve Cryptography and Matrix, *International Conference on Intelligent Computing, Communication & Convergence (ICCC-2015)*,Elsevier,pp.276-281,2015.
- VIII. A.M. Vengadapurvaja, G. Nisha, R. Aarthy, and N. Sasikaladevi, An Efficient Homomorphic Medical Image Encryption Algorithm For Cloud Storage Security, *7th International Conference on Advances in Computing & Communications, ICACC-2017* ,Elsevier,pp.643-650,2017.
- IX. William Fu,Raymond Lin,and Daniel Inge, Fully Homomorphic Image Processing,*arXiv:1810.03249v1 [cs.CR]* ,pp. 1-12,2018.
- X. J.B.Shen,X.G.Jin, and C.A Zhou, A color image encryption algorithm based on magic cube transformation and modular arithmetic operations, *Proceedings of Advances in Multimedia Information Processing*, Vol.3768,pp.270-280,2015.
- XI. Pan Yang,Xiaolin Gui,Jian An,and Feng Tian,An Efficient Secret Key Homomorphic Encryption Used in Image Processing Service, *Hindawi Security and Communication Networks*, Wiley, Volume 2017,Article ID 7695751,pp 1-11,2017.
- XII. Craig Gentry, A fully homomorphic encryption scheme”, *PhD thesis, Stanford University*,2009.

- XIII. Zia Brakerski, Craig Gentry, Vinod Vaikuntanathan, (Leveled) Fully homomorphic encryption without bootstrapping, *ITCS*, pp.309- 325, 2012.
- XIV. Gorti VNKV Subba Rao, Md. Sameeruddin Khan, Mr. A. Yashwanth Reddy and Mr. K. Narayana, Data Security in Bioinformatics, *IJARCCSE*, Vol.3, no.11, pp .590-598, 2013.
- XV. R. Kanagavalli and Vagdevi S, A Survey of Homomorphic Encryption Schemes in Cloud Data Storage. *IJRDET*, Vol 3, pp .71-75, 2014.
- XVI. R. Kanagavalli and Dr. Vagdevi S, A Mixed Homomorphic Encryption Scheme for Secure Data Storage in Cloud. *Proceedings of IEEE International Advanced Computing Conference IACC-2015*, D.O.I: 10.1109/IADCC.2015.7154867, 2015.
- XVII. R. Kanagavalli and Vagdevi S, Secured Data Storage in Cloud using Homomorphic Encryption, *International journal on Cloud Computing: Services and Architecture*, Vol.9, No.4, pp.1-11, 2019.
- XVIII. Kashif Munir, Dr. Sellappan, Framework for Secure Cloud Computing, *International Journal of Cloud Computing : Services and Architecture*, Vol .3, No.2, pp.21-35, 2013.
- XIX. C. Wang, Q. Wang, K. Ren, and W. Lou, Ensuring Data Storage Security in Cloud Computing, *Proc. 17th Int'l Workshop Quality of Service (IWQoS '09)*, pp. 1-9, 2009.