ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 11, November 2014

Performance of Ring Based Fully Homomorphic Encryption for securing data in Cloud Computing

S.Hemalatha[1], Dr. R.Manickachezian[2]

Ph.D Research Scholar, Research Department of Computer Science, N.G.M College, Pollachi, India[1]

Associate Professor, Research Department of Computer Science, N.G.M College, Pollachi, India[2]

Abstract: Cloud Computing is a promising paradigm which has become most recent research due to its ability to reduce the costs associated with computing. Securing data is very important because of the critical nature of cloud computing and the large amounts of compound data it carries . This security gap is bridged by the most popularly used encryption techniques for encrypting the tenuously stored data . Fully Homomorphic Encryption is a good basis to enhance the security measure of untrusted systems or applications that stores and manipulates sensitive data. In this paper, the design and implementation of a Fully Homomorphic Encryption (FHE) scheme based on Ring is reported.

Keywords : Cloud Computing, Homomorphic encryption, Data Security, Data Storage.

I. INTRODUCTION

Homomorphic encryption is the conversion of data into cipher text that can be analyzed and worked with as if it were still in its original form. Homomorphic encryptions allow complex mathematical operations to be performed on encrypted data without compromising the encryption. In mathematics, homomorphic describes the transformation of one data set into another while preserving relationships between elements in both sets. The term is derived from the Greek words for "same structure." Homomorphic encryption is expected to play an important part in cloud computing, allowing companies to store encrypted data in a public cloud and take advantage of the cloud provider's analytic services.

In encryption schemes, Bob encrypts a plaintext message to obtain a ciphertext. Alice decrypts the ciphertext to recover the plaintext. In Fully Homomorphic Encryption[2], parties that do not know the plaintext data can perform computations on it by performing computations on the corresponding ciphertexts. A major application of FHE is to cloud computing.

The main aim of Homomorphic encryption is to enhance the security of cloud computing. The data should be encrypted and sent to the cloud. After it is send, the computations are made on the encrypted data and the result of this computation is an encrypted data too. If the result of the computation is decrypted, then the plain text version of the result is got back. The question is, why it is assumed that it enhances the cloud computing? The data is just encrypted and sent to cloud. When the computations are to be made , these data are queried to the computers and decrypted, made computations, then send back to the cloud if needed. Homomorphic encryption in and of itself solves the problem of computation on encrypted data. For example, if the encryptions of a and b are given as(E(a),E(b))then E(ab) and E(a+b) can be computed. That is the problem that fully homomorphic encryption (FHE) solves. There are also partially homomorphic encryption methods that allow you to only compute one operation (e.g., Paillier and Elgamal).

By far the most popular request was for some background on the recent results is computing on encrypted data, or 'Fully-Homomorphic Encryption'.

II. RELATED WORK

In 1978, shortly after the invention of the RSA cryptosystem, Rivest, Adleman, and Dertouzos [RAD][6] came up with the idea of fully homomorphic encryption, which they called "privacy homomorphisms". Their paper states, "although there are some truly inherent limitations on what can be accomplished, we shall see that it appears likely that there exist encryption functions which permit encrypted data to be operated on without preliminary decryption of the operands, for many sets of interesting operations. These special encryption functions we call 'privacy homomorphisms'; they form an interesting subset of arbitrary encryption schemes". Despite the optimism of Rivest, Adleman, and Dertouzos[6], fully homomorphic encryption remained out of reach for many years.

Homomorphic encryption schemes that allow simple computations on encrypted data have been known for a long time. For example, the encryption systems of Goldwasser and Micali [11], El Gamal [12], Cohen and Fischer , and Paillier[20] support either adding or multiplying encrypted ciphertexts, but not both operations at the same time. Boneh, Goh and Nissim [21] were the first to construct a scheme capable of performing both operations at the same time – their scheme handles an arbitrary number of additions but just one multiplication. More recently, in a breakthrough work, Gentry [9,10] constructed a fully homomorphic encryption scheme (FHE) capable of evaluating an arbitrary number of additions and multiplications (and thus, compute any function) on encrypted data.

Aside from Gentry's scheme (and a variant thereof by Smart and Vercauteren [7] and an optimization by Stehle and Steinfeld [8]), there are two other fully homomorphic encryption schemes [19, 18].

III.THE HOMOMORPHIC ENCRYPTION ALGORITHM

The three basic requirements for a practical Homomorphic Encryption scheme are

- Versatility
- Speed
- Ciphertext Size

By checking the partial HE schemes and the FHE schemes against the three requirements, the following results are tabulated

TABLE I PARTIAL HE SCHEMES VS. FHE SCHEMES

Type	Versatility	Speed	Ciphertext Size
Partial HE	Low	Fast	Small
Fully HE	High	Slow	Large

Homomorphic encryption allows complex mathematical operations to be performed on encrypted data without revealing the contents of the original plain data. For plaintexts P1 and P2 and corresponding ciphertext C1 and C2, a homomorphic encryption scheme permits meaningful computation of P1 ϴ P2 from C1 and C2 without revealing P1 or P2.The cryptosystem is additive or multiplicative homomorphic depending upon the operation ϴ which can be addition or multiplication.
A homomorphic encryption scheme consists of the following four algorithms:

KeyGen (λ):
- Input-the security parameter λ.
- Output-a tuple (sk ,pk) consisting of the secret key sk and public key pk .

Encrypt (pk ,π):
- Input-a public key pk, and a plaintext π .
- Output-ciphertext ψ.

Decrypt (sk , ψ):
- Input-a secret key sk and a ciphertext ψ .
- Output-the corresponding plaintext π .

Evaluate (pk ,C , ψ):
- Input-a public key pk , a circuit C with t inputs (of the set C of allowed circuits) and a set ψ of t ciphertext $\psi_1 , ... \psi_t$.
- Output-a ciphertext ψ.

Therefore, a homomorphic encryption scheme consists of all algorithms of a conventional publickey encryption scheme and an extra one. The correctness-condition for the conventional part of a homomorphic encryption scheme is identical to that of a (non-homomorphic) public key encryption scheme.

The additional algorithm Evaluate is supposed to do the following:
If is a ciphertext ψ_i corresponding to the plaintext π_i for i= 1 … t and $\psi= (\psi_1, \psi_2 . . . . \psi_t)$, then Evaluate (pk ,C , ψ) shall return a ciphertext ψ corresponding to the plaintext $C(\pi_1, ... , \pi_t)$ for a circuit C with t inputs.
A homomorphic encryption scheme is said to correctly evaluate C(a set of circuits), if the correctness-condition on the algorithm Evaluate from above holds for all circuits c∈C .

IV. FULLY HOMOMORPHIC ENCRYPTION SCHEME

Fully homomorphic encryption allows the addition and multiplication of encrypted data without the need for full decryption. Cloud computing technology[13] has rapidly evolved over the last decade, offering an alternative way to store and work with large amounts of data. However data security[16,17] remains an important issue particularly when using a public cloud service provider. Homomorphic encryption could address this need. A significant amount of research on homomorphic cryptography has appeared in the literature over the last few years.

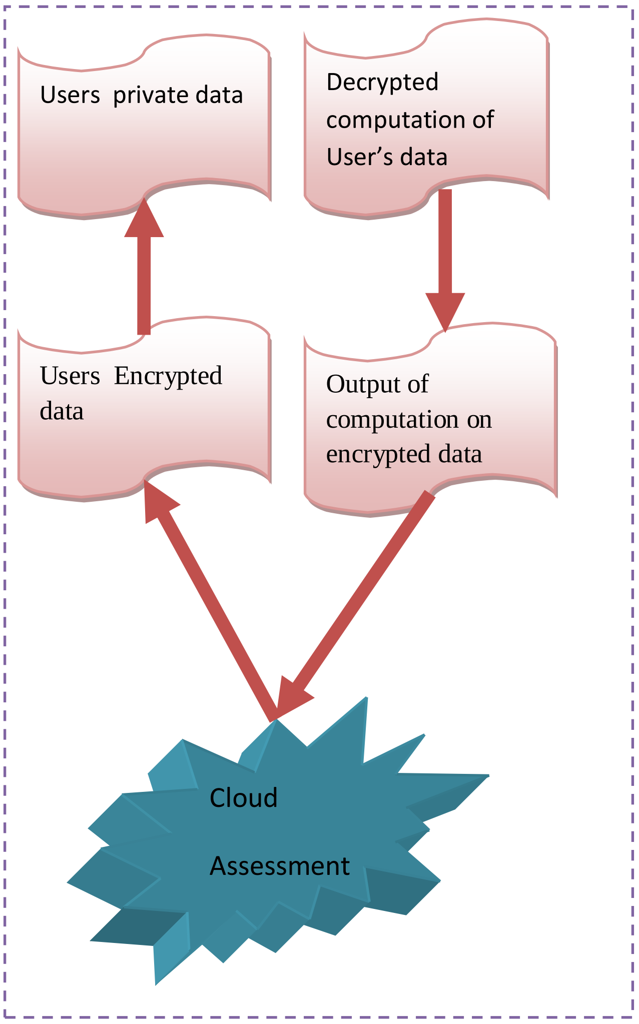
Users private data
Decrypted computation of User's data
Users Encrypted data
Output of computation on encrypted data
Cloud
Assessment

Fig: 1 Implementing FHE algorithm into Cloud Environment

Applying fully homomorphic encryption[1],the data security can be assured for cloud computing. The key concept is that the data is encrypted by homomorphic encryption and stored in cloud server, through which it is highly benefited. Dealing with the cipher text directly in server the data security can be assured because anyone else who doesn't know the key can't decrypt it. The homomorphic symmetric encryption [2] is used to construct the data secure scheme.

The symmetric homomorphic encrypt scheme:

- Select encrypt parameter: r, p and q, $r \sim 2^n$, $p \sim 2^{n2}$, $q \sim 2^{n5}$ and p is prime

P is the secret key

- Encrypt: for plain text m

Compute c=pq+2r+m where c is the cipher text

- Decrypt: m=(c mod p) mod 2
- Correctness: because pq is larger than 2r+m so (c mod p) =2r+m

Finally (c mod p) mod 2= (2r+m) mod 2=m

Homomorphic: for two cipher text

$C_1=q_1p+2r_1+m_1$

$C_2=q_2p+2r_2+m_2$

Compute:
$C_1+C_2=(q_1+q_2)\ p+2(r_1+r_2)+m_1+m_2$
So if $2(r_1+r_2)+m_1+m_2<<p$
Then $(C_1+C_2) \bmod p=2(r_1+r_2)+m_1+m_2$
So, its additive homomorphic.
And $C_1*C_2=[q_1*q_2p+(2r_1+m_1)+(2r_2+m_2)]\ p+2(2r_1 r_2+r_1m_1+r_2 m_1)+m_1 m_2$
So if $2(2r_1 r_2+r_1m_1+r_2 m_1)+m_1 m_2<<p$
Then $(C_1*C_2) \bmod p=2(2r_1 r_2+r_1m_1+r_2 m_1)+m_1 m_2$
So, it is multiplicative homomorphic.

Applying above encrypt scheme, the following cloud data secure scheme is designed:

As shown in the below figure, the scheme uses symmetric homomorphic encryption to enhance data security[14]. First, the user login and the server and assign a key-generation seed to user.Then, user generate the secret key at client using this seed. So the server don't know the secret key at all. This procedure can be repeated, and it enables the user to get the same secret key at any time. Secondly, the user can use this key to encrypt data[15] which the user wants to transmit and save it in the cloud server.

While transmitting the data, also the other cryptographic technology such as digital signature can be applied to assure the integrity and non repudiation. At last, the user can send request to cloud server (also encrypted) and the server do the operation even without knowing the content of the operation. With this scheme, not only the stored data but also the transmitted data is encrypted, so it is not worried whether the data is dropped or stolen. It also can provide secure data audit service because the third audit party can deal with the encrypted data directly.

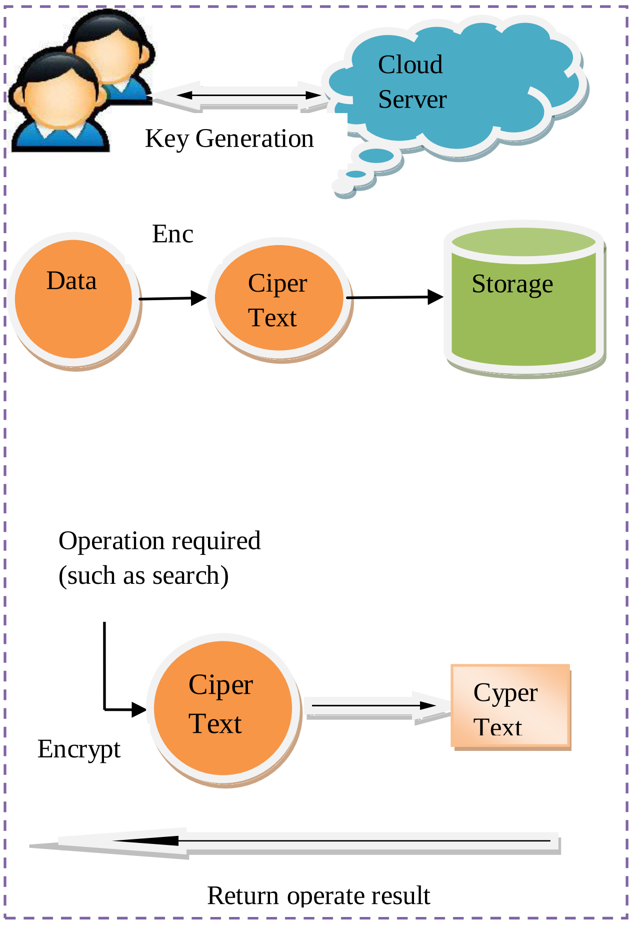

Fig: 2 The data security scheme for cloud computing

V. RING BASED FULLY HOMOMORPHIC ENCRYPTION

A ring R is an abelian group with a multiplication operation $(a,b) \to ab$ that is associative and satisfies the distributive laws: $a(b+c)=ab+ac$ and $(a+b)c = ac+bc$ for all $a,b,c \in R$. The most important structure is the ring R. Let d be a positive integer and define $R = Z[X]/(\Phi_d(X))$ as the ring of polynomials with integer coefficients modulo the d- th cyclotomic polynomial $\Phi_d(X) \in Z[X]$. The degree of Φ_d is $n = \phi(d)$, where ϕ is Euler's totient function. The elements of R can be uniquely represented by all polynomials in Z[X] of degree less than n. Arithmetic in R is arithmetic modulo $\Phi_d(X)$, which is implicit whenever the terms or equalities involving elements in R is written. An arbitrary element $a \in R$ can be written as......
$a=\sum_{i=0,}^{n-1} a_i x^i$ with $a_i \in Z$ and *a* is identified with its vector of coefficients $(a_0,a_1,...,a_{n-1})$. In particular, *a* can be viewed as an element of the R-vector space R^n. The maximum norm on R^n is chosen to measure the size of elements in R. The maximum norm of *a* is defined as $\|a\|_\infty = \max_i\{|a_i|\}$.

Basic Scheme

In this section, the basic public key encryption scheme[1] is described which is the foundation for the leveled schemes . The scheme is parameterized by a modulus *q* and a plaintext modulus $1 < t < q$.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 11, November 2014

Ciphertexts are elements of $R = Z[X]/(\Phi_d(X))$ and plaintexts are elements of R/tR [11]. Secret keys and errors are generated from different distributions, for example Gaussian distributions of different width. The secret key is derived from the distribution χ_{key}, and errors are sampled from the distribution χ_{err}. The "Regev-style" encryption is used as in [2] and [3]. The scheme consists of the following algorithms.

• Basic.ParamsGen(λ): Given the security parameter λ, fix a positive integer d that determines R, moduli q and t with $1 < t < q$, and distributions χ_{key},χ_{err} on R. Output $(d,q,t,\chi_{key},\chi_{err})$.
• Basic.KeyGen($d,q,t,\chi_{key},\chi_{err}$): Sample $f',g \leftarrow \chi_{key}$ and let $f = [tf' +1]_q$. If f is not invertible modulo q, choose a new f'. Compute the inverse $f^{-1} \in R$ of f modulo q and set $h = [tg^{f-1}]_q$. Output the public and private key pair $(pk,sk) = (h,f) \in R^2$.
• Basic.Encrypt(h,m): The message space is R/tR. For a message m + tR, choose $[m]_t$ as its representative. Sample $s,e \leftarrow \chi_{err}$, and output the cipher text $c = [[q/t][m]_t + e + hs]_q \in R$.
• Basic.Decrypt(f,c): To decrypt a ciphertext c, compute $m=[[t/q.[fc]_q]]_t \in R$.
There are various lemma's stated where the messages are referred as an element *m* in the ring R although the message space is R/tR, keeping in mind that encryption always takes place on the representative $[m]_t$ and that by decrypting, all that can be recovered is *m* modulo *t*.

VI. PERFORMANCE

The implementation of Gentry's[5] fully homomorphic encryption scheme with the model of several dimensions are tested, corresponding to several security levels. From a "toy" setting in dimension 512, to "small," "medium," and "large" settings in dimensions 2048, 8192, and 32768, respectively. The public-key size ranges in size from 70 Megabytes for the "small" setting to 2.3 Gigabytes for the "large" setting. Performance of Fully Homomorphic Encryption with the parameters such as dimensions, key generation time, size of the public key is tabulated. The table II shows the results for the parameter-setting that are used to generate the public challenges [4].

TABLE II RESULTS FOR PARAMETER SETTING

Dimension	KeyGen	PK Size	AND/Dec
2048 800,000-bit integers	40 Sec	70 MByte	31 sec
8192 3,200,000-bit integers	8 min	285 MByte	3 min
32768 13,000,000-bit integers	2 hrs	2.3 GByte	30 min

TABLE III DEPENDENCE OF BIT LENGTHS OF MODULI Q_I, AS A FUNCTION OF RING DIMENSION FOR P = 2

Ring Dimension *n*	Bit length $\log_2(q_i)$
512	44
1024	45
2048	47
4096	48
8192	50
16384	51

The table III shows how many bits are required to represent $q_1,...,q_t$ for varying ring dimensions for p = 2. Note that all $q_1,...,q_t$ can be represented in less than 64 bits.

VII. CONCLUSION

The Ring based fully homomorphic encryption has been proposed for securing data in cloud computing. Here the homomorphic encryptions allow the complex mathematical operations to be performed on encrypted data without compromising the encryption based on Ring Structure. The parameters and implementation results of Ring based fully homomorphic encryption are presented in this work.

REFERENCES

[1]. JoppeW. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In Martijn Stam, editor, Cryptography and Coding, volume 8308 of Lecture Notes in Computer Science, pages 45–64. Springer Berlin Heidelberg, 2013.
[2]. Z. Brakerski. Fully homomorphic encryption without modulus switching from clas- sical GapSVP. In Advances in Cryptology - Crypto 2012, volume 7417 of Lecture Notes in Computer Science, pages 868–886. Springer, 2012.
[3]. J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. http://eprint.iacr.org/.
[4]. C. Gentry and S. Halevi. Public Challenges for Fully-Homomorphic Encryption. TBA, 2010.
[5]. Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, STOC, pages 169–178. ACM, 2009.
[6]. R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. In Foundations of secure computation, volume 4, pages 169–180. New-York: Academic Press, 1978.
[7] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In Phong Q. Nguyen and David Pointcheval, editors, Public Key Cryptography, volume 6056 of Lecture Notes in Computer Science, pages 420–443. Springer, 2010.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 11, November 2014

[8] Damien Stehl´e and Ron Steinfeld. Faster fully homomorphic encryption. In Masayuki Abe, editor, ASIACRYPT, volume 6477 of Lecture Notes in Computer Science, pages 377–394. Springer, 2010.
[9] Craig Gentry. A fully homomorphic encryption scheme. PhD thesis, Stanford Univer- sity, 2009. crypto.stanford.edu/craig.
[10] Craig Gentry. Toward basing fully homomorphic encryption on worst-case hardness. In CRYPTO, pages 116–137, 2010.
[11] Julien Bringe and al. An Application of the Goldwasser-Micali Cryptosystem to Biometric Authentication, Springer-Verlag , 2007.
[12] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory, 469-472, 1985.
[13] S.Hemalatha, Dr.R.Manickachezian, "Present and Future of Cloud Computing: A Collaborated Survey Report"., International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-1, Issue-2, July 2012.
[14] S.Hemalatha, Dr.R.Manickachezian, "Implicit Security Architecture Framework in Cloud Computing Based on Data Partitioning and Security Key Distribution"., International Journal of Emerging Technologies in Computational and Applied Sciences, ISSN (Online): 2279-0055 , pp. 76-81, Feb.2013.
[15] S.Hemalatha, Dr.R.Manickachezian " Dynamic Auditing Protocol using Improved RSA and CBDH for Cloud Data Storage".,International Journal of Advanced Research in Computer Science and Software Engineering Volume 4, Issue 1, January 2014.
[16] S.Hemalatha, Dr.R.Manickachezian, "Security Strength of RSA and Attribute Based Encryption for Data Security in Cloud Computing"., International Journal of Innovative Research in Computer and Communication Engineering Vol. 2, Issue 9, September 2014
[17] S.Hemalatha, Dr.R.Manickachezian , "An Efficient RSA Based Data Integrity Schema in Secured Cloud Storage ",Research Journal of Applied Sciences, Engineering and Technology , Maxwell Scientific Organization, 2014
[18] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. FOCS, 2011.
[19] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In EUROCRYPT 2010, pages 24–43.
[20] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In EUROCRYPT, pages 223–238, 1999.
[21] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Theory of Cryptography - TCC'05, volume 3378 of Lecture Notes in Computer Science, pages 325–341. Springer, 2005.

BIOGRAPHIES

S.Hemalatha Raguram (04/01/1980) received her B.Sc Mathematics and Master of Computer Applications from NGM College, Pollachi, Coimbatore, India. She completed her Master of Philosophy in Bharathiar University, Coimbatore. Presently she is working as an Assistant Professor in the Department of Computer Applications in NGM College (Autonomous), Pollachi. She has published 19 papers in various International Journals and Conferences. Her area of interest includes cloud computing, Object Oriented Analysis and Design and Data Mining. Now she is pursuing her Ph.D Computer Science in Dr. Mahalingam Center for Research and Development at NGM College, Pollachi.

Dr. R. Manickachezian (05/06/1965) received his M.Sc Applied Science from PSG College of Technology, Coimbatore, India in 1987. He completed his M.S. degree in Software Systems from Birla Institute of Technology and Science, Pilani, Rajasthan, India and Ph.D degree in Computer Science from School of Computer Science and Engineering, Bharathiar University, Coimbatore. He served as a Faculty of Maths and Computer Applications at P.S.G College of Technology, Coimbatore from 1987 to 1989. Presently, he is working as an Associate Professor of Computer Science in NGM College (Autonomous), Pollachi. He has published 150 papers in various International Journals and Conferences. He is a recipient of many awards like Desha Mithra Award and Best paper awards. His research focuses on Network Databases, Data Mining, Network Security, Bio-Informatics and Distributed Computing.