Analysis and Comparison of Various Fully Homomorphic Encryption Techniques

Pratibha Chaudhary
Amity University
Uttar Pradesh
pratibha.chaudhary4991@gmail.com

Ritu Gupta
Amity University
Uttar Pradesh
ritu4006@gmail.com

Abhilasha Singh
Amity University
Uttar Pradesh
abhilashasingh28@gmail.com

Pramathesh Majumder
Amity University
Uttar Pradesh
pramathesh@outlook.com

***Abstract*— Calculations can be carried out on encrypted form of data- is the essence of homomorphic encryption. Homomorphic encryption has resolved the security issues for storing data on the third-party systems (e.g. cloud or untrusted computer, service providers etc.). Most significant category of homomorphic encryption is fully homomorphic encryption. It permits unbounded number of operations on the encrypted form of data and output by system is within the ciphertext space. This paper provides the essentials of Homomorphic encryption and its various classifications i.e. partially homomorphic encryption, somewhat homomorphic encryption and fully homomorphic encryption. Mainly emphasizes on full homomorphic encryption and an investigation of different full homomorphic encryption schemes that uses the hardness of Ideal-lattice, integers, learning with error, elliptic curve cryptography based.**

Keywords*— homomorphic encryption (HE), partial homomorphic encryption (PHE), somewhat homomorphic encryption (SWHE), full homomorphic encryption (FHE), third-party, elliptic curve cryptography (ECC).

I. Introduction

In the world of IT, the data is generally stored and processed on third-party (cloud or untrusted computer, service providers etc.). Due to this, two major concerns arise. First, the legacy encryption schemes don't provide the security of data at third- party (i.e. stores unencrypted data). Third-party may misuse the data. Second, to perform any kind of operations on encrypted data first it is decrypted. These concerns may lead to the discovery of homomorphic encryption that permits performing calculations on data in encrypted form which is stored on third-party without decrypting it. The idea of performing computation over encrypted data called privacy homomorphism [1] came in 1978. In the same year 1978, RSA was also introduced by them. RSA [1] is asymmetric encryption scheme i.e. have open key (i.e. public) for encrypting and confidential key (i.e. secret) for decrypting known only to authorized user. RSA allows only multiplication operations on encrypted data. The homomorphic encryption scheme that allows both add and multiply unbounded operations on encrypted data called fully homomorphic encryption scheme. Fully homomorphic encryption scheme (FHE) was an open problem. Many researchers proposed their schemes that were El-Gamal scheme [2] but works for multiplication operations, Paillier scheme [3] but works for addition operations etc.

First fully homomorphic encryption scheme from somewhat homomorphic encryption scheme was given by Gentry [4], a noteworthy leap forward. This was based on the hardness ideal lattices [5].

Homomorphic encryption has numerous useful applications. For instance, Paillier employed in online voting system and threshold scheme [6], RSA is used to secure internet, Banking and credit card transaction [6], El-Gamal is used in hybrid systems, also used in secure multi-party computation [6] and so on.

II. Literature Survey

In this paper, we will investigate the previous researches done in the field of cryptography related to homomorphic encryption. Study and analyze the various homomorphic encryption schemes. Provide the insight of the best category of homomorphic encryption called fully homomorphic encryption and its various categories. Homomorphic encryption concept is taken from abstract algebra. Considering plaintext $y1, y2 \ldots yn$ and Encryption scheme E if it follows:

A. *Additive property:*

$$(y1 + y2 + y3 \ldots\ldots + yn) = (y1) + E(y2) + E(y3) \ldots + E(yn) \quad (1)$$

B. *Multiplicative property:*

$$(y1 * y2 * y3 \ldots * yn) = (y1) * E(y2) * E(y3) \ldots E(yn) \quad (2)$$

Then it is homomorphic.
For creating any kind of homomorphic encryption scheme, we use combination of addition and multiplication Boolean circuits (AND and XOR gate). Consider homomorphic encryption scheme for Asymmetric encryption is divided into four algorithms as follows:

$$H = (generating\ key, encrypt, evaluate, decrypt) \quad (3)$$

The running time of all these algorithms depends on security parameter λ.

a). Creating key Algorithm:

It takes security parameter λ as input i.e. number of bits in keys and output secret key s and public key p.

b). Encrypt Algorithm:

It takes plaintext $y1, y2 \ldots yn$ as input. It divides plaintext into bits (0,1) $b1, b2 \ldots bi$ and output ciphertext $c1, c2, \ldots$

ci.

$$c = Encrypt\,(p, bi) \tag{4}$$

c). Evaluate Algorithm:

It takes p and function fun as input and output new ciphertext $c1, c2 \ldots. cj$.

$$c = Evaluate(p, fun, ci) \tag{5}$$

d). Decrypt Algorithm:

Takes ciphertext $c1, c2. \ldots cj$ and s as input and output the plaintext $y1, y2 \ldots yn$.

$$yn = Decryp(s, cj) \tag{6}$$

The homomorphic encryption is categorized into three class as shown in Fig. 1 that are:

1. *Partially homomorphic encryption (PHE):*

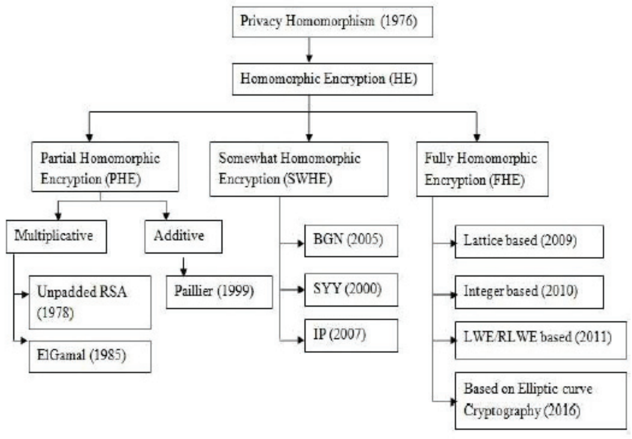
It allows either addition or multiplication operations i.e. only one kind of any number of operations. For example – Unpadded RSA, El-Gamal etc.

2. *Somewhat homomorphic encryption (SWHE):*

It allows add and multiply operations, but number of operations are limited. For example – BGN [7] etc.

3. *Fully homomorphic encryption (FHE):*

It allows arbitrary number of add and multiply operations. For example – lattice-based [5], integer based [10], learning with error based [14] etc.



Privacy Homomorphism (1976)
Homomorphic Encryption (HE)
Partial Homomorphic Encryption (PHE)
Somewhat Homomorphic Encryption (SWHE)
Fully Homomorphic Encryption (FHE)
Multiplicative
Additive
Unpadded RSA (1978)
ElGamal (1985)
Paillier (1999)
BGN (2005)
SYY (2000)
IP (2007)
Lattice based (2009)
Integer based (2010)
LWE/RLWE based (2011)
Based on Elliptic curve Cryptography (2016)

Fig. 1. Classification of Homomorphic encryption schemes

III. FULLY HOMOMORPHIC ENCRYPTION SCHEMES

Fully Homomorphic Encryption is a type of cryptosystem that encrypts the ciphertext arbitrarily. It allows the programs to run on any type of functionality that supports the encryption system.

A. *FHE scheme formed using Ideal Lattice (2009)*

Bootstrappable encryption scheme is created using ideal lattices [5] because:

- Decryption circuit complexity of algorithms is very low in typical lattice-based than the schemes like unpadded RSA or ElGamal.
- Addition and multiplication property of homomorphism is provided by ideal lattices.

Gentry's construction involves three steps that are:

- Construction of somewhat homomorphic encryption: Constructing to evaluate low-degree polynomials for a limited number of ADD and MULT operations. A ciphertext is of form

$$c = u + n \tag{7}$$

Where u is vector in the ideal lattice [5] and n is an "error" or "offset" or "noise" vector attached with plaintext m. The error component is increasing by applying each homomorphic operation it reaches a value that is inevitable which at a certain point does not allow ciphertext decryption correctly anymore.

- Squashing: Squashing is used to minimize the depth of decryption circuit as it increases to higher value. SWHE scheme Decryption circuit works well with small depth.
- Bootstrapping: Bootstrapping is a process of refreshing the ciphertext to reduce noise. If a somewhat homomorphic encryption scheme can assess itself (augmented) decrypting circuit, then it is said to be bootstrappable.

B. *Fully Homomorphic Encryption without using Bootstrapping:*

In 2010, Gentry et. al. proposed the idea of fully homomorphic encryption schemes using bootstrapping mechanisms. This allowed to use any number of additions to the original data but with only single multiplication.

In 2013, Wei et. al. presented the idea that allowed any number of additions and multiple number of multiplications on the original data.

Symmetric Learning based Fully Homomorphic Encryption scheme:

Later, Symmetric learning mechanism based fully homomorphic encryption scheme was proposed that was mainly based on machine learning techniques to learn about the encryption pattern. This method excluded the use of bootstrapping and used machine learning approach instead. This technique used dimension-reduction mechanisms which allowed shortening the original text and reduced the complexity of decryption mechanisms which doesn't affect the additional assumptions.

Ring Fully Homomorphic Encryption schemes:

Following SFHE scheme, a new FHE scheme was developed – RFHE (Ring Fully Homomorphic Encryption). This scheme didn't used noise reduction mechanisms and concerned a non-octonion ring over the noise reduction.

Later, Liu proposed an idea of symmetric fully homomorphic encryption scheme extending the RFHE scheme, which was depended on non-commutative ring fields. This idea also excluded the possibility of bootstrapping and allowed a greater number of additions and multiplications on the original data. It didn't used noise reduction and was dependent on approximate-GCD mechanisms.

C. *Implementation of Fully homomorphic encryption scheme (2011)*

For implementing fully homomorphic encryption, Gentry and Halevi [8] used variant similar to Smart and Vercauteren [9]. Smart and Vercauteren were unable to implement bootstrapping functionality when converting somewhat homomorphic encryption to fully homomorphic encryption. The optimization includes bootstrapping functionality; a batching technique for encryption etc is done in all aspects to implement fully homomorphic encryption. In Somewhat homomorphic encryption, key generation does not require full polynomial inversion [8].

TABLE I. SHOWS LATTICE DIMENSIONS AND SETTING IN DIMENSIONS

Lattice dimensions	Setting in dimension (Size)
Toy	$512 = 2^9$
Small	$2048 = 2^{11}$
Medium	$8192 = 2^{13}$
Large	$32768 = 2^{15}$

Table 1 above shows the lattice with several dimensions taken into consideration for implementing FHE.

TABLE II. PUBLIC KEY SIZE RANGES

Lattice dimensions	Public key size	One Bootstrapping operation runtime
Small	70 Megabytes	30 seconds
Large	2.3 Gigabytes	30 minutes

Table 2 above shows the public key size ranges and one bootstrapping operation runtime used for small and large lattice dimensions. The One Bootstrapping operation runtime of 1-CPU 64-bit machine with large memory [8] is considered.

D. *FHE scheme formed using integers (2010)*

Van Dijk et al [10] came up with fully homomorphic encryption scheme using Gentry's construction [5] but defined over integer instead of ideal lattices because they are conceptually simpler to show that thecomplicated fully homomorphic encryption scheme can be accomplished using "basic" techniques. The scheme used hardness of the approximate integer greatest common divisors (approximate GCD) problem [11] was analyzed by Howgrave-Graham.

General construction:

- Creating key Algorithm: An odd integer key , $k \in [2n-1, 2n)$.
- Encrypting Algorithm$(k,\ x\ )$: For encrypting a bit $x \in \ \{0,\ 1\}$, set ciphertext an integer value whose residue mod p has the similar parity as the plaintext. Like set $c = k * q + 2r + x$, where the integers q, r are chosen at random such that $2r$ is less than $k/2$ in absolute value.
- Decrypting Algorithm $(k,\ c)$: $x = Output\ (c\ mod\ k)\ mod\ 2$.
- Constraint: $x + 2r < k/2$ and choose r: $r \approx 2\sqrt{n}\ and\ q \approx 2n\hat{}3$

Homomorphism over addition:

$$(x1) + E(x2) = x1 + 2r1 + kq1 + x2 + 2r2 + kq2 = (x1 + x2) + 2(r1 + r2) + (q1 + q2)$$

$Nois = (x1 + x2) + 2(r1 + r2)$, it grows linearly as shown in Fig. 2(a).

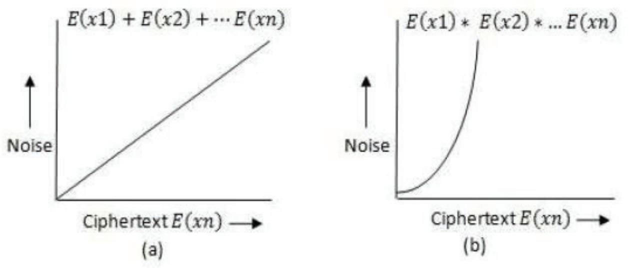


Fig. 2. Noise expansion with encryption

Homomorphism over Multiplication:

$$(x1) * E(x2) = (x1 + 2\ r1 + kq1)(\ x2 + 2\ r2 + kq2) = x1\ x2 + 2(x1r2 + x2r1 + 2\ r1\ r2) + (kq1\ q2 + 3q1r2 + x1q2 + x2q1)k \quad (9)$$

$Nois = x1\ x2 + 2(x1r2 + x2r1 + 2\ r1\ r2)$, it grows exponentially as shown in Fig.2 (b).

This is a symmetric scheme, but easily converted into asymmetric scheme. The public key is formed using "encryptions of zero", particularly integers

$$p = qi * k + 2ri \quad (10)$$

Where $qi,\ ri$ are described above, k is private key. Same decryption is used for decrypting the ciphertext. As no efficient algorithm here exists, for recover k from the given pis in polynomial time, therefore the scheme is considered as secure.

E. *FHE scheme formed using learning with error (2011)*

Learning with Error (LWE) is one of the hardest problems to solve in practical time for even post-quantum algorithms as mentioned [12]. Regev had first introduced "learning from parity with error" problem [13] that reduced worst case hardness of lattice problems like shortest vector problem (SVP) [14] to learning with error (LWE) problem. Brakerski and Vaikuntanathan came up with the scheme which uses ring learning with error (RLWE) [14]. RLWE used polynomial- LWE (PLWE). For understanding hardness assumption of RLWE, for instance, take a polynomial samples over a ring having form $(ri, rik + yi)$, where k is a random "secret ring element", $ri's$ are uniformly random in the ring, and yi are "small" elements of ring, an eavesdropper cannot distinguish random pairs of ring elements in this sequence of samples [14]. This is further decreased to worst case problem of SVP. This FHE scheme was also constructed from SWHE using Gentry's bootstrapping and squashing techniques [5]. To avoid squashing in this scheme, decrease to worst case hard problems at the cost of depending on a "sparse" version of LWE [14]. The alternate of squashing is re-linearization. Re- linearization reduced the ciphertext size. No generation of lattice basis required for generating a key. The RLWE scheme is circular secure that means it permits to safely encrypt its self-secret key i.e. ring element. RLWE used for practical applications more than LWE as it is more efficient. RLWE is circular secure (key-dependent security) with respect to linear functions of secret key.

F. FHE formed using elliptic curve cryptography (2016)

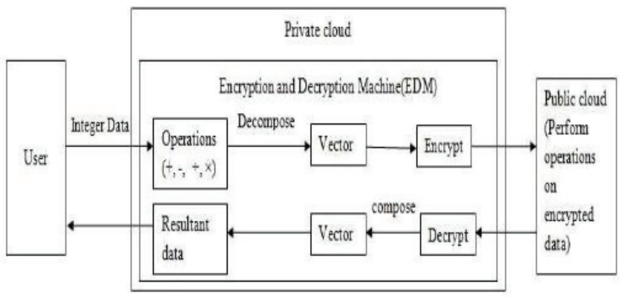


Fig. 3. Homomorphic encryption using elliptic curve cryptography

Figure 3 above shows the step involved in homomorphic encryption using elliptic curve cryptography. User wants to process data, first it goes to private cloud. Private cloud has the Encryption and Decryption Machine (EDM) that is responsible for encrypting and decrypting the data. Public cloud performs the computational operations on the data in encrypted form and returns the obtained result backward to the EDM. EDM will provide the required resultant data back to user.

IV. RESULTS

Our goal is to achieve a practically feasible FHE scheme that is unrestricted. One example of practical implementation for FHE schemes is provided as follows:

1. Lattice- based scheme is implemented on IBM system ×3500 server [8] and 6 Intel xeon 2.4 GHz processor [16] using language C++ and NTL library.
2. Integer-based scheme is implemented on SAGE software on Intel core i5, 3.30 GHz x4, 8 Gb RAM [17].
3. RLWE is implemented in HELib using C++ language and a mathematical library NTL [18].
4. Elliptic curve cryptography based HE is empirically evaluated and test on GPS data [15].

TABLE III: COMPARISON OF FULLY HOMOMORPHIC ENCRYPTION SCHEME

Property	**Ideal lattice based**	**Integer based**	**Ring learning with error**	**Elliptic curve cryptography based**
Introduced in year	2009	2010	2011	2016
Hardness	Closest vector problem, sparse subset sum problem	Approximate GCD problem	Rings learning with errors	Discrete logarithm of an arbitrary elliptic curve component concerning to known base point is impractical
Security	More	Less	More than Ideal based	Most
Simplicity	Less	Most	Simpler than Ideal based	More
Limitations	Public key size is too large	Reduced security	NA	Does not support floating point calculation
Efficiency	Efficient	Least efficient	More efficient than ideal lattice based	Most efficient

After critically studying and analyzing the above mentioned fully homomorphic encryption schemes, the comparison is done.

Some properties like hardness, introduced year, security of scheme, efficiency etc. are identified while comparing and the result of comparison are listed in Table 1.

V. CONCLUSION AND FUTURE WORK

Homomorphic encryption existed from 30 years still is open problem for research. Various Homomorphic encryption schemes came have a scope of improvement.

The improvement can be done in any aspect like improving generation of key or encryption, decryption etc. The implementation cost is very high, and complexity of homomorphic encryption scheme is also high. It can be reduced in future. In this paper, we have classified Homomorphic encryption scheme and mainly focused on various types of FHE. FHE schemes formed using lattice, integer, ring learning with error and elliptic curve cryptography based is implemented in real-life applications. The example of each is provided above. FHE is not practically implemented on every platform. The mentioned FHE schemes supports only single user setting notwithstanding with applications that take multiple data from different users [19].

REFERENCES

[1] Rivest, R. L., Adleman, L., & Dertouzos, M. L. (1978). On data banks and privacy homomorphisms. Foundations of secure computation, 4(11), 169-180.

[2] ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE transactions on information theory, 31(4), 469-472.

[3] Paillier, P. (1999, May). Public-key cryptosystems based on composite degree residuosity classes. In Eurocrypt (Vol. 99, pp. 223-238).

[4] Gentry, C. (2009). A fully homomorphic encryption scheme. Stanford University.

[5] Gentry, C. (2009, May). Fully homomorphic encryption using ideal lattices. In STOC (Vol. 9, No. 2009, pp. 169-178).

[6] Parmar, P. V., Padhar, S. B., Patel, S. N., Bhatt, N. I., & Jhaveri, R. H. (2014). Survey of various homomorphic encryption algorithms and schemes. International Journal of Computer Applications, 91(8).

[7] Boneh, D., Goh, E. J., & Nissim, K. (2005, February). Evaluating 2-DNF Formulas on Ciphertexts. In TCC (Vol. 3378, pp. 325-341).

[8] Gentry, C., & Halevi, S. (2011, May). Implementing Gentry's Fully-Homomorphic Encryption Scheme. In EUROCRYPT (Vol. 6632, pp. 129-148).

[9] Smart, N. P., & Vercauteren, F. (2010, May). Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes. In Public Key Cryptography (Vol. 6056, pp. 420-443).

[10] Van Dijk, M., Gentry, C., Halevi, S., & Vaikuntanathan, V. (2010, May). Fully homomorphic encryption over the integers. In Annual International Conference on the Theory and Applications of Cryptographic Techniques (pp. 24-43). Springer Berlin Heidelberg.

[11] Howgrave-Graham, N. (2001, March). Approximate integer common divisors. In CaLC (Vol. 1, pp. 51-66).

[12] Acar, A., Aksu, H., Uluagac, A. S., & Conti, M. (2017). A Survey on Homomorphic Encryption Schemes: Theory and Implementation. arXiv preprint arXiv:1704.03578.

[13] Regev, O. (2009). On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM (JACM), 56(6), 34.

[14] Brakerski, Z., & Vaikuntanathan, V. (2011, August). Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Annual cryptology conference (pp. 505-524). Springer, Berlin, Heidelberg.

[15] Hong, M. Q., Wang, P. Y., & Zhao, W. B. (2016, April). Homomorphic Encryption Scheme Based on Elliptic Curve Cryptography for Privacy Protection of Cloud Computing. In Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), 2016 IEEE 2nd International Conference on (pp. 152-157). IEEE.

[8] Smart, N. P., & Vercauteren, F. (2014). Fully homomorphic SIMD operations. Designs, codes and cryptography, 1-25.

[9] Gerasimov, A. N., Epishkina, A. V., & Kogos, K. G. (2017, February). Research of homomorphic encryption algorithms over integers. In Young Researchers in Electrical and Electronic Engineering (EIConRus), 2017 IEEE Conference of Russian (pp. 398-403). IEEE.

[10] Halevi, S., & Shoup, V. (2013). Design and implementation of a homomorphic-encryption library. IBM Research (Manuscript), 6, 12-15.

[11] Tourky, D., ElKawkagy, M., & Keshk, A. (2016, October). Homomorphic encryption the "Holy Grail" of cryptography. In Computer and Communications (ICCC), 2016 2nd IEEE International Conference on (pp. 196-201). IEEE.

Distributed Image Encryption Based On a Homomorphic Cryptographic Approach

Mamadou I. Wade
Dept. of Electrical Eng. and Comp. Sc.
Howard University
Washington, DC, USA
mamadou.wade@bison.howard.edu

Mohamed Chouikha
Dept. of Electrical Eng. and Comp. Sc.
Howard University
Washington, DC, USA

Tepper Gill
Dept. of Electrical Eng. and Comp. Sc.
Howard University
Washington, DC, USA
tgill@howard.edu

Wayne Patterson
Dept. of Electrical Eng. and Comp. Sc.
Howard University
Washington,DC, USA

Talitha M. Washington
dept. of Mathematics
Howard University
Washington, DC, USA

Jianchao Zeng
Senior Standards Advisor
Food and Drug Administration (FDA)
Silver Spring, MD, USA

Abstract— **The objective of this research is to develop a novel image encryption method that can be used to considerably increase the security of encrypted images. To solve this image security problem, we propose a distributed homomorphic image encryption scheme where the images of interest are those in the visible electromagnetic spectrum. In our encryption phase, a red green blue (RGB) image is first separated into its constituent channel images, and then the numerical intensity value of a pixel from each channel is written as a sum of smaller pixel intensity sub-values, leading to having several component images for each of the R, G, and B-channel images. A homomorphic encryption function is used to separately encrypted each of the pixel intensity sub-values in each component image using an encryption key, leading to a distributed image encryption approach. Each of the encrypted component images can be compressed before transmission and/or storage. In our decryption phase, each encrypted component image is decompressed if necessary, and then the homomorphic property of the encryption function is used to transform the product of individually encrypted pixel intensity sub-values in each encrypted component images, to the encryption of their sum, before applying the corresponding decryption function with a decryption key to recover the original pixel's intensity values for each channel image, and then recovering the original RGB image. Furthermore, a special case of an RGB image encryption and decryption where a pixel's intensity value from each channel is written as a sum of only two sub-values is implemented and simulated with a software. The resulting cipher-images are subject to a range of security tests and analyses. Results from these tests shown that our proposed homomorphic image encryption scheme is robust and can resist security attacks, as well as increases the security of the associated encrypted images. Our proposed homomorphic image encryption scheme has produced highly secure encrypted images.**

Index Terms*— *Distributed Image Encryption, Homomorphic Encryption, Image Encryption, Paillier Cryptographic System, RGB Image Encryption.

I. Introduction

This research addresses information security problems also known as Cyber Security in general, and particularly image security. The security of encrypted images can always be improved through new encryption approaches and methods. Therefore, new encryption schemes that can efficiently protect information and counter any malicious Cyber behaviors are constantly researched. Our goal is to develop a novel homomorphic image encryption scheme that can be used to encrypt images before transmitting them through unsecured channels without compromising their contents, and then recover the encrypted images using a decryption process. The encryption scheme should also protect images when stored in computer servers or files. The application domains for our proposed homomorphic image encryption scheme include confidential images from satellites, military application images, industrial application images, some type of medical images, fingerprint images, and other images in the visible electromagnetic spectrum from any areas where there is a need to protect against security breach and to ensure their confidentiality and integrity. Image encryption schemes have been developed and presented in the literature for many years by various researchers from academia, industry, and other areas. Among these image encryption schemes, one can list chaos based encryptions, for some cases, use sequences or system of equations with chaotic behavior when encrypting a pixel intensity value from an image. The Chaos-based approach has been proposed by many researchers [4], [9], [15], [16], [23], [25]. For instance, Z. H. Guan, F. Huang, and W. Guan [25] proposed a Chaos-based image encryption algorithm where the position of the pixels in the special-domain is shuffled using the Arnold cat map, while each a pixel intensity value is changes using Chen's chaotic system. R. Tao, X. Meng, and Y. Wang in [17] have proposed an image encryption scheme based on Multiorders of Fractional Fourier Transforms (FRFT) from which they

978-1-7281-3885-5/19/$31.00 ©2019 IEEE

obtained an encrypted image by the summation of different orders inverse discrete FRFT of an image that is interpolated. In [12], L. D. Singh and K. M. Singh discussed an Elliptic Curve Cryptosystem in which an image encryption scheme is applied to a group of pixels to obtain a corresponding cipher-image. G. Ye and X. Huang proposed in [6] an image encryption scheme that uses an electrocardiography (ECG) signal to generate the initial encryption key used to encrypt a plain-image. In [11], J. Zhou, X. Liu, O. C. Au, and Y. Yan Tang proposed an efficient image encryption-then-compressed (ETC) system which operates in the prediction error domain and provides a high security level for its encrypted and compressed images.

A paradigm shift is proposed in this research where one plain-image is decomposed into multiple component images that are each encrypted to obtain multiple component cipher-images using a homomorphic function properties. The encrypted component cipher-images are then decrypted and combined to obtain the original image.

The organization of the paper is as follows: In Section II, the proposed image cryptographic scheme is presented. Section III provides the performance and security analysis results, while Section IV is the conclusion of the paper.

II. Proposed Image Cryptographic Scheme

Consider the following two propositions and associated encryption and decryption schemes [14].

Proposition 1

Let $g(i,j)$ be a 2-dimensional array representation of an image with M rows and N columns, where (i,j) is the spatial coordinate of each pixel, for $i=1,2,3,...,M$ and $J=1,2,3,...,N$. Let the image's data class be unsigned 8-bit integer, leading to having each pixel intensity value in the interval $[0,(L-1)]=[0,255]$, where $L=256$ is the number of pixel intensity levels. Let each pixel's intensity values y belongs to the finite Galois Field $Z_p=\{0,1,2,\dots(p-1)\}$, where p is a prime number chosen to be equal to $p=257$. For an 8-bit image, each pixel's intensity value y is in the interval $[0,255]$, but we chose $p=257$, the closet prime number to 255. This could lead to having a pixel intensity value of 256, which is out of the range $[0,255]$. So, if it happens that $y=256$, one can perform special processing to account for it, or map it $y=255$, which will have very little effect on the actual image because of redundancy or other factors. Note that the same concepts can be applied to images with data class unsigned 16-bit integer or others.

Proposition 2

Let E be a homomorphic Encryption function mapping from the finite field Z_p such that $E(y_1+y_2+y_3+\dots+y_k)=E(y_1)\times E(y_2)\times E(y_3)\dots E(y_k)$, for all y_k in Z_p, and k is a positive integer such that $1<k<L$, where L is the number of pixel's intensity levels. This means that the encryption of a sum of k pixel intensity sub-values y_1, y_2, $y_3,\dots,y_k$ is equal to the product of their individual encryption, and vice versa.

A. *Homomorphic Image Encryption*

Let E be a homomorphic encryption function, and let y be the intensity value of a pixel in image $g(i,j)$ where $i=1,2,3,\dots M$ and $j=1,2,3,\dots,N$, where M and N are the number of rows and columns of pixels in the digital image, respectively. One can write a pixel's intensity value y as a sum of k pixels' intensity sub-values as shown:

$$y=y_1+y_2+y_3+\dots+y_k=\sum_{n=1}^{k}y_n, \tag{1}$$

where the number of pixel components k, also called the number of pixel intensity sub-values, which also corresponds to the number of component images, is an integer such that $1<k<L$ where L is the number of pixel intensity levels.

When the number of pixel intensity sub-values is greater than the pixel intensity value, meaning that $k>y$, extra special processing is needed, where the difference $d=k-y$ can be used to find the original pixel value y. Now, to encrypt a pixel's intensity value y using the homomorphic Encryption function E, one can write:

$$E(y)=E(y_1+y_2+y_3+\dots+y_k) \tag{2a}$$

$$=E\left(\sum_{n=1}^{k}y_n\right) \tag{2b}$$

$$=\Pi_{n=1}^{k}\left(E(y_n)\right) \tag{2c}$$

$$E(y)=E(y_1)\times E(y_2)\times E(y_3)\dots E(y_k) \tag{2d}$$

The final expression of $E(y)$ in (2)d has a profound implications. One can perform distributed and/or parallel or sequential encryption processing of each $E(y_k)$ simultaneously, or at different time using the same or different encryptions keys. Each $E(y_k)$ can also be computed by the same or different processors at the same or different locations. This can greatly increase the security of the encrypted image because an intruder may not have access to all $E(y_k)$ that can be stored at different locations, or transmitted at different time intervals. Also, if different encryption keys are used for each $E(y_k)$, opponents who have access to some of the decryption keys may not have access to other decryption keys, leading to not being able to decipher all corresponding encrypted component images without all the decryption keys. Also, each y_k can be randomly generated, the only requirement is that their sum should be equal to y. It is also important to note that the larger the value of k, the more secure the encrypted image is, but also the higher the computational cost.

In addition, each of the encrypted values $E(y_k)$ could be a very large integer, out of the range $[0,(L-1)]$, of the associated image's pixel intensity values. So, to make these $E(y_k)$ meaningful from an image point of view, one can apply $(\mod p)$ to each of the encrypted values $E(y_k)$, to map them back to Z_p, and obtain pixels' intensity values within the range $[0,(p-1)]$ that can be meaningful from an image point of view. For instance the pixel intensity values range is

$[0,255]$ for the case of an 8-bit image, and p can be chosen to be $p=257$. So, we can write:

$$C_1 = E(y_1) \tag{3}$$

$$C_2 = E(y_2) \tag{4}$$

$$\vdots$$

$$C_k = E(y_k) \tag{5}$$

Applying $\mod p$ to the above equations, we have

$$C_{p1} = C_1 \mod p = E(y_1) \mod p \tag{6}$$

$$C_{p2} = C_2 \mod p = E(y_2) \mod p \tag{7}$$

$$\vdots$$

$$C_{pk} = C_k \mod p = E(y_k) \mod p \tag{8}$$

The quantities C_{p1}, C_{p2} … ,C_{pk} represent the encrypted values for each of the pixel's intensity sub-values y_1, y_2, … ,y_{pk}. They also represent the secure image pixel's intensity sub-values that will be transmitted or stored.
It is also important to note another quantity needed for the decryption. It is the greatest integer less than or equal to $(E(y_k)/p)$ also known as the floor of $(E(y_k)/p)$ or $\lfloor(E(y_k)/p\rfloor$. It also represents the quotient when $E(y_k)$ is divided by p. This quantity is not secrete but can also be encrypted by other means and transmitted at the transmitter side to increase security, or it can be computed at the receiver side. Without $\lfloor(E(y_k)/p\rfloor$ reconstruction of $E(y_k)$ for decryption purposes at the receiver side may be difficult.
So, we can write:

$$qt_1 = \lfloor(E(y_1)/p\rfloor \tag{9}$$

$$qt_2 = \lfloor(E(y_2)/p\rfloor \tag{10}$$

$$\vdots$$

$$qt_k = \lfloor(E(y_k)/p\rfloor \tag{11}$$

B. Homomorphic Image Decryption Phase

Let $C_{p1} = E(y_1) \mod p$, $C_{p2} = E(y_2) \mod p$, $C_{p3} = E(y_3) \mod p$, and in general $C_{pk} = E(y_k) \mod p$, be the individual encrypted pixel intensity sub-values mapped to Z_p, and available at the receiver side, where E is an homomorphic encryption function. Also, let $qt_1 = \lfloor(E(y_1)/p\rfloor$, $qt_2 = \lfloor(E(y_2)/p\rfloor$, $qt_3 = \lfloor(E(y_3)/p\rfloor$, …, $qt_k = \lfloor(E(y_k)/p\rfloor$ be decryption parameters that are also available at the receiver side. To decrypt the encrypted pixel intensity value $E(y)$ and obtain the pixel intensity value y, one must first reconstruct or compute the individual encrypted pixel intensity sub-values $E(y_1)$, $E(y_2)$,$E(y_3)$, …, and $E(y_k)$ as follows:

$$E(y_1) = qt_1 \times p + C_{p1} \tag{12}$$

$$E(y_2) = qt_2 \times p + C_{p2} \tag{13}$$

$$\vdots$$

$$E(y_k) = qt_k \times p + C_{pk} \tag{14}$$

where $qt_k \times p + C_{pk}$ is a different constant integer for each k value. Once the above quantities from (12) through (14) are computed, one can apply the corresponding decryption function D to the following product in (15) and recover y. First, compute the product.

$$E(y) = E(y_1) \times E(y_2) \times E(y_3) \times \ldots \times E(y_k) \tag{15}$$

Applying the decryption function gives:

$$D[E(y)] = D\left[E(y_1) \times E(y_2) \times \ldots \times E(y_k)\right] \tag{16}$$

$$D[E(y)] = D\left[\Pi_{n=1}^{k} E(y_n)\right] \tag{17}$$

$$D[E(y)] = D[E(\sum_{n=1}^{k} y_n)] \tag{18}$$

$$D[E(y)] = D[E(y_1 + y_2 + y_3 + \ldots + y_k)] \tag{19}$$

$$D[E(y)] = y_1 + y_2 + y_3 + \ldots + y_k \tag{20}$$

$$D[E(y)] = y \tag{21}$$

Note that the transition from (17) to (18) is achieved using the homomorphic property of the encryption function E. In addition, if the encryption/decryption keys for each individual pixel intensity sub-values y_k are different, one can first decrypt each $E(y_k)$, then add the sum $y_1 + y_2 + y_3 + \ldots + y_k$ to obtain the pixel intensity value y. For implementation efficiency, the image's pixel intensity values can be processed together as a matrix instead of single pixels.

C. Special Case Implementation for $k = 2$ Component Images

In order to implement the proposed image encryption scheme and to verify that the proposed theoretical approach will provide expected results, we chose to implement the special case for $k = 2$, where k is the number of pixel intensity sub-values $y_1, y_2, \ldots y_k$, which we also named the number of pixel components.

1) Special Case Encryption Phase Implementation for $k = 2$ Component Images : Let $y = y_1 + y_2$ and $E(y) = E(y_1) \times E(y_2)$. We need to have an encryption function, E, with the above homomorphic property, where the encryption of a sum of two pixel intensity sub-values y_1 and y_2 equal to the product of the individual encrypted sub-values $E(y_1)$ and $E(y_2)$.
Consider Paillier's Cryptographic System with its encryption and decryption functions [3], [18], [24], where a value y can be encrypted as follows:

$$E(y) = g^y x^N \mod N^2 \tag{22}$$

where $N = s \times q$, and s, q are two prime numbers, while x is a random number such that

$x \in Z_N^* = \{1, 2, \ldots, (N-1)\}$, and g is an integer whose order l is a multiple of N, that is $g^l \equiv 1(\mod N)$, a value of $g = 1+N$ satisfies this condition when prime numbers s and q have the same length.
One can also note that the Paillier encryption scheme is a public-key cryptographic system that is also probabilistic. For the encryption scheme to be secure, the public key N must be a very large integer with for example more than 300 digits. One can show that Paillier encryption function is homomorphic and satisfies (2d) for $k = 2$. Using (22), one can encrypt each of the two pixel intensity sub-values y_1 and y_2 with Paillier encryption function and the same public key N as follows:

$$C_1 = E(y_1) = g^{y_1} x_1^N \mod N^2 \tag{23}$$

and

$$C_2 = E(y_2) = g^{y_2} x_2^N \mod N^2 \tag{24}$$

Applying $\mod p$ as shown in (6) and (7) to the above equations (23) and (24) for $k = 2$ gives

$$C_{p1} = E(y_1) \mod p = [g^{y_1} x_1^N \mod N^2] \mod p \tag{25}$$

and

$$C_{p2} = E(y_2) \mod p = [g^{y_2} x_2^N \mod N^2] \mod p \tag{26}$$

The quantities C_{p1} and C_{p2} represent the cipher values corresponding to each of the pixel intensity sub-values y_1 and y_2. These encrypted values C_{p1} and C_{p2} represent the secure image pixel intensity sub-values that will be transmitted and / or stored.

2) Special Case Decryption Phase Implementation for $k = 2$ Component Images: For the special case where the number of pixel intensity sub-values $k = 2$, assume that the same key is used to encrypt y_1 and y_2, and C_{pk} and qt_k from (8) and (11) are available at the front end of the receiver. Before applying the decryption function D, one must first calculate the encrypted pixel intensity sub-values $E(y_1)$ and $E(y_2)$ using the expression from (14) for $k = 2$ as shown:

$$E(y_1) = qt_1 \times p + C_{p1} \tag{27}$$

and

$$E(y_2) = qt_2 \times p + C_{p2} \tag{28}$$

Using Eqs. (16) through (21) for $k = 2$, one can write

$$D[E(y_1) \times E(y_2)] = D[E(y_1 + y_2)] = D[E(y)] = y \tag{29}$$

When applying the Paillier Decryption function in the context of (2) d we can write:

$$C = E(y) = E(y_1) \times E(y_2) \tag{30}$$

and

$$y = \frac{L(C^\lambda \mod N^2)}{L(g^\lambda \mod N^2)} \mod N \tag{31}$$

$$y = \left[L(C^\lambda \mod N^2) \times \left(\left(L(g^\lambda \mod N^2) \right)^{-1} \mod N \right) \right] \mod N \tag{32}$$

where $N = s \times q$, and s, q are prime numbers, g can be set to be $g = 1+N$ when s and q have the same length as previously stated. The parameter λ is given by the least common multiple of $s-1$ and $q-1$, while the function $L(U)$ is defined as

$$L(U) = \frac{(U-1)}{N}. \tag{33}$$

3) Proposed Image Cryptographic Scheme Block Diagram: Figure 1 shows the block diagram for the special case implementation where the number pixels intensity sub-values, $k = 2$. It is possible to expand the component images homomorphic encryption sub-block in Fig. 1 to 3, 4, 5, 6, or more, based on the value of k, to produce more encrypted component images and increase the security.

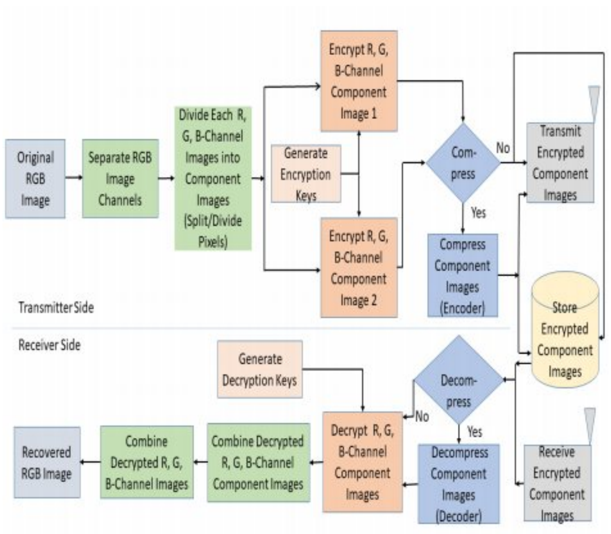

Original RGB Image
Separate RGB Image Channels
Divide Each R, G, B-Channel Images into Component Images (Split/Divide Pixels)
Encrypt R, G, B-Channel Component Image 1
Generate Encryption Keys
Encrypt R, G, B-Channel Component Image 2
Com-press
No
Yes
Transmit Encrypted Component Images
Compress Component Images (Encoder)
Transmitter Side
Receiver Side
Store Encrypted Component Images
Generate Decryption Keys
Decom-press
No
Yes
Recovered RGB Image
Combine Decrypted R, G, B-Channel Images
Combine Decrypted R, G, B-Channel Component Images
Decrypt R, G, B-Channel Component Images
Decompress Component Images (Decoder)
Receive Encrypted Component Images

Fig. 1: Proposed Special Case Image Cryptographic Scheme for $k = 2$ Component Images.

III. Performance and Security Analysis Results

The performance and security analyses are designed to verify that the proposed homomorphic image encryption scheme meets some required performance tests and can resist security attacks. This performance and security analyses include Correlation Analysis, Information Entropy, Cipher Cycle, Histogram Analysis, Chosen-Plaintext Attacks, and Brute force Attacks. Simulation results are obtained from a Mathematica software we have written and run using a Laptop computer with a processor having the following specifications: Intel (R) Core(TM) i3-4030U CPU @ 1.90 GHz 1.90 GHz.

A. Input Output and Encrypted Component Images

The test image (Baboon) used in this research is from the University of Southern California Signal and Image Processing Institute image set [22].
The recovered images and cipher-images are presented and discussed in this chapter. We randomly generate our private keys using our Mathematica code implementation, prime numbers s and q, with each 165 digits to be
$s = 135231806007162061442750686391904431697185 19$
756636700661898178469167610743760305379535 0082927
665613024925359860877093300237421910195680 1142939
4756777797433965568062 1
and
$q = 61599275563119689204225566492771724864296690$
58166213935794072925776888540609070902454787 2520
96403563328826147475033010782584067279640060 0750
3687722380089774424939227.
The number of bits for s and q are 546 bits and 548 bits, respectively. Therefore, the value of the public key $N = s \times q$ used for encryption has 329 digits and 1093 bits, and is given by
$N = 833018128313352036165812028938761904191512668$
904551362732364681607018389028803357833367866 51206
971819738814978313823253632444648110015513277 6532
099225547400460018876566048064044754229926275 0458
887672072976607578472070872148044189831835405 0969
886598589399599689009510092135223462538502161 0968
2134503606251866477057119002324661 9967.
The original RGB image shown in Fig. 2 (a) is processed channel-wise by separating each of its R, G, and B-channel images which are shown in Figs. 2 (d), 2 (g) and 2 (j), respectively. Each of these channel images is processed separately by first decomposing it into two component images before encrypting each of them. For instance, the R-Channel original image R in Fig. 2 (d) is decomposed into two component images R1 and R2 which are not shown here, then encrypted to produce Encrypted Component Images R1 and R2 shown in Figs. 2 (e) and 2 (f), respectively. Similarly, the G and B-Channel original images have each two component images that are encrypted to produce Encrypted Component Images G1 and G2 shown in Figs. 2 (h) and 2 (i) for the G-Channel, while

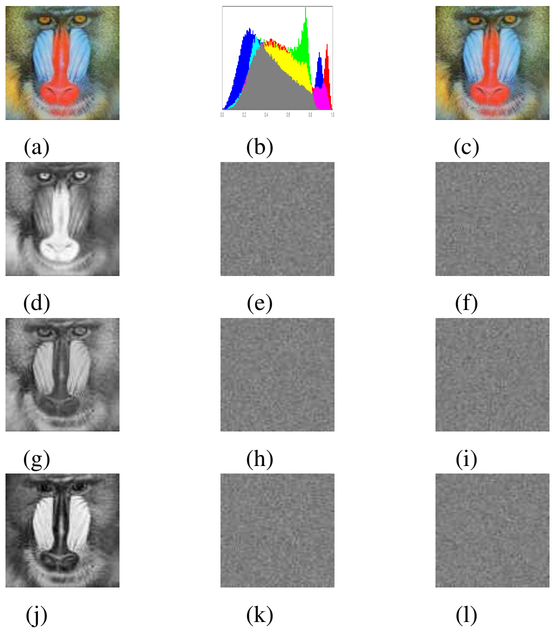


Fig. 2: (a) Original Baboon RGB Image;
(b) Histogram of Original Baboon RGB Image;
(c) Recovered Baboon RGB Image;
(d) Original R-Channel Baboon Image R;
(e) Encrypted Baboon Component Image R1;
(f) Encrypted Baboon Component Image R2;
(g) Original G-Channel Baboon Image G;
(h) Encrypted Baboon Component Image G1;
(i) Encrypted Baboon Component Image G2;
(j) Original B-Channel Baboon Image B;
(k) Encrypted Baboon Component Image B1;
(l) Encrypted Baboon Component Image B2

Figs. 2 (k) and 2 (l) correspond to the encrypted component images for the B-Channel original image. Fig. 2 (b) shows the histogram of the original Baboon image that is discussed next, while Fig. 2 (c) displays the recovered RGB image obtained by combining the recovered R, G, and B-Channel images.
The number of component images is not just limited to the special case of two component images for each channel. For instance, the R-Channel component images can also be extended to R3, R4, R5, . . . Rk as explained in section II.
On the receiver side, the encrypted component images for each channel are used to recover the corresponding original channel images. For instant, the encrypted component images R1 and R2 in Figs. 2 (e) and 2 (f) are used to recover the R-Channel image shown in Figure 3 (a). Similar approach is taken for the G and B-Channel images. Also, the recovered R, G, and B-Channel images in Fig. 3 are combined to form the recovered RGB image in Fig. 2 (c).

B. Histogram Analysis

The histogram of a digital image provides information about the distribution of its pixels intensity values. For an

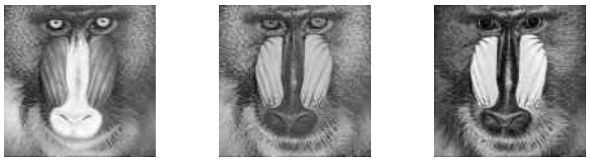


Fig. 3: (a) Recovered R-Channel Baboon Image
(b) Recovered G-Channel Baboon Image
(c) Recovered B-Channel Baboon Image

image with intensity levels in the discrete interval $[0, L-1]$, the histogram is given by the discrete function $h(l) = n_l$, where l is the l^{th} intensity value, and n_l represents the number of pixels in the image with intensity value l. [19]
A histogram analysis of the cipher-images produced by our proposed homomorphic image encryption scheme is performed for each channel image and its associated encrypted component images. So, the histogram analysis for the R-Channel original image R in Fig. 4 (a) is shown in Fig. 4 (d), and it is nonuniform, while the encrypted component images R1 and R2 in Figs. 4 (b) and 4 (c) have their histograms in Figs. 4 (e) and 4 (f), respectively; which also show that the encryption algorithm is able to produce a uniform-like distribution of the pixel intensity values for each of the encrypted component images; and therefore, can resist histogram analysis attacks.
Similar histograms for the G, and B-Channel original images, and associated encrypted component images G1, G2, B1, and B2 also provide similar results.

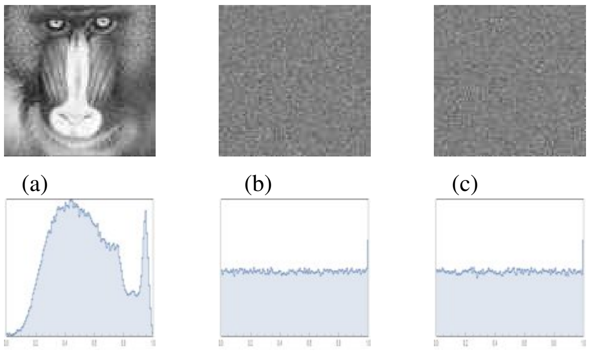


Fig. 4: (a) Original R-Channel Baboon Image R;
(b) Encrypted Baboon Component Image R1;
(c) Encrypted Baboon Component Image R2;
(d) Histogram of Original R-Channel Baboon Image R;
(e) Histogram of Encrypted Baboon Component Image R1;
(f) Histogram of Encrypted Baboon Component Image R2

C. Cipher Cycle

One of the requirements of an image encryption scheme is to produce an encrypted image that is very different from its original plain-image. To quantify this difference between the corresponding pairs of plain-image and cipher-image, one can use two criteria denoted by the Number of Pixel Change Rate (NPCR) and the Unified Average Change Intensity (UACI) [5], [9], [10], [20], [21]. The NPCR and UACI also may be used as a security test to protect against differential attacks, which consist of making a slight change on the cipher-image and observing the changes on the result. On one hand, the expression for the NPCR, which measures the average number of pixels in difference of a color component between two images C and C', is given by:

$$NPCR_{R,G,B} = \frac{\sum_{i,j} D_{R,G,B}(i,j)}{N} \times 100\% \tag{34}$$

where N is the image's total number of pixels, and the definition of $D_{R,G,B}$ is given by:

$$D_{R,G,B}(i,j) \triangleq \begin{cases} 0, & if \quad C_{R,G,B}(i,j) = C'_{R,G,B}(i,j) \\ 1, & if \quad C_{R,G,B}(i,j) \neq C'_{R,G,B}(i,j) \end{cases} \tag{35}$$

where $C_{R,G,B}(i,j)$ and $C'_{R,G,B}(i,j)$ represent the values of corresponding color component R, G, and B in images C and C', respectively.
Given two random images, an expression for the expected value of $NPCR$ can be found to be

$$\mathbb{E}(NPCR) = (1 - 2^{-L_{R,G,B}}) \times 100\% \tag{36}$$

where $L_{R,G,B}$ represents the number of bits used to encode each color components R, G, or B. For instance, given two random images each with size 512×512 and 24-bit true color with 8 bits for each R, G, and B channels $(L_R = L_G = L_B = 8)$, the expected value of the NPCR is given by:

$$NPCR_R = NPCR_G = NPCR_B = 99.609375\% \tag{37}$$

One the other hand, the expression for the $UACI$ is defined as

$$UACI_{R,G,B} = \frac{1}{N}\left[\sum_{i,j} \frac{\left|C_{R,G,B}(i,j) - C'_{R,G,B}(i,j)\right|}{2^{L_{R,G,B}}-1}\right] \times 100\% \tag{38}$$

where $L_{R,G,B}$ represents the number of bits used for each color component of Red (R), Green (G), or Blue (B), respectively.
Given two random images, the expected value of $UACI_{R,G,B}$ is given by

$$\mathbb{E}(UACI_{R,G,B}) = \frac{\frac{1}{2^{2L_{R,G,B}}-1}\left(\sum_{i=1}^{2^{L_{R,G,B}}-1} i(i+1)\right)}{2^{L_{R,G,B}}-1} \times 100\% \tag{39}$$

For the case of an RGB image where each channel is encoded using 8 bits, we have the following expected value:

$$\mathbb{E}(UACI_R) = \mathbb{E}(UACI_G) = \mathbb{E}(UACI_B) = 33.46354\% \tag{40}$$

The NPCR and UACI analyses comparing the plain-images and cipher-images produced by our image encryption scheme is performed and results for the B-Channel are shown in Table I. Similar results are also obtained for R and G-Channel images but not shown here. As can be seen in Table I, the NPCR between the original B-Channel image B and its associated encrypted component images B1 and B2 is 99.6136 and 99.6273, respectively. These values are very close to the expected value of 99.60937, and therefore, the encryption algorithm performs very well on changing the pixel intensity values in the B-Channel component images B1 and B2. The values for the UACI between the original B-Channel image B and its corresponding encrypted component images B1 and B2 shown in Table I, are very close to the expected value of 33.4635; therefore, the encryption algorithm perform also well for this case.

TABLE I: NPCR and UACI for B-Channel Original and Encrypted Images

B-Channel			
Test Type	B and B1	B and B2	Expected Value
NPCR (%)	99.6136	99.6273	99.60937
UACI (%)	31.3253	31.3231	33.4635

D. Correlation Analysis Results and Discussions

Many images in the visible electromagnetic spectrum range have redundancies that sometimes lead to a high correlation between neighboring pixels at the horizontal, vertical, and diagonal directions. A good image encryption scheme should break this correlation between neighboring pixels and reduce its value to almost zero; and therefore, producing an image more resistant to statistical attacks. The expression for a correlation coefficient between adjacent pixel pairs in an image is discussed by several authors that include A. Soleymani, A. Daneshgar, A. Kano [2], [4], [5], [7]–[10], [12], [20], [21].

$$r_{XY} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{41}$$

where $cov(X,Y)$ and $\sigma_X \sigma_Y$ are the covariance and product of the standard deviation of X and Y, respectively. The covariance and standard deviation for pairs of adjacent pixels have the following forms

$$cov(X,Y) = \frac{1}{N}\sum_{i=1}^{N}\left[(x_i - \mu_X)(y_i - \mu_Y)\right] \tag{42}$$

where x_i and y_i are values of adjacent pixel pairs selected at random, N the total number of adjacent pixel pairs (x_i, y_i) from the image, μ_X and μ_Y are the mean or expected values of X and Y, respectively, and are given by

$$\mu_X = \frac{1}{N}\sum_{i=1}^{N} x_i \quad \text{and} \quad \mu_Y = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{43}$$

Using the variance, the expression for the standard deviation σ_X and σ_Y are:

$$\sigma_X = \left(\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_X)^2\right)^{\frac{1}{2}} \quad \text{and} \quad \sigma_Y = \left(\frac{1}{N}\sum_{i=1}^{N}(y_i - \mu_Y)^2\right)^{\frac{1}{2}} \tag{44}$$

A correlation analysis comparing each channel original images and their associated encrypted component cipher-images is conducted using a set $N = 2500$ random pairs of adjacent pixels in the horizontal and vertical directions. Results for this correlation analysis for the G-Channel images is shown in Table II. The correlation analyses for the R and B-Channel images are also conducted, and they provide similar results.
Table II shows that adjacent pairs of pixel intensity values in the original G-Channel image G are highly correlated in the horizontal and vertical directions, with correlation coefficient values close to 1. However the encrypted component images G1 and G2 have very uncorrelated adjacent pairs of pixels' intensity values because the associated correlation coefficients are close to 0. These results for the G-Channel image correlation analyses are also displayed in Fig. 5 Based on these results, one can conclude that the proposed image encryption scheme performed very well breaking up the correlation of adjacent pixel pairs in the horizontal and vertical directions; and therefore increases the security of the encrypted images by making them more resistant to statistical correlation attacks.

TABLE II: Correlation Coefficients of 2500 Adjacent Pixel Pairs for Baboon G-Channel Images

G-Channel			
Direction	Original G	Encrypted G1	Encrypted G2
Horizontal	0.86158	−0.000659307	0.00697017
Veritical	0.760878	0.0401025	−0.00679211

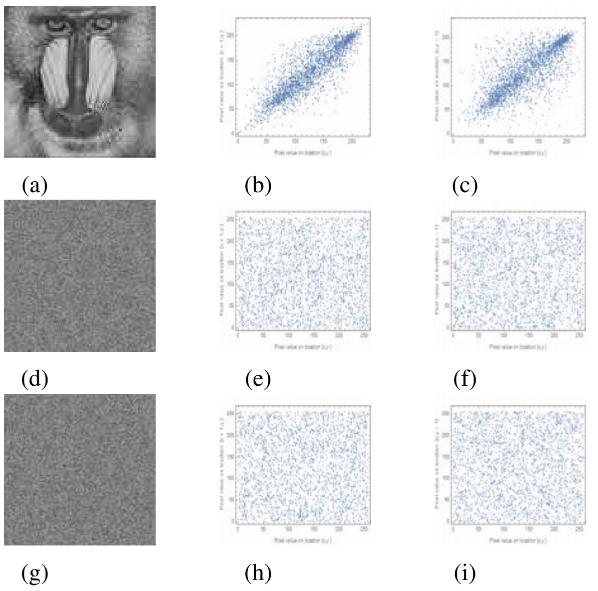


Fig. 5: (a) G-Channel Original Baboon Image G;
(b) Horizontal Correlation of Image G;
(c) Vertical Correlation of Image G;
(d) Encrypted Component Image G1;
(e) Horizontal Correlation of Image G1;
(f) Vertical Correlation of Image G1;
(g) Encrypted Component Image G2;
(h) Horizontal Correlation of Image G2;
(i) Vertical Correlation of Image G2
$N = 2500$ Adjacent Random Pixel Pairs.

E. Information Entropy

The degree of uncertainty in a random variable can be evaluated using its Information Entropy. Among all features of randomness, entropy is one of the most important. Given a source S with N symbols, where $N = 2^k$ and k the number of bits used to represent a symbol S_i, one can obtain the information entropy $h(S)$ as follows [4], [5], [10], [19], [20], [23]:

$$h(S) = -\sum_{i=0}^{N-1} p(S_i)\log_2[p(S_i)] \tag{45}$$

where $p(S_i)$ is the probability of occurrence for the symbol S_i, N the total number of symbols generated by the source,

TABLE III: Entropy Analysis for Baboon R, G, and B-Channel Images

R-Channel		
Encrypted R1	Encrypted R2	Expected Value
7.94141	7.94136	8
G-Channel		
Encrypted G1	Encrypted G2	Expected Value
7.94062	7.94512	8
B-Channel		
Encrypted B1	Encrypted B2	Expected Value
7.94333	7.94229	8

and the log base 2 is used in order to express the entropy in bits. When S is a truly random source, $p(S_i) = \frac{1}{2^k}$ for all i, the Entropy can be calculated to be

$$h(S) = k \tag{46}$$

Results of our entropy analysis are given in Table III.
Table III shows the entropy values for each of the encrypted channel component images R1, R2, G1, G2, B1, and B2, are very close to the expected value of 8. So, these encrypted component images are truly random; and therefore, our proposed homomorphic image encryption scheme is robust enough to protect against Entropy attacks.

F. Chosen-Plaintext Attacks

In the chosen plaintext attacks, the opponents have access to at least a pair of plaintext and ciphertext in addition to the encryption algorithm and try to find the structure of the encryption key. If the key is found, all past and future ciphers encrypted using this key can be decrypted using the found key [12], [13], [23]. Our proposed Homomorphic cryptographic approach can resist this type of attack because it always produces a difference cipher-image when the same original image is encrypted several times using the same encryption key as shown in Fig. 6 . Similar results are also obtained for the encrypted G, and B-Channel images but not shown here.

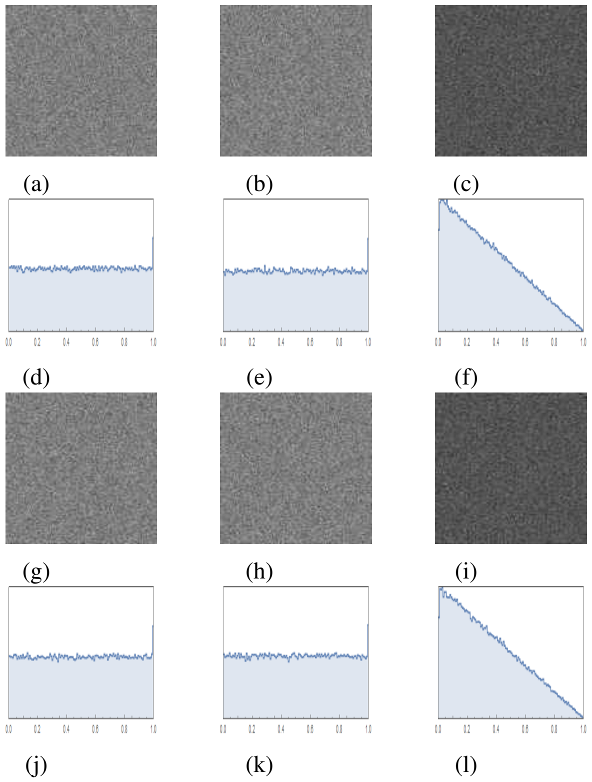


Fig. 6: (a) Encrypted Comp. Image R11 and its Hist. in (d); (b) Encrypted Comp. Image R12 and its Histogram in (e); (c) Pixel-wise difference $|R11-R12|$ and its Histogram in (f); (g) Encrypted Component Image R21 and its Histogram in (j); (h) Encrypted Component Image R22 and its Histogram in (k); (i) Pixel-wise difference $|R21-R22|$ and its Histogram in (l).

G. Timing and Recovered Image Quality Analyses Results and Discussions

This section presents and discusses the results for the timing and quality of the recovered images in Tables IV and V for the R-Channel. The timing simulation results are obtained using a Laptop computer with an Intel (R) processor with Core(TM) i3-4030U CPU @ 1.90 GHz 1.90 GHz, and they can change when using a processor with a different speed. Timing results can also change based on the efficiency of the algorithms used to implement the encryption and decryption functions. In Table IV, the columns denoted by component image R1 encryption time (R1ET) and component image R2 encryption time (R2ET) give the time in seconds (S) or minutes (min) it takes to encrypt component images R1 and R2, respectively, while the column decryption time (DT) provides the time needed to decrypt and recover the R-Channel original image. In Table V, The column NPCR (Number of Pixels Change Rate) gives the data that compares the R-Channel original and recovered images in order to show the percentage of corresponding pixel values that have changed as a result of the encryption and decryption processes, when the encrypted images are not compressed. It is desired to have the values on this NPCR colunm as small as possible, meaning that the original and recovered R-Channel images are almost the same and very little information is lost during the encryption and decryption operations. The column RGBNPCR gives the data that compares the original and recovered RGB images similar to the previous NPCR column. Finally, the last column named Quality includes terms such as VL = Very Low, H = High, VH = Very High are subjective qualitative description used to judge the quality of the recovered images based on the percentage results shown in the previous two columns, namely columns NPCR and RGBNPCR; which show that very high quality recovered images can be obtained when the number of bits of the public encryption key N is ≥ 20. Similar tables are obtained for the G and B-Channel.

TABLE IV: R-Channel Timing Analysis

R-Channel:

n (Digits)	n(Bits)	R1ET	R2ET	DT
4	11	31.42 S	30.89 S	16.27 S
8	27	55.59S	54.25 S	29.85 S
20	65	2.18 min	2.17 min	1.79 min
60	197	6.05 min	6.03 min	5.99 min
100	330	10.60 min	10.54 min	10.81 min
150	496	17.32 min	17.73 min	18.65 min
200	662	23.42 min	23.38 min	25.75 min
249	827	32.95 min	32.57 min	38.73 min
329	1093	53.18 min	53.69 min	64.43 min

TABLE V: R-Channel Recovered Image Quality Analysis

R-Channel:

n (Digits)	n(Bits)	NPCR	RGBNPCR	Quality
4	11	16.56%	41.89%	VL
8	27	0.068%	0.18%	H
20	65	0.025%	0.055%	VH
60	197	0.0256%	0.056%	VH
100	330	0.026%	0.056%	VH
150	496	0.0256%	0.056%	VH
200	662	0.0256%	0.056%	VH
249	827	0.0256%	0.056%	VH
329	1093	0.0256%	0.056%	VH

H. Image Compression Results and Discussions

This section provides the image compression results shown in Table VI, for the R-Channel. For each encrypted channel component image R1 and R2, the number of bytes before and after compression is found using our simulation program. The last column of this table shows a reduction in encrypted image data size by a factor approximately equal to 3.5 for the R, G, and B-Channel encrypted images. This means that less bandwidth and time are needed to transmit the encrypted data, and less memory is needed to store this encryption information for future decryption processing. Similar results are also obtained for the G and B-Channel encrypted images.

TABLE VI: Image Compression Results

R-Channel

Images	Before (Bytes)	After (Bytes)	Reduction
Encrypted R1	2097304	594648	1 : 3.53
Encrypted R2	2097304	594768	1 : 3.53

IV. Conclusion

In this research, a novel image encryption scheme that uses a homomorphic function property to encrypt an image and produce more than one cipher-image for each plain-image is proposed. During the encryption phase, an original RGB image is separated into its R, G, and B-channel images, then each pixel intensity value in each channel image is divided or decomposed into a sum of several pixel intensity sub-values to produce many component channel images that are separately encrypted using the same encryption key, compress if necessary, and then transmitted or stored. On the decryption side, the encrypted component channel images are decompressed if necessary, then decrypted using the same key and combined to produce each of the R, G, and B-channel recovered images that are also combined to obtain the recovered RGB image. Simulation results show that the associated component cipher-images can withstand a wide range of security and analysis attacks including Histogram Analysis, Entropy Analysis, Correlation Analysis, Chosen-Plaintext Attacks, Brute Force Attacks, and others. Also, high quality recovered channel images and recovered RGB image is obtained, meaning that very little information is lost as a results of applying our proposed encryption, decryption, and other image cryptographic and processing actions. Our proposed homomorphic image encryption scheme can be used in non real-time applications whereby highly secure encrypted images are needed, such as confidential satellite images, some confidential medical images, confidential fingerprint images, and any confidential images in the visible electromagnetic spectrum range. However real-time applications may be possible if faster encryption and decryption algorithms are implemented, as well as faster microprocessors or hardware are used instead.

Our main contribution is the formulation of a novel homomorphic image encryption scheme where each pixel intensity value in the original image is written as a sum of several sub-values, leading to producing many component images that are encrypted to produce many corresponding cipher-images; and therefore, increase the security of the associated images. The formulation includes encryption and decryption phases, as well as a block diagram for a special case implementation.

Future research could include the use of a homomorphic encryption and decryption functions that will require less computational time, for possible real-time applications. Our proposed encryption scheme could also potentially be extended to video encryption when using the appropriate homomorphic functions and special rapid processing approaches. Our proposed approach can be applied to any data that can be mapped to number greater than 1. For instance, alphabet letters from a to z mapped to numbers greater than 1 can be encrypted using our approach by writing each number as a sum of several numbers, then use a homomorphic encryption function to encrypt each and take the necessary steps as described in our approach to increase security.

References

[1] A. Daneshgar and B. Khadem, *"A self-synchronized chaotic image encryption scheme,"* Signal Processing: Image Communication 36 (2015) 106-114.
`www.elsevier.com/local/image`

[2] A. Soleymani, Md. J. Nordin, and Z. Md. Ali, *"A novel public key Image encryption based on elliptic curves over prime group field,"* Journal of Image and Graphics, Vol. 1, No. 1, March, 2013.

[3] A. K. A. Hassan, *"Reliable implementation of Paillier cryptosystem,"* Iraqi Journal of Applied Physics, IJAP, Vol. 10, No. 4, October-December 2014, pp. 27-29

[4] A. Daneshgar and B. Khadem, *"A self-synchronized chaotic image encryption scheme,"* Signal Processing: Image Communication 36 (2015) 106-114.
`www.elsevier.com/local/image`

[5] A. Kanso, M. Ghebleh, *" A novel image encryption algorithm based on a 3D chaotic map,"* Commun Nonlinear Sci Numer Simulat 17 (2012) 2943–2959,
`www.elsevier.com/locate/cnsns`

[6] G. Ye and X. Huang, *"An image encryption algorithm based on autoblocking and electrocardiography,"* Published by the IEEE Computer Society. April-June 2016.

[7] G. Zhang, and Q. Liu, *"A novel image encryption method based on total shuffling scheme,"* Optics Communications 284 (2011) 2775–2780,
`www.elsevier.com/locate/optcom`

[8] G. Chen, Y. Mao, and C. K. Chui, *"A symmetric image encryption scheme based on 3D chaotic cat maps,"* Chaos, Solitons and Fractals 21 (2004) 749–761,
`www.elsevier.com/locate/chaos`

[9] H.S. Kwok, Wallace K.S. Tang, *"A fast image encryption system based on chaotic maps with finite precision representation, "* Chaos, Solitons and Fractals 32 (2007) 1518–1529,
`www.elsevier.com/locate/chaos`

[10] H. Liu, X. Wang, and A. kadir, *"Image encryption using DNA complementary rule and chaotic maps , "* Applied Soft Computing 12 (2012) 1457–1466,
`www.elsevier.com`

[11] J. Zhou, X. Liu, O. C. Au, and Y. Yan Tang, *"Designing an efficient image encryption-then-compression system via prediction error clustering and random permutation,"* IEEE Transactions on Information Forensics and Security, VOL. 9, NO. 1, January 2014.

[12] L. D. Singh and K. M. Singh, *"Image Encryption using elliptic curve cryptography, "* Procedia Science 54 (2015) 475-481,
`www.sciencedirect.com`

[13] M. Kumar, D. C. Mishra, and R. K. Sharma, *"A first approach on an RGB image encryption, "*Optics And Lasers in Engineering 52 (2014) 27–34,
`www.elsevier.com/locate/optlaseng`

[14] Mamadou I. Wade, *" Distributed mage encryption based on a homomorphic cryptographic approach , "* Ph.D. Dissertation, Howard University, May 2017

[15] N. K. Pareek, V. Patidar, and K. K. Sud, *"Image encryption using chaotic logistic map, "* Image and Vision Computing 24 (2006) 926-934,
`www.elsevier.com/locate/optlaseng`

[16] P. P. Dang and P. M. Chau, *"Image encryption for secure internet multimedia applications, "* IEEE Trans. On Consumer Electronics, Vol. 46. No. 3, August 2000.

[17] R. Tao, X. Meng, and Y. Wang, *"Image encryption with multiorders of fractional fourier transforms, "* IEEE Trans. Inf. Forensics and Security, Vol. 5, No 4, Dec 2010.

[18] R. Rivest, Lecture Notes 15, Computer and Network Security: *"Voting, homomorphic encryption,"* October, 2002

[19] R. C. Gonzalez and R. E. Woods, *"Digital image processing.,"*3rd ed. Person Education Inc., 2008.

[20] R. Rhouma, S. Meherzi, and S. Belghith, *"OCML-based colour image encryption, "* Chaos, Solitons and Fractals 40 (2009) 309–318,
`www.elsevier.com/locate/chaos`

[21] S. Mazloom and A. M. E-Moghadam, *"Color image encryption based on coupled nonlinear chaotic map, "* Chaos, Solitons and Fractals 42 (2009) 1745–1754,
`www.elsevier.com/locate/chaos`

[22] *" University of Southern California, signal and image processing institute"*
`http://sipi.usc.edu/database/`

[23] Y. Zhou, L. Bao, C. L. P. Chen *"A new 1D chaotic system for image encryption,"* Signal Processing 97 (2014) 172–182,
`www.elsevier.com/locate/sigpro`

[24] Yi Xun, P. Russell, B. Elisa, *" Homomorphic encryption and applications "* 2014 XII, 126 p. 23 illus.,
`http://www.springer.com/978-3-319-12228-1`

[25] Z. H. Guan, f. Huang, and W. Guan , *" Chaos-based image encryption algorithm , "* Physics Letters A 346 (2005) 153-157 ,
`www.sciencedirect.com ; www.elsevier.com/locate/pla`

Secure Cloud Computing Algorithm Using Homomorphic Encryption And Multi-Party Computation

Debasis Das
Department of Computer Science and Information Systems
BITS Pilani, K.K. Birla Goa Campus,
Zuarinagar, Goa-403726, India.
Email: debasisd@goa.bits-pilani.ac.in

***Abstract*—Cloud computing is a developing technology that is yet unclear to many security issues. Data in the untrusted clouds can be encrypted using encryption algorithm. Randomizing this data provides more security which can be achieved by padding concept in the cloud. In this paper, the user's data is encrypted using padding scheme, called Optimal Asymmetric Encryption Padding (OAEP) together with Hybrid Encryption algorithm that is based on RSA (i.e., HE-RSA), in order to allow multiple parties to compute a function on their inputs while preserving Integrity and Confidentiality. The Homomorphic Encryption(HE) is performed on the encrypted data without decrypting it in computationally powerful clouds and the Secure Multi-Party Computation (SMPC) can be used in the cloud to ensure security and privacy of the users. In this paper, we have proposed a scheme that integrates the multi-party computation with homomorphic encryption to allow calculations of encrypted data without decryption. The cryptographic techniques used in our cloud model are described and the overheads are compared with Homomorphic Encryption and Multi-Party Computation.**

***Index Terms*—Cloud Computing; Optimal Asymmetric Encryption Padding; Homomorphic Encryption; Multiparty Computation.**

I. Introduction

There is a need for an appropriate or more suitable big data infrastructure [1] that supports the storage and processing on a high scale. Now a days the world is data centric, hence the big data processing and analysis have become the most important chore for any large establishment. The cloud computing is a model to provide convenient, on-demand access to share the computing resources. Organizations can simply connect to the cloud and use the available resources on the proper usage basis. The cloud computing has become a tool for analyzing big data using shared computing resources while easily handling changes in the volume and variety of the data[1]. The cloud provides many advantages such as more fault tolerance and multi-factor authentication to secure the information in the cloud. However, the cloud computing also comes with risks in maintaining the confidentiality and integrity of data due to these properties. In the last few years, there have been increasing the number of data breaches in the cloud as a result of malicious and intrusive actions. Encryption keeps the data at rest secure but data is lost if we lose the encryption key. Thus, to prevent the malicious attacks on the cloud, it is necessary to develop efficient cryptographic techniques [3], [4] which is resistant to active attacks as well as performing calculations of encrypted data without decryption. The cloud computing-based solutions have become increasingly popular in the past few years. The cloud computing platform analyzes and extracts useful information from the Big data cloud. One of the main concerns with cloud computing has been the privacy and confidentiality [5], [6], [7] of the data in the cloud. One solution is to send the data encrypted to the cloud. However, we still need to support useful computations on the encrypted data and Fully Homomorphic Encryption (FHE) [8], [9], [10], [11] is a way of supporting such computations on encrypted data. We note that while other mechanisms exist for secure computation, they generally require the different data providers to exchange information. Because FHE schemes are public key schemes, FHE is much better suited for the scenario where we have many sources of data.

The Secure Multi-Party Computation (MPC) [12], [13] guarantees that everyone learns the correct output of a joint computation but nothing else about anyone else's inputs, even when some of the user performing the computation might be actively or passively malicious. Secure MPC can be done for arbitrary computations and for any number of parties. Hence, we can view secure MPC protocols as compilers that take as input a specification of a function and output a protocol that computes the function securely. Hence, we can view secure MPC protocols as compilers that require as input a specification of a mapping and output is a protocol that computes the function securely. The Secure MPC [14], [15] offers both confidentialities as well as integrity which is much better than fully homomorphic encryption and verifiable computation. Dependable MPC can be performed for arbitrary computations and for any number of parties in Cloud Environment.
The benefits of our model:

- Integrates multi-party computation with homomorphic encryption

• Proposes a plan to use Optimal Asymmetric Encryption Padding(OAEP)-Homomorphic Encryption(HE)-RSA for encryption

In Section II, we describe related works. In Section III, we define the problem statement for secure cloud computing. In Section IV, we describe the Objective. In Section V, we describe the our contribution. Then, in Section V and VII, we describe the cryptographic techniques and give a detailed description of our scheme. In Section VIII, we describe the results. Finally, in Section IX, we conclude and describe goals achieved by our approach.

II. Related Work

M. Tebba et al. [2], proposed a technique to execute operations on encrypted data in the cloud which will provide us with the similar results after calculations as if we have worked directly on the raw data. Z. Wang et al. [3] gave a fresh definition of homomorphic signature for identity management in mobile cloud computing. S. Yakoubov et al. [4] conducted a survey of Cryptographic Approaches to Securing Big-Data Analytics in the Cloud. C. Rong et al. [5] conducted a survey on different security challenges in Cloud Computing. C. Gentry [6] computed arbitrary functions of encrypted data which describes a fully homomorphic encryption technique that keeps information private, but that leaves a worker that does not possess the private decryption key to compute any result of the data, even when the purpose of the data is really complex. C. Wang et al. [7] proposed an effective scheme with two salient features to ensure the correctness of the user's data in the cloud. Y. Yu et al. [8] investigated the active adversary attack in three auditing mechanisms for shared data in the cloud and also proposed a solution to remedy the weakness without sacrificing any desirable features of these mechanisms. L. Wei et al. [9] proposed a privacy cheating discouragement and secure computation auditing protocol, or SecCloud, which is a first protocol bridging secure storage and secure computation auditing in cloud. A. Lopez-Alt et al. [10] showed a new type of encryption scheme which they called multi key FHE. F. F. Moghaddam et al. [11] proposed a hybrid encryption algorithm based on RSA Small-e and Efficient-RSA for cloud computing environments.E. Shen et al. [13] proposed a scheme which is called Cryptographically Secure Computation [14] in the cloud using the concept of secure multi-party computation. This is a cryptographic approach that enables information sharing and analysis while keeping sensitive inputs secret faster and easier to use for application software developers. M. Bellare et al. [12] proposed Optimal Asymmetric Encryption with RSA. This work aimed to use the cryptography concepts in cloud computing communications and to increase the security.

D. Zissis et al. [18] addressed the details of cloud computing security issues and they proposed Public Key Infrastructure operating based on SSO and LDAP, to ensure the authentication, integrity and confidentiality of involved data and communications. The solution, presents a horizontal level of service, available to all implicated entities, that realizes a security mesh, within which essential trust is maintained. C. Hongbing et al. [20] presented an alternative approach called secure Big Data Storage and Sharing Scheme for Cloud Environment. Tenants which divides big data into sequenced parts and stores them among multiple Cloud storage service providers. Instead of protecting the big data itself, the proposed scheme protects the mapping of the various data elements to each provider using a trapdoor function. The cloud computing reviewed was presented in the paper [15]. In this paper [15] authors have discussed the relationship between big data and cloud computing, big data storage systems, and Hadoop technology [17], [19]. Furthermore, research challenges are investigated, with focus on scalability, availability, data integrity, data transformation, data quality, data heterogeneity, privacy, legal and regulatory issues, and governance. Lastly, open research issues that require substantial research efforts are summarized.

J. Zhou et al. [16] proposed a scheme called Secure and Privacy Preserving Protocol for Cloud-based Vehicular DTNs which solved the open problem of resisting layer-adding attack by outsourcing the privacy-preserving aggregated transmission evidence generation for multiple resource constrained vehicles to the cloud side from performing any one-way trapdoor function only once. The vehicle privacy is well protected from both the cloud and transportation manager.

III. Problem Formulation for Secure Cloud Computing

Consider three parties (shown in Fig. 1): a user Alice that stores her data in the cloud; a user Bob with whom Alice wants to share data; and a cloud service provider that stores Alice's data. To use the service, Alice and Bob begin by downloading a client application that consists of a data processor, a data verifier and a token generator. Upon its first execution, Alice's application generates a cryptographic key. We will refer to this key as a master key and assume it is stored locally on Alice's system and that it is kept secret from the cloud service provider. Cloud security is measured in terms of Availability, Integrity, and Confidentiality and encryption techniques are prone to a number of attacks like:

A. Availability

In this scenario cloud service providers have multiple servers. When one server fails, there is no security issue as another server is ready to provide services.

B. Integrity

The data integrity means the correctness and trustworthiness of the data. It ensures that the computation on sensitive data is correct. The data cannot be altered by the unauthorised user.

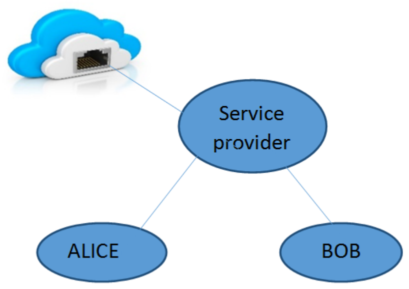


Fig. 1. *Cloud Architecture under various Cryptographic Techniques*

C. *Confidentiality*

Confidentiality is to prevent sensitive information from the reach of the attacker, while making sure that the authorised user have access to it. Services require user to trust the cloud with their data. But in the untrusted cloud Data owners do not trust the cloud. Thus user side protection is necessary. Users encrypt their data before storing into the cloud with the help of a public key .

D. *Cycle attack*

In this attack, the cipher text is encrypted repeatedly and the no of iterations are counted until the original text appears. It can decrypt any cipher text.

E. *Cipher text attack*

In this attack, both the plaintext and the cipher text is known to the attacker and he can use this to discover private exponent and once it discovered it is easy to find then. Multiple parties wish to perform operations on their inputs. This requires decryption of their data. This poses security problems in the case of untrusted Clouds.

Can multiple parties store their data with efficient cryptographic techniques that are resistant to attacks and perform the computations without decrypting their data?

IV. Objective

The Cloud environment requires protection and confidentiality of user data while leveraging computation ability of entities in the cloud network directly on encrypted data. This paper focuses on an issue that is attractive to many types of research, which is a data encryption for cloud computing. Cloud environment requires security and confidentiality of user data while leveraging the computational ability of entities in the cloud network directly on encrypted data. In this paper, we have proposed a scheme that integrates the multi-party computation with homomorphic encryption to allow calculations of encrypted data without decryption.

V. Our Contribution

In our paper, we have proposed an efficient cryptographic technique by padding the multiple party data before encrypting it. The user's data is encrypted using padding scheme Optimal Asymmetric Encryption Padding (OAEP) together with Hybrid Encryption algorithm that is based on RSA Small-e and Efficient RSA (HE-RSA). In order to allow multiple parties to compute a function on their inputs while preserving Integrity and Confidentiality. The homomorphic encryption is performed on the encrypted data without decrypting it in computationally powerful clouds. The proposed scheme integrates the multi-party computation with homomorphic encryption to allow calculations of encrypted data without decryption. The output of this nature allows maintaining confidentiality and integrity in the cloud environment.

VI. Cryptographic Technique for Secure Cloud Computing

A. *Preliminaries And Notations*

$$G : \{0,1\}^{K0} \longrightarrow \{0,1\}^{K0} \tag{1}$$

G is a mask generation function based off a hash function with SHA1 as defined by RFC 3447. G expands the K0 bits of r to K-K0 bits.

$$H : \{0,1\}^{K-K0} \longrightarrow \{0,1\}^{K-K0} \tag{2}$$

H is SHA-256 hash function. H reduces the K-K0 bit to K0 bits. r is a random seed of size K0.

B. *The OAEP Cryptosystem*

It is a padding scheme, proposed by Bellare and Rogaway ([12]), which prevents partial decryption of ciphertexts by adding an element of randomness.

Encode Operation

Select random integer r such that $1 < r < n$.

$$r \longleftarrow \{0,1\}^{K0} \tag{3}$$

$$S = (M \parallel 0^{K1}) \oplus G(r) \tag{4}$$

$$t = r \oplus H(s) \tag{5}$$

Return $s \parallel t$

Decode Operation

$$r = t \oplus H(s) \tag{6}$$

$$S = (M \parallel 0^{K1}) \oplus G(r), \tag{7}$$

r is a random seed of size K1.

C. HE-RSA

Hybrid Encryption algorithm that is based on RSA Small-e and Efficient RSA (HE-RSA) [11]. Efficient RSA was introduced as a scheme that employs the general linear group of order h with values that was intentionally selected randomly from the ring of integer mod n. n is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.

$$p, q \in prime, n = p.q \tag{8}$$

$$\emptyset(n) = (p-1)(q-1) \tag{9}$$

$$\Upsilon(n,h) = (p^h - p^0)(p^h - p^1)..(p^h - p^{h-1})(q^h - q^0)..(q^h - q^{h-1}) \tag{10}$$

Select random integer r such that $1 < r < n$ and $gcd(r, \Phi) = 1$ and $gcd(r, \Upsilon) = 1$

Compute e such that $r.e = 1 mod \Phi, 1 < e < \Phi(n)$

Compute d such that $d.e = 1 mod \Upsilon, 1 < d < \Upsilon(n)$

Public key: (e, n)

Secret key: (r, d, n)

D. Homomorphic Encryption

Operations can be performed on encrypted data using Homomorphic Encryption [2], [11], [20]. The result of these operations when decrypted using a secret key is same as if we had performed operations on the original data. Multiplicative Homomorphic encryption which allows only products on the original data.

$E(x.y) = E(x).E(y)$

VII. Proposed Algorithm for Secure Cloud Computing

The proposed secure cloud computing algorithm is ensuring the security and privacy [21], [22] of individual data in the cloud along with the enhancement of the security mechanism like Homomorphic Encryption and Multi Party Computation (MPC).

The Proposed Algorithm is based along the four phases: Key Generation, Encryption, Homomorphic Encryption (HE) and Multi Party Computation (MPC), and Deception. The main goal is to minimize the running time, cost, and the overhead during these four phases. In the proposed Algorithm, the number of exponents during key generation (in the step 1) has been enlarged in comparison to the some existing Algorithms (i.e., RSA). In addition to that, a dual encryption process (in the step 2) has been implemented in this Algorithm to prevent the general attacks against some existing techniques. In step 3, we have integrated the fully homomorphic encryption and multi-party computation to allow the calculations of encrypted data without decryption (in the step 4) in the cloud.

To interact with various services in the cloud and to store the data generated/processed by those services, several security capabilities are required. Suppose Alice and Bob send data to M1 and M2 respectively to the cloud and Alice's data (M1) is padded using Optimal Asymmetric Encryption Padding (OAEP) scheme before being encrypted using HE-RSA resulting in ciphertext C1 which is shown in Fig. 2.

Alices data (M1) is padded using Optimal Asymmetric

Algorithm 1 Algorithm for Secure Cloud Computing

```
**Step 1**: Key generation algorithm: keygen(p,q)
Randomly choose two large primes p ,q and compute n=p.q
.
$\emptyset(n) = (p-1)(q-1)$
$\Upsilon(n,h) = (p^h - p^0)(p^h - p^1) \ldots\ldots\ldots\ldots\ldots (p^h - p^{h-1})(q^h - q^0)(q^h - q^1) \ldots\ldots\ldots (q^h - q^{h-1})$
Select random integer r such that $1 < r < n$ and $gcd(R, \Phi) = 1$ and $gcd(R, \Upsilon) = 1$
Compute e such that $r.e = 1 \text{ mod } \Phi, 1 < e < \Phi(n)$
Compute d such that $d.e = 1 \text{ mod } \Upsilon, 1 < d < \Upsilon(n)$
Public key(pk): (e,n)
Secret key(sk): (r,d,n)
**Step 2**: : Encryption: Enc(M,pk)
Suppose Sender and Receiver send data to M1 and M2 respectively to the cloud
$G : \{0,1\}^{K0} \longrightarrow \{0,1\}^{K0}$
$H : \{0,1\}^{K-K0} \longrightarrow \{0,1\}^{K-K0}$
$r \longleftarrow \{0,1\}^{K0}$
$S = (M \parallel 0^{K1}) \oplus G(r)$
$t = r \oplus H(s)$
$C \longleftarrow \ll S \gg^e \text{ mod } n \gg^e \text{ mod } n$
Return C
**Step 3**: Homomorphism and Multi-party computation
Homomorphic computations are performed on Sender and Receiver encrypted data C1 and C2 respectively.
$C1 = ((M1)^e \text{ mod n})^e \text{ mod } n$
$C2 = ((M2)^e \text{mod } n)^e \text{ mod } n$
$C1.C2 = [((M1)^e \text{ mod } n)^e \text{ mod } n][((M2)^e mod \text{ n})^e \text{ mod } n]$
$= ((M1)^e \text{ mod } n)^e((M2)^e \text{ mod } n)^e \text{ mod } n$
$= ((M1M2)^e \text{ mod } n)^e \text{ mod } n$
$LetC = C1.C2, M = M1M2$
$C = (M^e \text{ mod } n)^e \text{ mod } n$
**Step 4**: Decryption: Dec(C,sk)
Sender and Receiver decrypt the computed data C using their respective private keys
$W \longleftarrow (C^r \text{ mod } n)^d \text{ mod } n$
Parse W as $s \| t$
$r \longleftarrow H(s) \oplus t$
$M^1 \longleftarrow s \oplus G(r), parseM^1asM\|Z$
```

Encryption Padding (OAEP) padding scheme before being encrypted using HE-RSA resulting in ciphertext C1 (shown

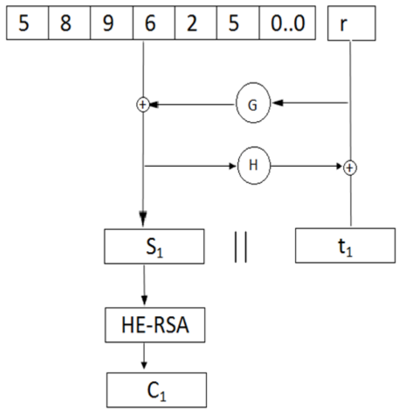


Fig. 2. *Sender's data M1(Encryption Process)*

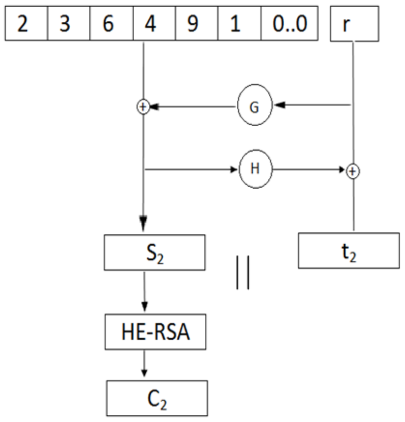


Fig. 3. *Receiver's data M2(encryption process)*

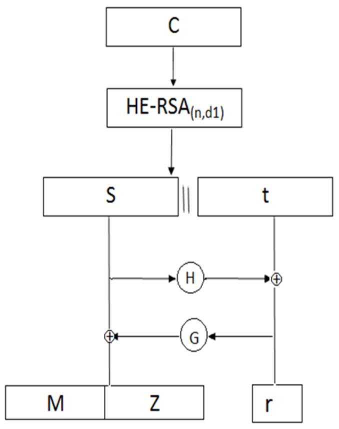


Fig. 4. *Sender's data(decryption process)*

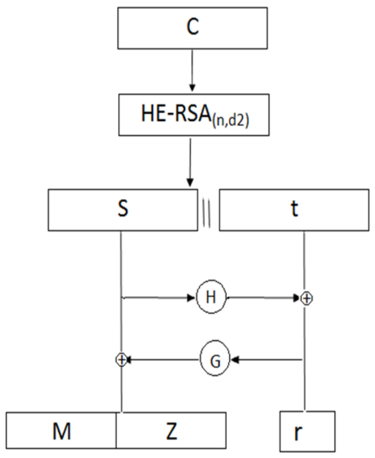


Fig. 5. *Receiver's data(decryption process)*

in Fig. 2). Receiver's data (M2) is padded using Optimal Asymmetric Encryption Padding(OAEP) scheme before being encrypted using HE-RSA resulting in ciphertext C2 (shown in Fig. 3). The data C resulting from homomorphic encryption is decrypted using Sender's private key and is then decoded using OAEP(shown in Fig. 4).

The data C resulting from homomorphic encryption is decrypted using Receiver's private key and is then decoded using OAEP(shown in Fig. 5). For instance, homomorphic encryption requires that all users and the eventual recipients of the results share a key to encrypt the inputs and decrypt the results, which may be difficult to arrange if they belong to different organizations. Also, homomorphic encryption does not allow for computation on data encrypted using different keys (without incurring additional significant overhead), thus making it impossible for users allow different access to data they contribute to the computation.

Secure multi-party computation (MPC) is suited to take advantage of the semi-trusted cloud setting. MPC leverage is the presence of honest parties, without necessarily knowing which parties are honest, to achieve confidentiality and integrity of the data and computation. Multi-party computation offers weaker security guarantees than FHE, but can be much more efficient. In MPC, no single party learns anything about the data, but if sufficiently many parties are corrupted by an adversary and pool their information, they can break confidentiality. The relative efficiency of MPC, as well as the applicability of the semi-trusted cloud model to the real world, make it a promising candidate for use in more practical secure cloud computation.

VIII. Results

After combining Homomorphic encryption and Multi Party Computation(HE +MPC), the confidentiality and integrity of the data is maintained and the overhead is less than Homomorphic Encryption but more than Multi Party Computation. So, we have received moderate overhead based on the Homomorphic Encryption and Multi Party Computation (shown in Table 1). Fig. 6 summarizes the approximate

TABLE I
COMPARISON OF CRYPTOGRAPHIC APPROACH

Cryptographic Technique	Confidentiality	Integrity	Interaction	Overheads
HE	YES	NO	NO	More Overheads
MPC	YES	YES	YES	Less Overheads
HE+MPC	YES	YES	YES	Moderate Overheads

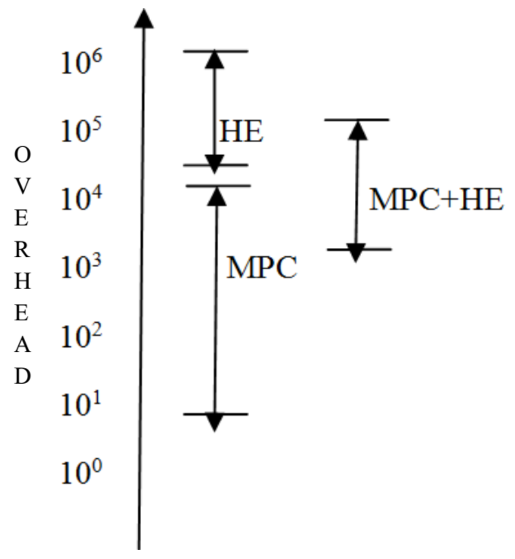


Fig. 6. *A graphical depiction of the multiplicative performance overheads over unsecured computation incurred by Homomorphic Encryption (HE), Multi Party Computation (MPC) and Homomorphic Encryption + Multi Party Computation (HE + MPC).*

efficiency, and cost of each of the techniques across a wide range of computations, depicting the multiplicative performance overhead incurred over unsecured computation. In addition to its inefficiency, homomorphic encryption has other limitations.

IX. Conclusion

In this paper, we proposed a secure cloud computing model in which efficient cryptographic technique Based on Homomorphic Encryption (HE) And Multi-party Computation(MPC) was used to encrypt user's data followed by operations on their data while maintaining integrity and confidentiality. The output is same as if the operations have been carried on raw data. A party is able to jointly perform computations without revealing their data to the other party. Here, we designed and developed secure homomorphic encryption and multi-party computation techniques tailored specifically for a private semi-trusted cloud setting. This setting allows developers to design the private cloud together with the cryptographic techniques (i.e., HE+MPC) necessary to protect it.

X. Acknowledgement

This work is partially supported by Early Career Research Award from Science and Engineering Research Board (SERB), Department of Science and Technology (DST), Govt. of India, New Delhi, India (Project Number : ECR/2015/000256) and BITS-Pilani K.K Birla Goa Campus Research Initiation Grant (RIG) Award 2017.

References

[1] Mell P, Grace T. *The NIST definition of cloud computing*, NIST Special Publication, 2009, pp. 800–145.
[2] Tebaa M, Hajji S.E, Ghazi A.E. *Homomorphic Encryption Applied to the Cloud Computing Security*, Proceedings of the World Congress on Engineering, London, U.K., Vol.1, No.1, 2014, pp. 4-6.
[3] Wang Z, Sun G, Chen D. *A new definition of homomorphic signature for identity management in mobile cloud computing*, Journal of Computer and System Sciences, Vol. 80, N0. 3, 2014, pp. 546-553.
[4] Yakoubov S, Gadepally V, Schear N, Shen E, Yerukhimovich A. *A Survey of Cryptographic Approaches to Securing Big-Data Analytics in the Cloud*, IEEE High Performance Extreme Computing Conference (HPEC), 2014, pp. 1–6.
[5] Rong C , Nguyen ST, Jaatun MG. *Beyond lightning: A survey on security challenges in cloud computing*, Computers and Electrical Engineering, Vol. 39, No. 1, 2013, pp. 47-54.
[6] Gentry C.*Computing Arbitrary Functions of Encrypted Data*, Communications of the ACM, Vol. 53, No. 3, 2010, pp. 97-105.
[7] Wang C, Wang Q, Ren K, Lou W. *Ensuring Data Storage Security in Cloud Computing*, Quality of Service, 2009, pp. 1–9.
[8] Yu Y, Niua L, Yang, G, Mu Y, Susilo W. *On the security of auditing mechanisms for secure cloud storage*, Future Generation Computer Systems, Vol. 30, 2014 pp. 127-132.
[9] Wei L, Zhu H, Cao Z, Dong X, Jia W, Chen Y, Vasilakos AV. *Security and privacy for storage and computation in cloud computing*, Information Sciences, Vol. 258, 2014, pp. 371-386.
[10] Lopez-Alt A, Tromer V, Vaikuntanathan E. *On-the-Fly Multiparty Computation on the Cloud via Multikey Fully Homomorphic Encryption*, Proceedings of the forty-fourth annual ACM symposium on Theory of computing, 2012, pp. 1219–1234.
[11] Brakerski Z,and Vaikuntanathan E. *Efficient fully homomorphic encryption from (standard) LWE*, SIAM Journal on Computing, Vol.43, No.2, 2011, pp. 831–871.
[12] Bellare M, and Rogawayy P. *Optimal Asymmetric Encryption How to Encrypt with RSA*, Advances in Cryptology Eurocrypt 94 Proceedings, Vol. 950, 1995, pp. 1–19.
[13] Shen E, Varia M, Cunningham RK, Vesey WK. *Cryptographically Secure Computation*, IEEE Computer Society, Vol. 48, No.4, 2015, pp. 78–81.
[14] Zissis D, Lekkas D. *Addressing cloud computing security issues*, Future Generation Computer Systems, Vol. 28, No. 3, 2012, pp. 583–592.
[15] Hashem IAT, Yaqoob I, Anuar NB, Mokhtar N, Gani A, Khan S.U. *The rise of 'big data' on cloud computing: Review and open research issues*, Information Systems, 2015, Vol. 47, pp. 98-115.
[16] Zhou J, Dong X, Cao Z, Vasilakos AV. *Secure and Privacy Preserving Protocol for Cloud-based Vehicular DTNs*, IEEE Transactions on Information Forensics and Security,Information Systems,Vo. 10, No. 6, 2015, pp. 1299 - 1314.
[17] Zhao J, Wang L, Tao J, Chen J, Sun W, Ranjan R, Kolodziej J, Streit A, Georgakopoulos D. *A security framework in G-Hadoop for big data computing across distributed Cloud data centres*, *Journal of Computer and System Sciences*, Vol. 80, No. 5, 2014, pp. 994-1007.
[18] Zuech R, Khoshgoftaar TM, Wald R. *Intrusion detection and Big Heterogeneous Data: a Survey*, Journal of Big Data, Springer, Vol. 2, No. 3, 2015, pp. 1-40.
[19] Hongbing C, Chunming R, Kai H, Weihong W, Yanyan L. *Secure Big Data Storage and Sharing Scheme for Cloud Tenants*, China Communications, 2015, pp. 106–115.
[20] Jajodia S, Kant K, Samarati P, Singhal A, Swarup V, Wang C. *Secure Cloud Computing*, Springer Science+Business Media, 2014, pp. 1-350.
[21] D. Das, R. Misra, A. Raj *Approximating Geographic Routing using Coverage Tree Heuristics for Wireless Network*, Wireless Networks (WINE), Springer US, Vol. 21. No. 4, 2015, pp. 1109-1118.
[22] T. Limbasiya, D. Das, *Secure Message Transmission Algorithm for Vehicle to Vehicle (V2V) Communication*, IEEE Region 10 Conference (TENCON)2016, Technologies for Smart Nation, Singapore, 22-25 Nov. 2016, Singapore, pp. 2507-2512.