THE INTERNATIONAL JOURNAL OF SCIENCE & TECHNOLEDGE

A Fused Homomorphic Encryption Technique to Increase Secure Data Storage in Cloud Based Systems

Desamsetti Harshith
Student, Department of CSE, KL University, Andhra Pradesh, India

Abstract:
As of now, data is everywhere, irrespective of useful or useless depending on the requirement, the whole data can be continuously updated to cloud to reduce maintenance costs and hardware. As it involves people to handle the data flow or to perform main operations, there is no guarantee of integrity and consistency of the data. Using homomorphic encryption scheme, computations can be easily applied on the data without which cannot be applied by the Data Centers, they can just maintain the cloud but not analyze the data because of the encryptions. Hence usage of an efficient technique such as homomorphic encryption is much needed as it guarantees integrity of the data by not decrypting it, performing operations on the encrypted data itself.

Keywords: *Homomorphic, cryptography, cloud, security*

1. Introduction

1.1. Homomorphic Encryption

The word Homomorphism comes from ancient Greek language, homos meaning “same” and morph meaning “form” or “shape”. Homomorphic encryption is a form of encryption that allows to be carried out on cipher text, thus generating an encrypted result which, when decrypted, matches the result of operations performed on the plain text.

$m1 * m2 \rightarrow n$
$C1 \# C2 \rightarrow d$
$enc(mi) \rightarrow Ci$
$dec(d) \rightarrow n$

Where m1 and m2 are the messages, C1 and C2 are the encrypted messages, enc () and dec () are the encryption and decryption operations respectively. [1]

Here the messages m1 and m2 are being applied a multiplicative operation, where the output is n.

This is the general answer the customer is looking for, whereas the cloud operation will be as C1 between C2 which gives the output d. To get the Ci, enc () function should be applied on mi. To get the output, dec () function should be applied on n.

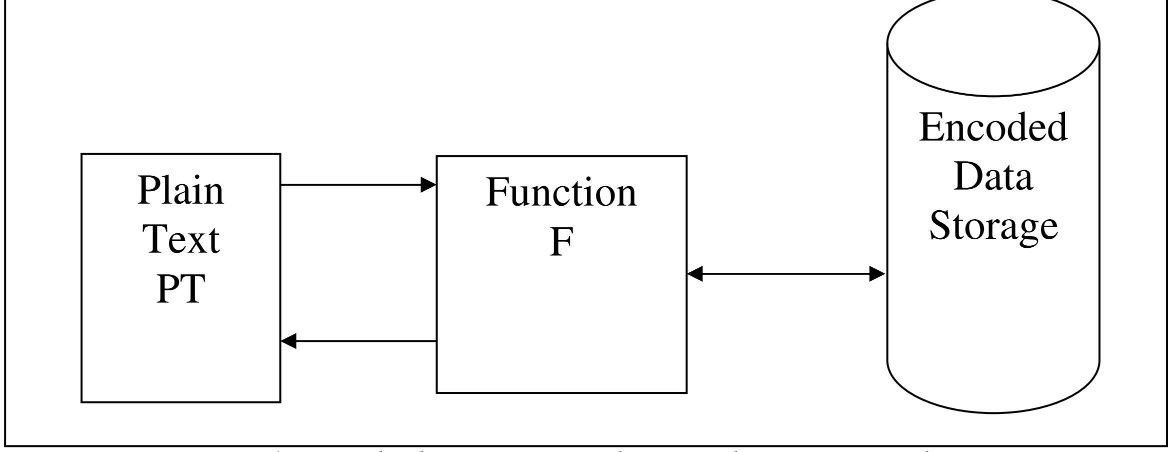

Figure 1: Basic cloud operations using homomorphic encryption technique

1.1.1. Types of Homomorphic Encryption Types

There are three types of Homomorphic encryption techniques. [2]
i) Somewhat Homomorphic
ii) Partially Homomorphic
iii) Fully Homomorphic

These Homomorphic encryption techniques differ based on the of operations each technique can perform.
i) Somewhat Homomorphic:-
In Somewhat Homomorphic encryption, the operations performed are either multiplicative homomorphic or additive homomorphic, but not both, that means it can only apply either multiplication operation on data or addition on data, but not both operations
ii) Partial Homomorphic: -
In Partially Homomorphic encryption, either additive or multiplicative operations are done at once but not both.
iii) Fully Homomorphic:-
In Fully Homomorphic encryption, arbitrary number of operations can be performed, that means any number of additive or multiplicative operations can be done.

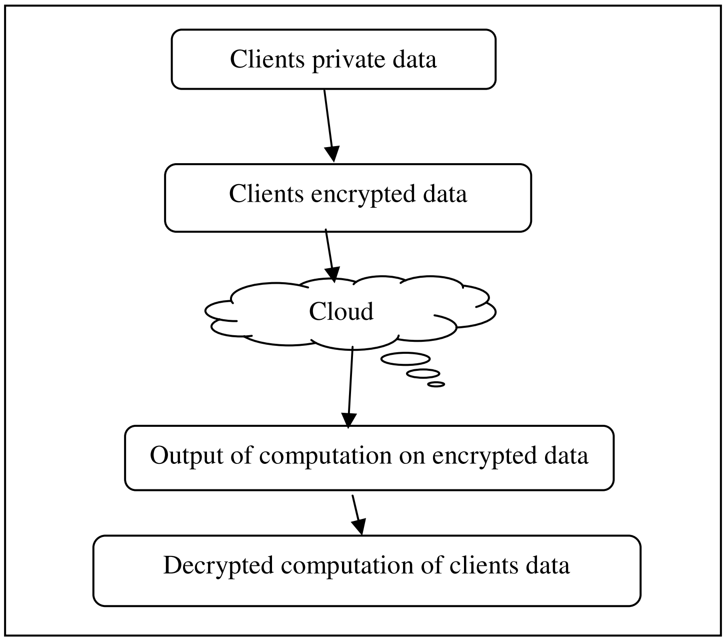
Clients private data
Clients encrypted data
Cloud
Output of computation on encrypted data
Decrypted computation of clients data

Figure 2: Homomorphic Encryption in Cloud

2. Literature Survey

The basic homomorphic encryption technique was created by Rivest, Shamir and Alderman (RSA) which is multiplicatively homomorphic and was published in 1977. { i.e., given Encryptions E(m1),…,E(mt) of m1,…,mt, one can efficiently compute a compact cipher text that encrypts f(m1,…,mt) for any efficiently computable function f. This problem was posed by Rivest et al. in 1978.
Zvika Brakersky, Gentry and Vaikuntanathan scheme proposed the construction of FHE without using the Gentry's bootstrapping procedure in their paper "Leveled Fully Homomorphic encryption without bootstrapping".[7]
Gorti VNKV Subba Rao proposed Enhanced Homomorphic Encryption Scheme for Homomorphic encryption/decryption.

3. Theoretical Analysis

3.1. RSA Encryption

RSA algorithm was the basic Homomorphic encryption algorithm which was implemented by Rivest, Shamir and Alderman, published in 1977. RSA is a Somewhat Homomorphic encryption technique which can be used to apply multiplications on data.

3.2. EHES Encryption

Gorti Subba Rao from India created Enhanced version of the old Homomorphic encryption (EHES) with IND CCA system. Comparatively, EHES has been much faster and efficient than RSA. The encryption time and cipher text size are relatively efficient in EHES than in RSA. [5]
Even though by using Homomorphic encryption techniques we can provide security, there are challenges involved in using it, they are:

Partial Homomorphic Encryption techniques are effective when they are applied practically [3]. Be that as it may, these frameworks have certain limitations because most support either multiplication or addition. For practical applications, to be more efficient, these are used along with Homomorphic applications to make more supportive and efficient applications secure.
The efficiency of Homomorphic encryption systems is also dependent on the size of the keys used. Whenever a large size key is used, the system takes more time to apply operations. The usage of large size key makes Homomorphic encryption systems secure too.[7] On the other hand, by using large size key for Homomorphic encryption, the encryption, decryption time are effected

4. Experimental Investigations

4.1. Properties of Homomorphic Encryption
The main properties of Homomorphic technique are Additive and Multiplicative. [7][8]

4.1.1. Additive Homomorphic: A Homomorphic technique is additive, if
Encode (P1 + P2) = Encode (P1) + Encode (P2)

When the output is decrypted, it is same as if the addition is applied on the decrypted text itself.

4.1.2. Multiplicative Homomorphic: Homomorphic technique is multiplicative, if
Encode (P1 X P2) = Encode (P1) X Encode (P1)
When the output is decrypted, it is same as if the multiplication is applied on the decrypted text itself.

4.2. Operations Performed by Homomorphic Encryption
Homomorphic encryption contains of 4 parts.
- Generation of the Key
- Encoding
- Evaluation
- Decoding

4.2.1. Generation of the Key
In this part, the primary part is to create a couple of keys which are Public key and Private key, utilized for the encoding and decoding processes.

4.2.2. Encoding
In this part, the customer will encode the data utilizing the private key and generates the cipher text. This cipher text alongside public key in general will be sent to the server.

4.2.3. Evaluation
In this part, assume ordinarily in cloud storage, the operations to be performed in the form of services can be applied on the encoded data. This implies the required function is evaluated on the encoded data. Hence no need of decryption of the data here.

4.2.4. Decoding
In this part, when the customer retrieves the operated and encoded output of the data, decoding is applied on it utilizing his security key produced at first part. Here security can be assured in case of cloud service providers to the customers.
Sub tags used in the procedure,
- Plaintext

The understandable message which will be changed over into an encoded message.
- Ciphertext

A message in encoded form, converted from a plain text.
- Key

A value used in the encoding and decoding processes.
- Cryptosystem

A system to encode and decode data.
- Symmetric Cryptosystem

A cryptosystem that uses the same key to encode and decode data.
- Asymmetric Cryptosystem

A cryptosystem that uses one key to encode and a different key to decode data.
- Cryptography

The utilization of cryptosystems to maintain the confidentiality of data.
- Cryptanalysis

The study of breaking into cryptosystems and data which is secured.

5. Experimental Results

5.1. Enhanced Homomorphic Encryption System

5.1.1. Generation of Keys

Refined Algorithm representation is as follows, [4]
Choose two prime numbers a=3 and b=5
Compute n=a*b=3*5=15
Public Key is P_k= n=15
Private Key is S_k= (a, b) = (3,5)

5.1.2. Encryption

Refined Algorithm representation is as follows,
The encryption is Enc (m, $P_{k,}$ Sk)
Generate a random number r=10
Compute c=m+r*(a^b) mod(n)
c_1=2+ 10 *(3^5)mod(15)=32
where m=2 and r=10
c_2=3+ 10 *(3^5)mod(15)=33
where m=3 and r=10
$c_{1*}c_2$ = 32*33=1056

5.1.3. Decryption

Refined Algorithm representation is as follows,
For multiplication,
c_{1*} c_2 mod(n)= 1056 mod (15)=6
For addition,
c_1 + c_2 mod(n)= 65mod(15)=5

6. Discussion of Results

As we discussed that cloud computing is facing a problem regarding the network security. Homomorphic encryption is used to ensure security in cloud based systems. In homomorphic encryption the data is altered or modified in the form of cipher text only. The other users cannot identify the data which is altered in the cloud. Homomorphic encryption is the best method to increase the security in cloud based systems. We discussed two types of encryption schemes. RSA encryption schema in this we can perform only multiplication operations to the data. So it is somewhat homomorphic.
EHES encryption in this we can perform both addition and multiplication operations on the data, so it is fully homomorphic [6]. When we compare both these RSA algorithm and EHES algorithm we can conclude that EHES which is fully homomorphic is more efficient and advanced encryption scheme. The time complexity of EHES is less than the RSA algorithm.
Most of the cloud based systems use EHES Algorithm as it is more efficient.

7. Conclusion

Nowadays every internet or network based systems are striving for security. So security has an important role in the present generation. We also here that cloud is the present trend where all the network users are going for cloud computing. The use of cloud computing has been increased. In the same way the security to cloud also has been increased. Data security in cloud is one of the problems faced by the cloud. To ensure data security in cloud based systems we go for using Homomorphic Encryption. In homomorphic encryption technique the operations are performed on the cipher text only, so that there is no way to data leakage.
We included two types of encryption techniques like RSA and EHES techniques.
Both the techniques are the best encryption schemes for ensuring security to the data in the cloud.

8. References

i. Craig Gentry, “A fully homomorphic encryption scheme”, PhD thesis, Stanford University,2009.
ii. “An Architecture for Parallelizing Fully Homomorphic Cryptography on Cloud” by Ryan Hayward, Chia-Chu Chiang in 2013 Seventh International Conference on Complex, Intelligent, and Software Intensive Systems IEEE.
iii. “Challenges of Using Homomorphic Encryption to Secure Cloud Computing” by Khalid El Makkaoui, Abderrahim Beni Hssane in Cloud Technologies and Applications (CloudTech), 2015 International Conference on June 2015 IEEE.
iv. “A Algorithm of Fully Homomorphic Encryption” by Guangli Xiang,Benzhi Yu,Ping Zhu in 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2012).
v. Zvika Brakerski and Vinod Vaikuntanathan,” Fully homomorphic encryption from ring-LWE and security for key dependent messages”,CRYPTO 2011,Vol.6841,2011,pp.501-510.

vi. "A Hybrid Scheme of Public-Key Encryption and Somewhat Homomorphic Encryption" by Jung Hee Cheon, Jinsu Kim in IEEE Transactions on Information Forensics and Security IEEE May 2015.

vii. Zvika Brakerski,Craig Gentry, Vinod Vaikuntanathan, "(Leveled)Fully homomorphic encryption without bootstrapping",ITCS,2012,pp.309-325.

viii. C.Gentry and S.Halevi," Implementing gentry's fully homomorphic encryption scheme",EUROCRYPT,2011.