May - June 2020
ISSN: 0193-4120 Page No. 3511-3520

Securing Homogeneous Big Data Using Augmented and Light-Weighted Homomorphic Encryption Scheme

[1]D.Anuradha, [2]S.Bhuvaneswari

[1]Part Time Research Scholar, Department of Computer Science
Pondicherry University-Karaikal Campus
[2]Registrar, Central University of Tamilnadu, Thiruvarur
[1]anuradha.d@vit.ac.in, [2]sbhuvaneswari@cutn.ac.in

Article Info
Volume 83
Page Number: 3511-3520
Publication Issue:
May - June 2020

Abstract

As the business and scientific data size grows faster in tremendous rate, big data would be the choice for the storage and management of the data. Big data analytics improves the corporate revenues and customer satisfaction. Also, big data analytics helps governments to plan people welfare schemes effectively. But sharing sensitive data among data analysts may lead to data privacy leakage in some cases. Homomorphic class of encryption schemes allow us to do mathematical calculations on encrypted data and the results can be used for analysis purposes meaningfully. But the existing homomorphic algorithms are relatively expensive in terms of complexity and runtime if used for big data. A new augmented homomorphic encryption algorithm, which is light-weighted when compared with other traditional algorithms is proposed in this work. Even though this design is light-weighted, it is robust against cryptanalysis, as this new algorithm produce different cipher for same message at different occurrences in a data block. A new method to transform different ciphers of the same message into a cipher, which is very useful for map-reduce operation.is also proposed here.Java programs are tested in HDFC-Hive environment and the results obtained shows that my algorithms are faster than other existing homomorphic algorithms.

Keywords: *Big data security, homomorphic encryption, Transform for Map-Reduce, HDFS, Hive.*

Article History
Article Received: *19 August 2019*
Revised: *27 November 2019*
Accepted: *29 January 2020*
Publication: *12 May 2020*

1. Introduction

Big data refers to very large data sets collected from wide range of data sources which is impossible to handle with traditional database software and frameworks [1]. In recent years business data analysis started playing a vital role in the growth of corporates around the globe [2]. Both the secure data management issues and value of big data become the new era of research. The increase of sensitive data leakage will help the hackers to infringe the data privacy of legitimate users. If he data privacy threshold decrease, the private data of users may be poisoned. Data may lose its credibility if it is tempered by illegal access, such as modifying review information in review web sites. Any strong authentication system may be breached by a new emerging illegal intrusion method, as the hacking technology is developing too fast. Since big data deals with unstructured heterogeneous types of data, implementing efficient access control policies may fail at some extent [3]. Almost all these flaws of big data can be fixed by encrypting the data before further handling. Since the rate of data growth is increasing rapidly for every second of time, the encryption of big data would consume considerable amount of time and resources. Also, the analysis of big data usually require sensitive data and their computation results. Not all the big data analyst are trustable. So, it would be very

May - June 2020
ISSN: 0193-4120 Page No. 3511-3520

convenient to employ an encryption scheme which allow users to do computations on encrypted data itself. The results thus obtained can be decrypted to get the original results. Homomorphic class of encryption algorithms provide us the ability to do the same.

Proposal of anew augmented homomorphic encryption technique, which is light-weighted and deterministic over the mathematical operations addition, subtraction, and multiplication of text data is done here.

This paper consists of the following sections. The second section describes the other related research works done in the field of big data encryption and security.The third section presents the discussion about the fundamentals of homomorphic encryption techniques and four popular homomorphic algorithms such Goldwasser-Micali encryption, ElGamal encryption, Paillier encryption and Boneh-Goh-NissimEncrytion algorithms are reviewed. The fourth section includes the proposal of new, augmented and light-weighted homomorphic encryption scheme with detailed description ofalgorithms for generating keys, encrypting data, decrypting cipher, and transform procedure (which is needed for map-reduce process of big data). The above sited three algorithms (Goldwasser-Micali, ElGamal, and Paillier encryption algorithms) are selected for performance comparison with new proposed algorithm.

The experimental setup used to test the algorithms is discussed in section five. Based on the results obtained analysis is done and reports are given in section six. The final conclusions are cited in section seven. The last section includes the references used for this work.

2. Related works

A. Hybrid Attribute Based Encryption

In 2005, A. Sahai, and B. Waters [4] introduced Attribute-Based-Encryption(ABE). In their work information security and data access control are implemented by providing various encryption methods for different user categories. Users and their roles are dynamically changing over time. Other revocation methods of ABE deals with decrypting the whole data and encrypting again with new ABE key. In hybrid ABE scheme, data is encrypted using Advanced Encryption Standard (AES) and the AES keys are encrypted using ABE. At the time of revocation of users, it's enough to change the ABE cipher text i.e AES key and re-encrypt the data with the new AES key without decryption. But this process needs the entire data on the client (Data owner) side. The process of download, re-encrypt, and uploading data back to the store incur a huge cost of communication depending upon the size of the data.

A proxy re-encryption method is proposed by Yoshiko Yasumura et.al [5], in which the data could be encrypted again in the cloud server at revocation of rights. This will reduce the overhead of communication between data owner and cloud server without downloading. The proxy encryption scheme given in figure 1 below,employs a hybrid of AES and ABE. The data owner may wish to change the userrights by revoking some of the users at any time.

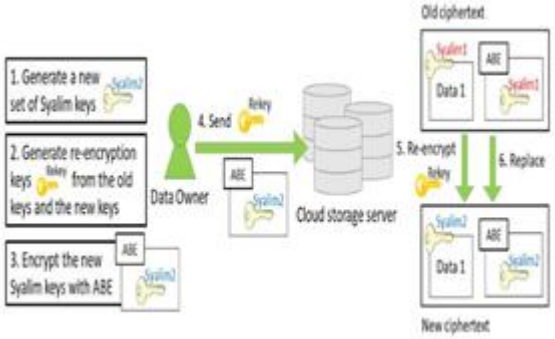

Figure 1: Proxy re-encryption protocol

At this point of time the data owner generates a new AES key and it was encrypted using ABE. Also a new re-encryption key (re-key) is generated using the old and new AES keys is generated by the data owner. Only the re-key and the cipher text of new AES key encrypted using ABE are sent to cloud server. At the cloud server, the encrypted data (using old AES key) is again encrypted with the re-key and the cipher text of old AES key encrypted using ABE is replaced with the cipher text of new AES key encrypted with ABE

So, the revoked users are not aware of the new AES key, that they cannot unpack the new AES key. Hence they cannot decrypt the data which was encrypted twice.This scheme greatly reduce the data traffic over network and also efficient in implementing access control. But the computation overhead of the proxy cloud server is increased as the re-encryption is to be done on entire data.

B. HDFS Data Encryption Scheme

Even though big data researchers believe that AES is the standard encryption algorithm for securing big data, Korean government have chosen ARIA algorithm for domestic data security. S. Parket. Al [6] developed a scheme which uses the AES using Split-able Compression Codec for encrypting HDFS data. The widely used AES algorithm [7,8] is known as symmetric key encryption algorithm and it was announced by US National Institute of Standards and Technology (NIST) – a non-regulatory US agency [9]. As the cryptographic algorithms involve highly complex mathematical computations, variouscorporate companies and government entities trust NIST recommendations on best cryptography. The ARIA algorithm [10,11]is commonly used for domestic purposes as Korean government found it as a best cryptographic algorithm. ARIA performs data encryption on data blocks of size 128-bit and uses key with 128/192/256 bit size. It supports 12/14/16 rounds of bit-wise operations such as XOR.

May - June 2020
ISSN: 0193-4120 Page No. 3511-3520

Table 1: Comparison between AES and ARIA algorithms

	AES	ARIA
Year	1999	2004
Authorizing entity	NIST	NSRI
Attribute	Domestic and international cryptography system	Alternative to AES in domestic applications
Service	National and private	Administrative

In their work, Y.Song et.al [12] both AES and ARIA algorithms are used in HDFS encryption. Figure 2 describes the architecture of the encryption process. The user choses the encryption algorithm either AES or ARIA. First the data is split into 64-

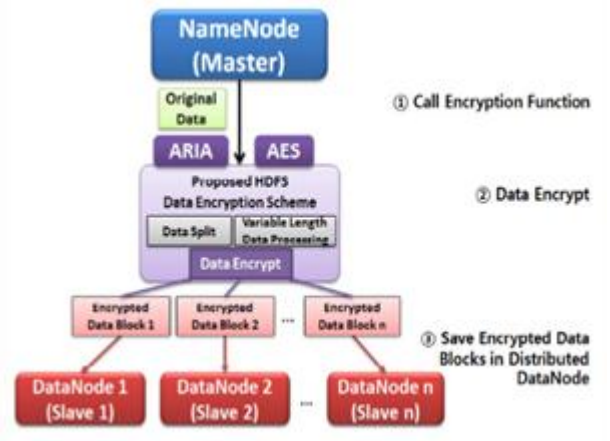

Figure 2: Architecture of HDFS encryption scheme

MB blocks. The module which is responsible for making all the data blocks are of same size, appends dummy data to the small data blocks. Each block is encrypted using the chosen encryption algorithm. The cipher blocks are mapped and merged together using Map-Reduce process to store in HDFS.

During the Map-Reduce process data need to be decrypted and encrypted again before storing in HDFS. If the data is very sensitive, decryption-encryption process is done for both Map and reduce operations separately. It has been proved that cost of encryption and decryption had a less dependency on data size in HDFS. ARIA based encryption can support various applications shows better performance when compared to AES based encryption in query processing.

C. BigCrypt Encryption Scheme for Big Data [13]

It is a probablistic encryption methodusing Preety Good Privacy (PGP) technique for encrypting big data, where both asymmetric and symmetric encryption algorithms are used. 80% of the typical victim of identity stealing are failed to detect the trap of stealing password and it was made easy to make unauthorized entry to databases or hijack the victim website[14]. A.A.Mamunet et. al.[13] presented a new encryption scheme using RSA (Rivest–Shamir–Adleman) [15] and Rijndael [16] for big data encryption.

NIST specifies three asymmetric ciphers RSA, DSA (Digital Signature Algorithm), and ECDSA (Elliptic Curve Digital Signature Algorithm)[9]. ECDSA is comparatively a new one and the other two algorithms are standard algorithms. RSA is preferred over DSA, since RSA allows us to use large keys. Rijndael encryption algorithm is preferred over the old DES by NIST in 2001 itself.

The BigCrypt scheme is shown below in figure 3. Receiver generates the required keys for RSA cipher and sends only the public key to the sender. The senderr can generate the Rijndael encryption key for data encryption. First the data is encrypted using Rijndael algorithm with a symmetric key. This cipher text is appended with its symmetric key, which is a cipher produced by RSA with its public key of the receiver.

The receiver seggregates the cipherr of the symmetric key and decrypts it using his/her private key of RSA. Using the decrypted symmetric key the cipher text is decrypted to get the original data back.This cipher scheme is tested in three different computing environments using a local Ubuntu server, an Apache web server and a MS Azure Linux SUSE cloud server. The test data set sizes varies from 100MB to 2GB. The tests are repeated using two different key lengths and the average values of each case are taken for conclusion. Local server has taken comparatively very less time for executing the algorithms in almost all the test cases. But when the data size increases the cloud server gave good performance when compared other two servers.

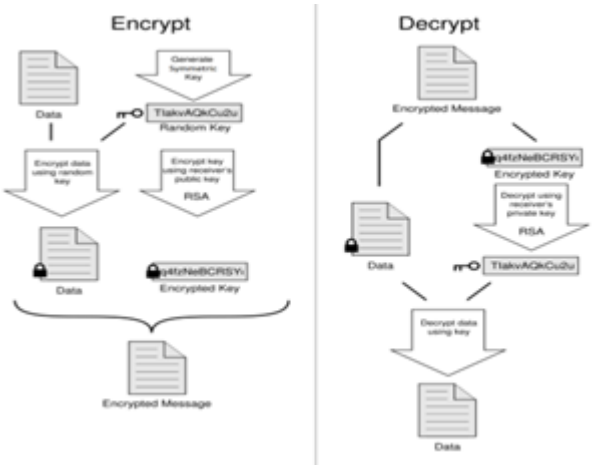

Figure 3: BigCrypt scheme

3. Homomorphic Encryption Schemes[17]

Encryption of data can be defined as the procedure of obscuring private data to make it indecipherable to people who are not intended users. Abstract algebra defines the

May - June 2020
ISSN: 0193-4120 Page No. 3511-3520

homomorphic property as the structure-preserving map between two algebraic structures. Homomorphic encryption has the property that, performing a computation on two values, (say 'a' and 'b') and encrypting the result is same as the result obtained by applying the same computation on the encrypted values of 'a' and 'b'.Hence, it decouples the ability to perform computations from the necessity to view the private data.

A. Goldwasser-Micali Encryption Scheme [18]

It is the first probabilistic homomorphic PKIestablished by ShafiGoldwasser and Silvio Micali in 1982. It produce cipher text which is few hundred times larger than the plain text. Hence this algorithm could not be used in applications. Goldwasser-Micali Encryption Scheme first decides a whether the chosenrandom value x is a square mod N, chosen to be the private key and N=p*q. The procedure for checking the square modulus is as follows:

- Calculate $x_p = x \pmod p$
- Calculate $x_q = x \pmod q$
- If $x_p^{(p-1)/2} = 1 \pmod p$ and $x_q^{(q-1)/2} = 1 \pmod q$ then, x is a quadratic residue of mod N.
- Otherwise, not

• Key generation

The sender selects two big prime numbers p, q randomly and independent to each otheri.e. p=q=3(mod4), and computes N as (p*q). The sender finds a non-residue 'a' as follows:

$$a_p^{(p-1)/2} = -1 \pmod p$$
$$a_q^{(q-1)/2} = -1 \pmod q$$

Then (a,N) is assumed to be the public key and (p,q) is private key for this cryptosystem.

- **Encryption**

For each and every bit 'm_i' in message 'm' a random value 'b_i' is generated i.e.GCD(b_i,N) =1. Using the public key the cipher is computed as $c_i = b_i^2 . a^{m_i} \pmod N$. The cipher text c= ($c_1$,$c_2$, … , c_n) is thus form for the message m = (m_1,m_2, ,,, ,m_n), where n is the bit count of message 'm'.

- **Decryption**

Upon receiving the cipher text 'c', the receiver determines whether each bit c_i in 'c'is a quadratic residue of mod N or not. If so then m_i is 0 otherwise m_i is 1. Thus the message is reconstructed in decryption as follows:

$$m_i = 0, \text{ if } c_i^{(p-1)/2} = 1 \pmod p$$
$$m_i = 1, \text{ if } c_i^{(p-1)/2} = -1 \pmod p$$

B. ElGamal Encryption Scheme [19]

It's anasymmetric key encryption designed by Taher ElGamal in 1985, which uses Diffee-Hellman key exchange scheme. It is used in GNU Privacy Guard software, new editions of PGP, and few other ciphers.

• Key generation

A cyclic group G is precisely chosen by the sender with an order q and generator g such that G_q: $g^q = I$, where I is an identity element of G with the defined operation. A random x belongs to the set {1,2,…,(q-1)} is selected and a value 'y' is computed as $y = g^x$. The set {(G,q,g), y} is made public and {x} is kept as the private key.

- **Encryption**

The sender choses a random r ∈ {1,2, … , q-1} to calculate the first half of the cipher C1 = g^r and a secret key S = y^r (using the public key). Now, the message m is converted into another element m' in the group G_q. the second half of the cipher is calculated as C2 = m' * S. The cipher text (C1,C2) = {g^r, m' * S} is sent to receiver.

- **Decryption**

The decryption the cipher text (C1,C2) with the secret key {x}, the receiver compute a value $t = C1^x$. Now, m` can be extracted from the encrypted dataas follows:

C2 = m` * S = m' * y^r [□ S = y^r]

C2 = m` * g^{x*r} [□ y = g^x] and

$C1^x = g^{x*r}$

To find m`,

m` = C2/C1 = m` * g^{x*r} / g^{x*r}

After m` is extracted from the cipher, the receiver converts m' in group G_q back to the original message m.

C. Paillier Encryption [20]

This is also a homomorphic public key cryptosystem, proposed by Pascal Paillier in 1999. The core logic of this encryption scheme is how to manage the residue in each calculation.

- **Key Generation**

Two big prime numbers p, q i.e.GCD(pq, (p-1)*(q-1)) = 1are chosen and the following computations are made:

- ☞ n = p x q
- ☞ λ = LCM((p-1), (q-1))
- ☞ Select a random integer 'g', where $g \in Z_{n^2}^*$
- ☞ $\mu = (L (g^{\lambda} \pmod{n^2}))^{-1} \pmod n$ where L = (u-1)/n

The sets {n, g} and {λ, μ}are assumed to be public and private keys, respectively.

- **Encryption**

Let m ∈ Z_n be the plain text and chose a random r∈ to compute the Z_n^* cipher c = g^m * $r^n \pmod{n^2}$. The cipher c is sent to receiver.

- **Decryption**

Upon receiving the cipher c, the receiver calculates the original message m as follows:

$$m = L(c^{\lambda} \pmod{n^2}) \times \mu \pmod n.$$

TEST Engineering & Management

May - June 2020
ISSN: 0193-4120 Page No. 3511-3520

D. Boneh-Goh-NissimEncrytion Scheme [21]

Boneh – Goh-Nissim cryptosystem resembles mainly Paillier encryption algorithm and it allows addition and multiplication with constant size cipher text. This algorithm takes the data domain as two additive groups G_1 and G_2 and a multiplicative group G_T with order 'p' for all the groups. Let $P \in G_1$be the generator of the groups $G_1$and $Q \in G_2$ be the generator of the group G_2. Device a pairing map e: G_1 X $G_2 \rightarrow G_T$, which hold good for the following:

- Bilinearity: $\forall a, b \in Z_p^*$ $e(P^a,Q^b) = a(P,Q)^{ab}$
- Non-degeneracy: $e(P,Q) \neq 1$
- The mapping e is computable in an efficient way.

- **Key generation**

For a random data security factor $\lambda \in Z^+$, a record is generated. It consists of the following:

- q_1 and q_2 are large prime numbers.
- G is a cyclic group with order N, where $N=q_1*q_2$
- e is a pairing map i.e. e: G X G $\rightarrow$ G_1

Choose two generators g, u, randomly from G to compute $h = u^{q}{}_2$. The public key is formed as the set {N, G, G_1, e, g, h} and the set {q_1} is assumed as the private key.

- **Encryption**

Let $m \in \{0, 1, 2, \ldots, t\}$, $t<q_2$and choose an integer 'r' from {1,2, … ,N} to compute the cipher text $c = g^m * h^r \in G$.

- **Decryption**

Using the key q_1 the plain text m can be computed from cipher as follows:

$$c^{q}{}_1 = (g^m * h^r)^{q}{}_1$$
$$= (g^m * u^{rq}{}_2)^{q}{}_1$$
$$= (g^{q}{}_1{}^m * u^{rq}{}_1{}^{q}{}_2)$$
$$= g^{q}{}_1{}^m \quad [\text{ u - generator of G}]$$

Since, $c^{q}{}_1 = g^{q}{}_1{}^m$, by finding the discrete logarithm of $c^{q}{}_1$ to the base of $g^{q}{}_1$, message m can be extracted.

4. Augmented Homomorphic Encryption

There are two data groups ($\hat{G}$, ɵ) and ($\hat{H}$, ϕ)[17]. The homomorphic property for groups from ($\hat{G}$, ɵ) to ($\hat{H}$, ϕ) is defined as a function f: $\hat{G}$ ➔ $\hat{H}$, i.e. for all 'g' and 'h' in $\hat{G}$ it is true that f(g ɵ h) = f(g) ϕ f(h). Let (Pl, Ci, K, En, De) be the cryptography system, where Pl and Ci are the plain test space and cipher text space respectively, K is the symmetric or asymmetric key space, "En" is the encryption algorithm to be used, and "De" is the decryption algorithm. In this work it is assumed that the plain text form the group (Pl, ɵ) space and cipher text forms the group (Ci, ϕ). The algorithm En is a mapping from Pl to Ci, i.e, En(k) : Pl ➔ Ci, where $k \in K$ and k can be either secret key or public key. The homomorphic property of my proposed algorithm is stated as follows:

$$\forall a, b \in Pl \text{ and } k \in K,$$
$$En(k, a \text{ ɵ } b) = En(k, a) \text{ ϕ } En(k, b)$$

In this design it is assumed that ɵ = ϕ. The operations that satisfied the above relations are addition, subtraction, and multiplication.

A. Key generation

The key used in this design is prepared using the first algorithm given below. The algorithm Key-Generator selects two large and random prime numbers $A \in Z^*$ and $B \in Z^*$ upon four criteria. The product of A and B is calculated. A homomorphic factor λ is prepared by finding out the LCM ((A-1),(B-1))

Algorithm 1 Key Generation

1: Key-Generation (Length l)
2: Generate A and B of length l for the following:
  a) A and B are large and random prime numbers
  b) $A \in Z^*$ and $B \in Z^*$
  c) A >> B
  d) GCD(A.B,(A-1).(B-1)) = 1
3: Calculate P ← A * B
4: Calculate a homomorphic factor λ by LCM((A-1),(B-1))
5: Return the key k ← {A, λ, P}

The set k = {A, λ, P} forms the key for the new proposed cryptosystem.

B. Encryption

The data to be encrypted in big data could be ofany format, i.e. structured, unstructured, or semi-structured data. Each data element from the big data file is first converted into numerical equivalent form. Let the data is 'M'. The encryption of 'M' (denoted as 'C') is given in the second algorithm.

The encrypted data 'C' is stored in big data centres and made available for trusted and authenticated using multi-factor authentication users. Users may save the result of their computation on 'C' back into the storage.

Algorithm 2 Encryption

1: En (k, M)
2: k ← {A, λ, P}
3: M ← Plain text message
4: Chose $\gamma \in Z_P$ is a random prime number
5: Calculate γ^{λ}
6: Calculate C ← [M + A * γ^{λ}] mod P
7: Return the cipher text C.

The data owners and/or the trusted users can decrypt the data if required. The encryption procedure is described pictorially in figure 4.

May - June 2020
ISSN: 0193-4120 Page No. 3511-3520

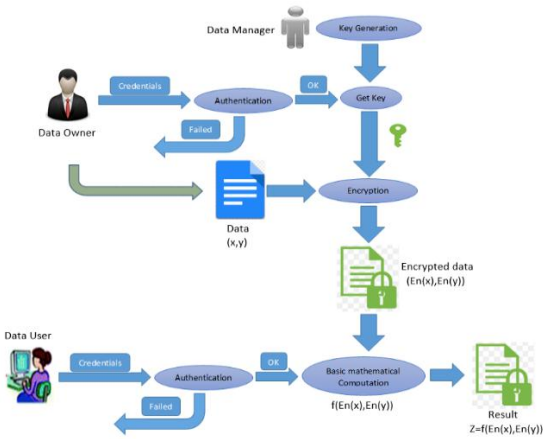

Figure 4: Proposed Encryption Scheme

C. Decryption

The data stored by the data owner can be retrieved back by decrypting 'C' to get the numerical form of the original data

Algorithm 3 Decryption

1: De(k, C)
2: k ← {A, λ, P}
3: C ← Cipher text
4: Calculate M ← [C] mod A
5: Return the plain text M.

Design of the decryptionprocess is given in Algorithm3. The decryption process can be described in figure 5.

Figure 5: Proposed Decryption Scheme

D. Proof of Correctness

The homomorphism of my algorithm can be proved as follows:Let m_1 and m_2 be the two plain text messages to be encrypted. When encryption algorithm is applied on messages m_1 and m_2, two ciphers c_1 and c_2 will be obtained.

$$c_1 \rightarrow (m_1 + A * (\gamma_1)^{\lambda}) \bmod P$$
$$c_2 \rightarrow (m_2 + A * (\gamma_2)^{\lambda}) \bmod P$$

Proof for addition operation

$$\begin{aligned} c_1 + c_2 &= [(m_1 + A * (\gamma_1)^{\lambda}) \bmod P] + [(m_2 + A * (\gamma_2)^{\lambda}) \bmod P] \\ &= [(m_1 + A * (\gamma_1)^{\lambda})] + [(m_2 + A * (\gamma_2)^{\lambda})] \bmod P \\ &= [(m_1 + m_2) + A*((\gamma_1)^{\lambda} + (\gamma_2)^{\lambda})] \bmod P \end{aligned}$$
$$c_1 + c_2 = (m_1 + m_2) + A*((\gamma_1)^{\lambda} + (\gamma_2)^{\lambda})$$

Decrypting the ciphers,

$$\begin{aligned} De(c_1 + c_2) &= [(m_1 + m_2) + A*((\gamma_1)^{\lambda} + (\gamma_2)^{\lambda})] \bmod A \\ &= m_1 + m_2 \end{aligned}$$

Proof for subtraction operation

Algorithm 4 Transform

1: Transform (k, C)
2: k ← {A, λ, P}
3: C ← Cipher text
4: Chose a R ∈ Z^+ is a random positive integer
5: Calculate $C^* \leftarrow (C * B * R) \bmod P$
6: Return C^*

$$\begin{aligned} c_1 - c_2 &= [(m_1 + A * (\gamma 1)^{\lambda}) \bmod P] - [(m_2 + A * (\gamma 2)^{\lambda}) \bmod P] \\ &= [(m_1 + A * (\gamma 1)^{\lambda})] - [(m_2 + A * (\gamma 2)^{\lambda})] \bmod P \\ &= [(m_1 - m_2) + A * ((\gamma 1)^{\lambda}) - ((\gamma 2)^{\lambda})] \bmod P \end{aligned}$$
$$c_1 - c_2 = (m_1 - m_2) + A * ((\gamma 1)^{\lambda}) - ((\gamma 2)^{\lambda})$$

Decrypting the ciphers,

$$\begin{aligned} De\,(c_1 - c_2) &= [(m_1 - m_2) + A * ((\gamma 1)^{\lambda}) - ((\gamma 2)^{\lambda})] \bmod A \\ &= (m_1 - m_2) \end{aligned}$$

Proof for multiplication operation

$$\begin{aligned} c_1 * c_2 &= [(m_1 + A * (\gamma 1)^{\lambda}) \bmod P] * [(m_2 + A * (\gamma 2)^{\lambda}) \bmod P] \\ &= [(m_1 + A * (\gamma 1)^{\lambda})] * [(m_2 + A * (\gamma 2)^{\lambda})] \bmod P \\ &= [(m_1 * m_2) + A * ((\gamma 1)^{\lambda}) * ((\gamma 2)^{\lambda})] \bmod P \end{aligned}$$
$$c_1 * c_2 = (m_1 * m_2) + A * ((\gamma 1)^{\lambda}) * ((\gamma 2)^{\lambda})$$

Decrypting the ciphers,

$$\begin{aligned} De\,(c_1 * c_2) &= [(m_1 * m_2) + A * ((\gamma 1)^{\lambda}) * ((\gamma 2)^{\lambda})] \bmod A \\ &= (m_1 * m_2) \end{aligned}$$

E. Transform (for Map-Reduce)

Same message can appear at different locations in the input plain text. In this case the encryption algorithm will produce different ciphers for the same message as follows:

May - June 2020
ISSN: 0193-4120 Page No. 3511-3520

Let's assume a message m is repeated at two locations in the plain text. The first encryption of m yields $c_1 \rightarrow (m + A * (\gamma_1)^{\lambda}) \bmod P$. The next encryption produces $c_2 \rightarrow (m + A * (\gamma_2)^{\lambda}) \bmod P$. this is the one of the advantages of my cryptosystem, as cryptanalysis will be very hard and in some cases it's impossible.

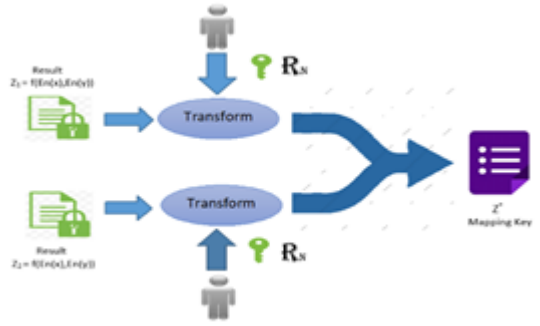

Figure 6: Transform operation

But when this new proposed cryptosystem to be used with Map-Reduce technique to handle big data, we need same cipher for all same valued plain text. To address this problem, a new Transform algorithm is formed, which will transform the different ciphers c_1 and c_2 of same message m into a new value c*. The algorithm works as follows:

Figure 6 describes the functionality of the transform algorithm. Now, c* can be used to map the same messages into the same cluster of any big data storage such as HDFS.

Proof of Transform operation

For a message m, assume two ciphers are computed as,

$$c_1 \rightarrow (m + A * (\gamma_1)^{\lambda}) \bmod P$$
$$c_2 \rightarrow (m + A * (\gamma_2)^{\lambda}) \bmod P$$

Now, applying the transform algorithm will yield a new value c* in both the cases as follows:

$$c_1^* = (c_1 * B * R) \bmod P$$
$$c_2^* = (c_2 * B * R) \bmod P$$

Proof is needed that if c_1 and c_2 were computed from same message m, then $c_1^* = c_2^*$. Since there is only one message m,

$$(c_1 * B * R) \bmod P = (c_2 * B * R) \bmod P$$
$$(c_1 * B * R) \bmod P - (c_2 * B * R) \bmod P = 0$$
$$[(c_1 * B * R) - (c_2 * B * R)] \bmod P = 0$$
$$[(c_1 - c_2) * B * R)] \bmod P = 0$$
$$[((m + A * (\gamma_1)^{\lambda}) \bmod P - (m + A * (\gamma_2)^{\lambda}) \bmod P) * B * R)] \bmod P = 0$$
$$[[(m + A * (\gamma_1)^{\lambda}) - (m + A * (\gamma_2)^{\lambda})] \bmod P) * B * R)] \bmod P = 0$$
$$[[(m - m) + A * ((\gamma_1)^{\lambda} - (\gamma_2)^{\lambda})] \bmod P) * B * R)] \bmod P = 0$$

Since P=A*B and P is very large,

m-m=0

So, if two messages $m_1=m_2$, then $c_1^* = c_2^*$

5. Experimental Setup

The experiments were carried out in Hadoop-HDFS system, version 2.7.2. The Apache Hadoop [22] is an open-source software framework for reliable, scalable, and dispersed computing for big data stored in numerous data cluster at different physical locations. Hadoop Distributed File System (HDFS) is anextremelyfault tolerant system, which can be deployed on low-cost hardware. It has master-slave architecture, in which a single 'Name node' and many 'Data nodes' are existing. The Name node is the master server and the data nodes manage the data storage in the clusters. The architecture of the HDFS is given below.

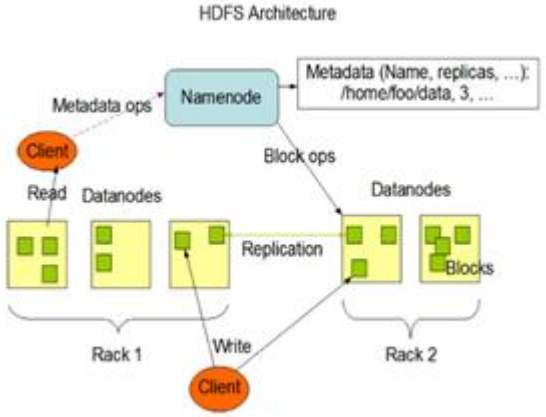

Figure 7: HDFS Architecture

In this work, a HDFS cluster setup is formed, where a name node and three data nodes are createdwith their IP addresses.The block size of the data nodes are set to 512MB. All the five systems are interconnected in LAN. The specifications of the four systems are given below.

Processor	Intel Xeon – 4 core
Speed	3.3 GHz
OS	Ubuntu14 – 64bits
Memory	4GB
Storage	500GB

Hive2.3.4is used to manage the data stored in HDFS clusters. Hive [23] is a data ware house infrastructure that provides facilities to read data, write data and other data manipulations using SQL queries. It's a distributed computing system, where the algorithms are executed in parallel in all the three data nodes.

Four different data sets are taken for testing purpose, whose sizes are 125M, 512M, 1GB, and 2GB. Experiments are repeated by changing the key (A and B) sizes as 256bit and 512bit.We have implemented the algorithms in Java, and used jdk1.8 and Eclipse 3.8.1 for executing the algorithms. Each test is done for six times and the average values are listed here.The time taken for

May - June 2020
ISSN: 0193-4120 Page No. 3511-3520

encryption for all combination of key sizes and data sets in four data nodes are given in tables 1.Similarly, the time taken for decryption operation is also recorded in table2.

Table 1: Execution time for encryption

		125M	512M	1GB	2GB
256 bit Key	Gold-Wasser	82.5	320.5	325.5	580.5
	ElGamal	70	280.5	281.1667	557.1667
	Paillier	75	290	287	562.6667
	Proposed algorithm	68	270.3333	272	545.3333
512 bit Key	Gold-Wasser	83.33333	323.1667	327	585.3333
	ElGamal	71.66667	283.1667	285.3333	560.5
	Paillier	78	295	289	565.1667
	Proposed algorithm	69	272	275	547.5

Table 2: Execution time for decryption

		125M	512M	1GB	2GB
256 bit Key	Gold-Wasser	72	268	271	499
	ElGamal	63	245.3333	244	486
	Paillier	67.5	252.1667	257	491.6667
	Proposed algorithm	60.66667	240.6667	241.1667	481
512 bit Key	Gold-Wasser	75	271.8333	275	502.1667
	ElGamal	65.16667	246.3333	247	489.1667
	Paillier	68	253.1667	258.5	495.1667
	Proposed algorithm	62.16667	241.3333	242.6667	485.5

6. System Analysis

We have chosen Goldwasser-Micali encryption system, ElGamal encryption system and Paillier encryption system for comparing with the proposed algorithm in performance.We have used jdk1.8 and Python3.2.7 in Eclipse3.8.1 IDE for executing the algorithms. To execute python implementation of Goldwasser-Micali encryption algorithm pydev plugin tool is installed in Eclipse3.8.1 IDE. All other three algorithms are in Java.The time taken for generating key is ignored as it is very negligible. The execution time taken for encryption and decryption algorithms are measured as performance in the evaluation.

All the four algorithms are executed by varying the input data se sizes and key sizes and time taken for encryption and decryption methods are listed in table 1 and 2. Each test is repeated for six times and the average value is taken. The graphs drawn for comparison of four algorithms shows the performance differences between proposed algorithm and all the other three algorithms.

The comparison of encryption runtimes of the four algorithms for the key size 256bit is shown as graph in figure 8. In all the graphs given here, the X axis is represents the different test data block sizes and the Y axis represents the run time in seconds. The encryption runtimes using 512bit key size is displayed in figure 9. The figures 10 and 11 describe the comparisons for decryption runtimes using key sizes 256bit and 512bit respectively.

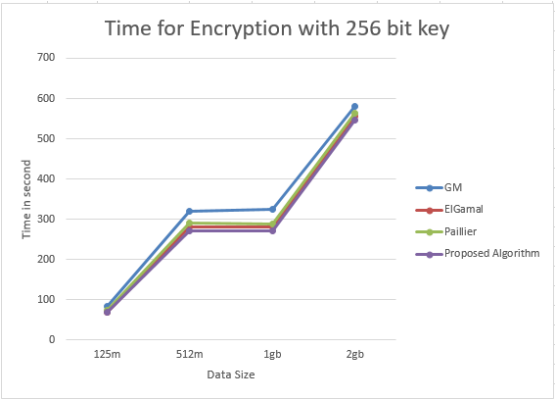

Figure 8: Comparison of Encryption runtimes with 256bit key

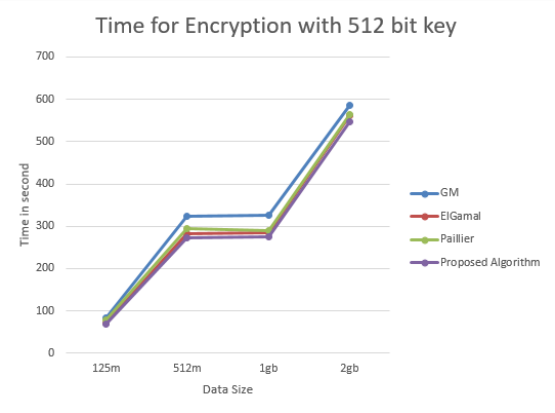

Figure 9: Comparison of Encryption runtimes with 512bit key

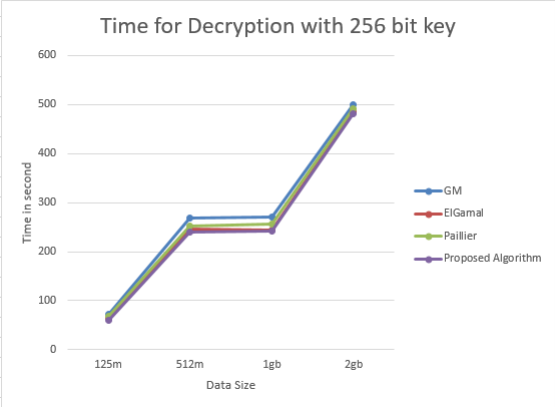

Figure 10: Comparison of Decryption runtimes with 256bit key

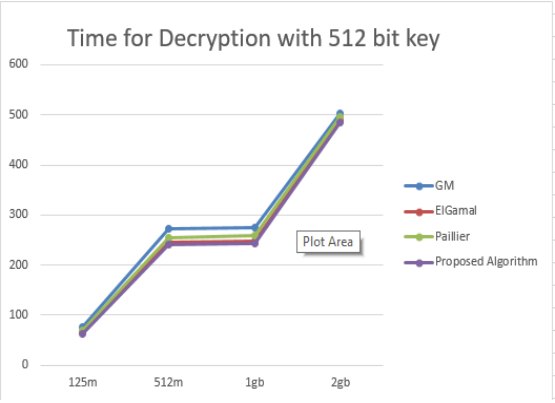

Figure 11: Comparison of Decryption runtimes with 512bit key

While the performance of all the four algorithms are examined, it was inferred that ouralgorithm shown best performance by consuming less time for encryption and decryption when compared to other three algorithms. This is because of the reason that the new proposed algorithm is relatively light-weighted when considering other three algorithms. At the same time ouralgorithms show their robustness towards cryptanalysis for diagnosing the plain text by analysing cipher text. Breaking our cipher has become very hard and impossible in some cases, as ourencryption algorithm produces different cipher text for the same plain text at different times. Also, if the key size is increased, the robustness of this cryptosystem will still be improved.

7. Conclusion

We have proposed new augmented, light-weighted, homomorphic cryptosystem for big data. The new proposed algorithm is proved to be homomorphic over addition, subtraction and multiplication operations. Evaluation of the new proposed algorithms are done by repeating experiments using four different data sizes and two key sizes. A fair comparison is done between my algorithm with Goldwasser-Micali, ElGamaland Paillier cryptosystems. From the test results, it can easily be realized that our algorithms showed better performance among the other algorithms as this new algorithm we developed is relatively light-weighted, but still robust towards cryptanalysis.

A vision of adding an efficient multilevel user authentication mechanism using graphical password and biometric images for analysing big data is under design stage.

8. Acknowledgment

We wholeheartedly thank VIT, Vellore, Tamilnadu, India for providing us the required lab facilities to carry out the experiments.

References

[1] Sharma, S. Rise of Big Data and related issues. In Proceedings of the 2015 Annual IEEE India Conference (INDICON), New Delhi, India, 17–20 December 2015; pp. 1–6.

[2] Eynon, R. The rise of Big Data: What does it mean for education, technology, and media research? Learn. Media Technol. 2013, 38, 237–240.

[3] Gang Zeng, “Big Data and Information Security”, International Journal of Computational Engineering Research (IJCER), ISSN (e): 2250 – 3005, Volume, 05, Issue, 06, June – 2015|

[4] A. Sahai, and B. Waters, "Fuzzy identity-based encryption," Advances in Cryptology–EUROCRYPT 2005, LNCS, vol. 3494, pp. 457-473, 2005.

[5] Yoshiko Yasumura, Hiroki Imabayashi, HayatoYamana, “Attribute-based Proxy Re-encryption Method for Revocation in Cloud Data Storage”, 2017 IEEE International Conference on Big Data (BIGDATA)

[6] Seonyoung Park and Youngseok Lee, “A Performance Analysis of Encryption in HDFS,”

May - June 2020
ISSN: 0193-4120 Page No. 3511-3520

Journal of KISS: Databases, Vol.41, Issue.1, 2014, pp.21-27

[7] Byeong-yoon Choi. "Design of Cryptographic Processor for AES Rijndael Algorithm," The Journal of The Korean Institute of Communication Sciences, Vol.26, Issue.10, 2001, pp.1491-1500

[8] Yong Kuk Cho, Jung Hwan Song, and Sung Woo Kang, "Criteria for Evaluating Cryptographic Algorithms based on Statistical Testing of Randomness," Journal of the Korea Institute of Information Security and Cryptology, Vol.11, Issue.6,2001,pp.67-76.

[9] N. I. of Standards and Technology, "STANDARDS AND GUIDELINES TESTED UNDER THE CAVP)," http://csrc.nist.gov/groups/STM/cavp/standards.html/.

[10] ARIA Development Team, Block Encryption Algorithm ARIA [Internet], http://glukjeoluk.tistory.com/attachment/ok110000000002.pdf.

[11] Korea Internet & Security Agency, ARIA specification [Internet], http://seed.kisa.or.kr/iwt/ko/bbs/EgovReferenceDetail.do?bbsId=BBSMSTR_000000000002&nttId=39&pageIndex=1&searchCnd=&searchWrd =.

[12] Youngho Song, Young-Sung Shin, Miyoung Jang, Jae-Woo Chang, Design and Implementation of HDFS Data Encryption Scheme using ARIA Algorithm on Hadoop, IEEE BigComp, 2017

[13] Abdullah Al Mamun, Khaled Salah, Somaya Al-maadeed, and Tarek R. Sheltami, BigCrypt for Big Data Encryption, Fourth International Conference on Software Defined Systems (SDS), 2017

[14] C. SANTA CLARA, "Success of phishing attacks with 80 percent of business users unable to detect scams," http://www.mcafee.com/us/about/news/2014/q3/20140904-01.aspx", September 4, 2014.

[15] Wikipedia, "RSA (cryptosystem)," http://en.wikipedia.org/wiki, 2015, [Online; accessed 14-April-2015].

[16] J. Daemen and V. Rijmen, The design of Rijndael: AES-the advanced encryption standard. Springer Science & Business Media, 2002.

[17] X. Yi et al., Homomorphic Encryption and Applications, Springer Briefs in Computer Science, DOI 10.1007/978-3-319-12229-8__2

[18] S. Goldwasser, S. Micali, "Probabilistic encryption and how to play mental poker keeping secret all partial information", in Proceedings of 14th Symposium on Theory of Computing, 1982, pp. 365–377

[19] T. ElGamal, A public-key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theory 31(4), 469–472 (1985)

[20] P. Paillier, Public key cryptosystems based on composite degree residue classes, Proceedings of Advances in Cryptology, EUROCRYPT'99, 1999, pp. 223–238

[21] D. Boneh, E. Goh, K. Nissim, Evaluating 2-DNF formulas on cipher texts, in Proceedings of Theory of Cryptography, TCC'05, 2005, pp. 325–341

[22] https://hadoop.apache.org/

[23] https://hive.apache.org/