IOSR Journal of Engineering (IOSRJEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
Vol. 07, Issue 10 (October. 2017), ||V2|| PP 20-29

Enhancing Data Security in Cloud Computation Using Addition-Composition Fully Homomorphic Encryption Scheme

Omollo R.O.[1], Raburu G.O.[2], Omolo-Ongati N.[3], Okelo B.N.[4]
[1,2]*Department of Computer Science and Software Engineering*
[3,4]*Department of Pure and Applied Mathematics*
Jaramogi Oginga Odinga University of Science and Technology
P.O. Box 210-40601, Bondo-Kenya.

Abstract: -This paper focusses on the use of homomorphic encryption schemes in solving data security in cloud computation. We started by acknowledging Gentry's contribution on partial homomorphic encryption schemes where he constructed a somewhat homomorphic encryption based on ideal lattices using both additive and multiplicative Homomorphisms. We also noted improvement made on his construction by Dijkwherehe used integers instead of ideal lattices. Again we appreciated efforts by Braskerski based on Learning with Errors (LWE) and Ring Learning with Errors (Ring LWE) problems. Besides all these efforts, the schemes still had strain on the computing storage and processing resources. We discovered that there is need to constructan encryption scheme which lessens the computational strain on the computing assets. Here, we found remedy in the use of both addition and composition operations to implement a fully homomorphic encryption scheme. A mathematical theory was translated into an algorithm implementable JAVA. We testediton a single hardware with minimum specifications: 1.5GHz, 4GB RAM and Windows 10 OS. The results showed that the new scheme supports faster encryption process, larger ciphertext sizes and is versatile. This new addition-composition fully homomorphic scheme enhances data security in cloud computation thus boosts consumer confidence in the cloud services and applications.

Keywords: -Addition-Composition, Cloud Computing, Data Security, Fully Homomorphic Encryption, Homomorphic Encryption Scheme.

Date of Submission: 16-10-2017 Date of acceptance: 02-11-2017

I. INTRODUCTION

1.1 Cloud Computing Definition

Cloud Computing has become the emergent technological term common with computing data storage and offsite computing resource management. A recent study on its definition reviewed twenty-two (22) different definitions that saw many of them overlapping in meaning [1]. However, its definition by the United States National Institute of Standards and Technology (US-NIST) has been viewed as more refined since it captures its practical meaning [2].

In order to arrive at a more functional understanding of Cloud Computing, there is need to understand the concept of the term CLOUD. Weinmann [3] derived five logical tautology and concludes that a cloud structure is a six-tuple (space, time, directed graph, set of states, transition function, and initial state) that satisfies the five axioms namely; Common, Location-independent, Online, Utility, and on-Demand (C.L.O.U.D).

The definitions of the five axioms were captured as follows;

- *Common*: It is common if and only if there are at least two nonzero flows that are commonly sourced.
- *Location-independent:* It is location-independent if there still exists some feasible allocation at any other possible locations given that at least one feasible allocation at time t given a particular configuration of locations.
- *Online*: It is online when there still exists a path between any two vertices on a network graph without emphasizing on the direction of each edge.
- *Utility*: It is utility since its pricing is proportional to quantity.
- *On-Demand*: it is on-Demand if and only if it is completely feasible.

1.2 Cloud Computing Architecture

From the definition of cloud computing [4], cloud computing is delivered by server-based applications accessible via the Internet. There are five evident qualities synonymous with cloud computing, namely, on demand self-service, broad network access, resource pooling, rapid elastic or expansion, and measured services.

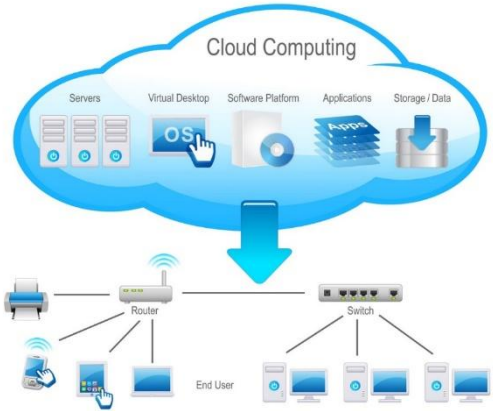

Fig.1 *Cloud computing architecture*

The architecture is designed in such a way that it has the front-end which is the visible interface that cloud users encounter through web-enabled client devices, and the back-end which comprises resources required to deliver cloud computing services. The back-end comprises of the bare metal servers, data storage compartments, virtual machines, implemented security mechanism, and cloud service(s) with conformity with a particular deployment model. It is the primary responsibility of the back-end to integrate built-in security mechanisms, traffic management control, and protocols. In the bare metal has the operating system referred to as hypervisor which makes use of well-defined protocols enabling efficient concurrent access onto the virtual machines.

The cloud computing architecture is functionalized in such a way that all applications are controlled and served by a central cloud server with replicated data remotely in the cloud configuration to create limitless efficiency and possibilities. The figure below shows a diagrammatic summary of cloud computing architecture.

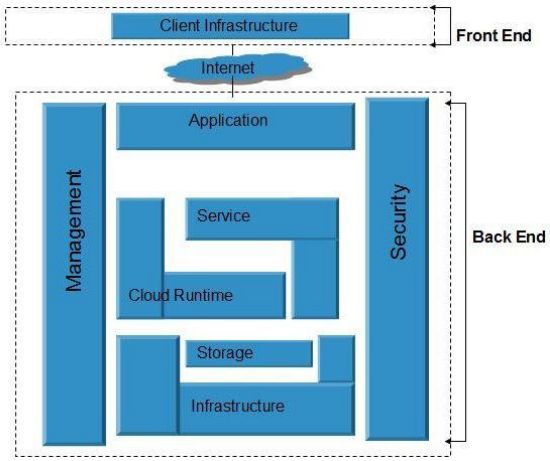

Fig.2 *Cloud Computing Structure*

Building on the five qualities of cloud computing mentioned above, the type of cloud computing can be categorized based on two models: cloud computing Service Delivery Models and Cloud Computing Deployment Models[5].

1.3 Cloud Service Delivery Models

The delivery of services within the cloud environment can be implemented through the following service delivery models;

1.3.1 Software as a Service (SaaS)

This is a service delivery model where applications are available to the cloud users over the network. The cloud providers takes the responsibility of hosting these applications used by the consumers under software distribution model. This model is characterized by application delivery from 1:M model, that is, single-instance, multitenant architecture as opposed to traditional 1:1 model, and also associated with pay-as-you-go subscription licensing model. The distinguishing bit of SaaS from other service delivery model is that it was designed to work with web browsers.

The main advantages with this model includes automated updates and patch management services, data compatibility across the enterprise and global accessibility of software of interest by the cloud users.

1.3.2 Platform as a Service (PaaS)

This is service delivery model where all facilities required to support the complete life cycle of building and delivery of web applications and services entirely available on the Internet. It is an outgrowth of SaaS model but concerns with web-based development unlike with SaaS where specific operating systems instance is favoured. The advantages with this service delivery model is that it allows users to focus on innovation and not the complex infrastructure as it provides all the infrastructure needed.

1.3.3 Infrastructure as a Service (IaaS)

This is service delivery model where cloud service providers manage the transition and hosting of selected applications on their infrastructure. In this model, the computing services are provided on a virtualized environment with cloud users maintaining ownership and management of the applications while hosting operations and infrastructure management are off-loaded to the cloud provider. The benefits that comes with this service delivery model includes availability of resources on demand from any location 24x7when required by the cloud users. The physical security of cloud users' data and any chance of system failure is smoothly handled by the cloud provider.

1.3.4 Communication as a Service (CaaS)

This is service delivery model where communication is outsourced and the provider is responsible for management of VOIP services, IM services, video conferencing et cetera consumed by their user base. This model allows a CaaS provider's business customer to selectively deploy communication feature and services throughout their company on a pay-as-you-go basis on the services used.

The advantages that comes with this model is that every component of this solution comes with 24/7 management by CaaS provider with no capital expenses needed. The customers has the benefit of multiple carrier-grade data centers with full redundancy built on to the system with minimized point of failure.

1.3.5 Monitoring as a Service (MaaS)

This is service delivery model the provision of security is outsourced especially on business platforms depends on the Internet to conduct business. This kind of protection benefits enterprise or single client assets from cyber threats since there is constant vigilance over security of infrastructure and critical organizational assets. The advantage that comes with this model is the early detection and protection against internal and external threats since the platform is under constant control and monitoring.

1.3.6 Network as a Service (NaaS)

This is a service delivery model where the provision of a virtual network service is by the owners of the network infrastructure to the third party. It is sometimes referred to as Telco As A Service (TaaS) since it involves optimization of resources allocation by considering network and computing resources over a connectivity. It is the business model that guarantees delivery of network infrastructure in a data center virtually over the Internet. The advantages of this service delivery model is the on-demand network resource usage as the network is maintained by the cloud provider and the ability to deliver network capabilities and services without the requirements of configuring individual devices.

1.4Cloud Deployment Models

Due to the principle of outsourcing computing resources through cloud computing, their deployment can be viewed under the following models.

1.4.1 Private Cloud

This is type of cloud infrastructure that is provisioned exclusive use by a single organization with multi-tenants resource access. It may be owned, managed and operated by the organization, a third party or some combination of them.

1.4.2 Public Cloud

This is type of cloud infrastructure that is provisioned for open use of resources dynamically by the general public. It may be owned, managed and operated by one or more organizations who sells the cloud services.

1.4.3 Community Cloud

This is type of cloud infrastructure that is provisioned for exclusive use by a specific community of clients from organizations that share common interests like mission, security requirements, policy or compliance considerations. It is owned, managed and operated by one or more organization in a community.

1.4.4 Hybrid Cloud

This is type of cloud infrastructure composed of two or more distinct cloud infrastructures characteristics (private, public or community) that remain unique entities, but are bounded together by standardized or proprietary technology that enables data and application portability. It is owned, managed and operated by both cloud infrastructures incorporated.

1.4.5 Virtual Private Cloud

This is a type of cloud infrastructure allocated within a public cloud environment provisioned for a certain level of isolation between the different organizations sharing the cloud resources. It is viewed as a hybrid model of cloud infrastructure in which a private cloud solution is provided within a public cloud infrastructure. Its ownership, management and operations is assumed by the public cloud infrastructure.

1.5Cloud Computing Security

In consideration of cloud architecture, its security components and services must be transparent and generic: transparent since there is need to be applied automatically without much human intervention, and generic to ensure adjustability on the part of clients, requirements, applications and required services. A functional cloud computing security architecture is composed of Security Access Points that provides front-end security services, Security infrastructure servers that manage all cloud stored data registered in the cloud, and secure client for cloud standard clients stations extended with some security components.

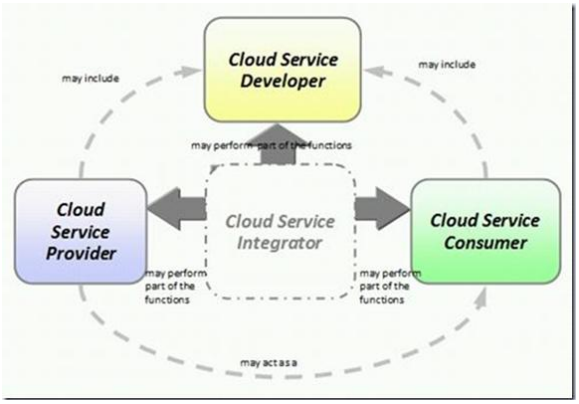

Fig.3 *Cloud Computing Security Infrastructure*

The multi-tenancy model and the pooled computing resources has introduced new security challenges such as shared resources on the same physical machines inviting unexpected side channels between machine resources and a regular resource [6]. Security being an important component of cloud computing deployment, there is need to address it to enable cloud consumers enjoy its benefits uninterrupted [7].

Research has pointed out that the most important security issue in cloud computing is data security. The problem is that the cloud service being a web application, it contains data remembrance or persistence remains an issue due to the replication and distribution of data even after user has left a cloud provider. This can be corrected by use of homomorphic encryption schemes to protect data on cloud [8].

1.6 Homomorphic Encryption Schemes and Cloud Data Security

The term *homomorphism* is borrowed from algebra meaning a structure-preserving map between two algebraic structures of the same type. The concept of homomorphism has been employed in explaining *homomorphic*, which means in literal sense same effect for different expression.

Encryption is one secure way of ensuring that cloud data is shielded from confidentiality abuse. The fact that cloud user has off-loaded his data onto the cloud environment despite uncertainty and hostility that comes with it, makes encryption a better candidate in security enhancement. A better proposal to ensure that

confidentiality of data is maintained while it is offloaded in the cloud environment is to find a way in which any operation on the said data can be effected without a third party being privy to its contents.

Homomorphic encryption enables computation to be performed on the encrypted data without requiring the secret key. This makes it a better approach to enhance security of sensitive data stored and manipulated on untrusted storage systems [9]. Therefore an efficient and fully homomorphic cryptosystem have great practical benefits to outsourcing private computations.

The concept of homomorphic encryption was introduced in 1978 by Rivest, Adleman and Dertouzous after the development of RSA algorithm [10], where they highlighted four possible encryption functions which include RSA as additive and multiplicative privacy homomorphism.

Since then, many encryption schemes have been developed but they were either using addition or multiplication homomorphic computations [11]. According to [12], an encryption scheme was developed that perform unlimited number of addition operations but single multiplication.

The realization of fully homomorphic encryption scheme appeared elusive for more than thirty years until in 2009 when Craig Gentry from IBM implemented the first version that could perform many additions and multiplications using ideal lattices and bootstrapping technique [13].

Even though Gentry's scheme was able to demonstrate the possibility of fully homomorphic encryption scheme, there was still need to improve on its complexity, efficiency and performance. For instance, he estimated that building a circuit to execute an encrypted Google search with encrypted keywords would multiply the current computing time by one trillion [11].

1.6.1 Gentry Fully Homomorphic Encryption Scheme

Gentry's scheme [13] based his construction on somewhat homomorphic encryption scheme, that is, it supported both addition and multiplication operations on ciphertexts. His idea was based on ideal lattices to provide additive and multiplicative homomorphism. His construction started from a somewhat homomorphic encryption scheme limited to evaluating low-degree polynomials over encrypted data. He then proceeded to show how to slightly modify the scheme to make it bootstrappable. Then finally, he proved that any bootstrappable somewhat encryption scheme can be converted to a fully homomorphic encryption scheme through a recursive self-embedding.

The bootstrapping technique "refreshes" the ciphertext to reduce noise factor responsible for making the ciphertext indecipherable, which was increasing with every increase in addition and multiplication of ciphertexts. With the introduction of periodical refreshing by the Boolean circuit whenever the noise increases, computation of arbitrary number of additions and multiplications was made possible, qualifying the scheme to be fully homomorphic.

He based the security of his scheme on assumed hardness of two problems:

- Certain worst-case problems over ideal lattices, and
- The sparse subset sum problem.

In general scenario, homomorphic encryption schemes are described by the following functions:

- KeyGen (λ): this is responsible for generating encryption keys where it takes the security parameter λ as an input, and generates the secret key sk and the public key pk.
- Enc (pk, m): this is responsible for encrypting the plaintext m with the public key pk and create ciphertext c.
- Dec (sk, c): this is responsible for decrypting the ciphertext c using the secret key sk to retrieve the plaintext m.

In Gentry's construction, he added another new function "Evaluate" as follows:

- Eval (pk, C, c_1, c_2, …….,c_l): this uses a Boolean circuit C to output a ciphertext c' of f(m) such that Decrypt (sk, m) = f(m).

There are three main elements evident in Gentry's scheme [11]:

- A Somewhat homomorphic encryption scheme that evaluates low degree polynomials
- A technique to "squash the decryption circuit" to obtain "bootstrappable" scheme, and
- A method of transferring "bootstrapping" the scheme into fully homomorphic scheme.

The goal of his scheme was to get a scheme that can evaluate high degree polynomials while the decryption procedure can still be expressed as low degree polynomial. Provided the scheme could evaluate its own decryption function plus an additional operation then it is called "bootstrappable" scheme and can be converted into a fully homomorphic scheme.

1.6.2 Fully Homomorphic Encryption Scheme over Integers

In June 2010, a new scheme was proposed based on the hardness of the approximate integer greatest common divisors (approximate GCD) problem [14]. It borrowed heavily from Gentry's construction though

didn't use ideal lattices. They considered a simpler mathematical approach where they replaced the Gentry's somewhat homomorphic scheme that uses ideal lattices with a simpler somewhat homomorphic scheme that uses integer over polynomial ring.

The main goal of this scheme was to reduce the complexity of Gentry's scheme. This scheme borrowed from [15] which supported only additions though can be improved to accept a limited number of multiplications, and also from [16] which was not even additively homomorphic.

In this Dijk's somewhat homomorphic encryption scheme, it has the following functions:

- KeyGen (λ): The secret key here is a random η-bit integer: $p \leftarrow (2Z+1)\Omega(2^{\eta-1}, 2^{\eta})$. Where Z is a real number and η is a bit-length of the secret key, γ is the bit-length of the integers in public key, ρ is the bit-length of the noise and τ is the number of integers in public-key. For the public key sample a random number x_i from the distribution $D_{\gamma,\rho}(p)$ for $0 \leq i \leq \tau$ subject to the condition that the largest x_i is odd. The public key $pk=<x_0,x_1,\ldots\ldots x_\tau>$.
- Enc (pk, m):This encrypts the plaintext message $m \in \{1,2,\ldots\ldots,\tau\}$ and a random integer r in $(-2^{\rho}, 2^{\rho})$ the output ciphertext c is calculated as $c \leftarrow [m+2r+2\Sigma_{i \in S} X_i]x_0$.
- Eval (pk, C, c_1, c_2,……,c_l):The *evaluate function* work as per Gentry's scheme, however all operations are performed over integers.
- Dec (sk, c):The outputs m' is calculated as $m' \leftarrow (c \bmod p) \bmod 2$.

The greatest challenge with this scheme is that it can only be fully homomorphic scheme if the noise stays sufficiently smaller than the secret key p, meaning if the noise grows bigger then the original plaintext cannot be revealed well.

1.6.3 Fully Homomorphic Encryption without Bootstrapping

This proposed scheme was based on Learning with Errors (LWE) and Ring Learning with Errors (Ring LWE) problems [17]. The bootstrapping technique was not employed here since there was need of having to "squash the decryption circuit" in order to realize a fully homomorphic encryption scheme.

The Braskerski's somewhat homomorphic algorithms consists of the following functions:

- KeyGen (l^{λ}): Sample $sk:=s \leftarrow Rq$ to be polynomial with small coefficients chosen from the error distribution χ.
- Enc (sk, μ, R_2): Sample $a \leftarrow Rq$ at random, and a polynomial e with small coefficient from error distribution χ. The output ciphertext c is calculated as $c:=(a, as+2e+\mu)$.
- Dec (sk, c=(a,b)): Sample $b \leftarrow Rq$, at random, $\mu:=b-as$ over Rq. The output $\mu:=\mu'(\text{mod}2)$.

Unlike in the earlier Fully Homomorphic Encryption schemes, the breakthrough in this scheme is the use of modulus switching in the management of noise so that it increases linearly with the multiplicative level instead of exponentially in precious cases. This allowed the scheme to increase the number of homomorphic operations without squashing or bootstrapping.

1.7 Analysis of the Fully Homomorphic Schemes

The fully homomorphic encryption schemes are compared based on three parameters, namely, security, efficiency and complexity factors.

1.7.1 Gentry's Fully Homomorphic Encryption Scheme

Complexity: This scheme is based on somewhat homomorphic encryption scheme with assumed hardness of certain worst-case problems over ideal lattices, and the sparse subset sum problem.

Performance: The ciphertext size and computation time increases with increase in security level making the scheme impractical [18].

Security:The security of this scheme is based on assumed hardness of certain worst-case problems over ideal lattices, and the sparse subset sum problem [19].

1.7.2 Fully Homomorphic Encryption over Integers Scheme

Complexity: This scheme was motivated by the need to reduce the complexity of Gentry's scheme. It significantly proved that different mathematical approaches and theories can be applied to construct a fully homomorphic encryption scheme using Gentry's blueprint.

Performance: The noise factor increases here with addition and multiplication operations performed, that is, it doubles with addition and squares with multiplication. For instance, in order to improve the security of the scheme, the ciphertext was selected to have a large value n^6, which increases with multiplication hence impact negatively on the efficiency.

Security: The security of this scheme is based on the hardness of the approximate-gcd problem. With appropriate selection of parameters the scheme has proved to resist different types of attacks to recover the secret key including brute-force attack with at least 2λ time. It has also been proved that the scheme can be attacked to recover the plaintext from ciphertext using lattice reduction algorithm [20].

1.7.3 Fully Homomorphic Encryption without Bootstrapping Scheme

Complexity: In comparison with Dijk's scheme, Brakerski's scheme uses more complex mathematical algorithms and notations as a result of Ring Learning With Errors instead of working with integers. Compared to Gentry's scheme, removal of bootstrapping technique has simplified the decryption functions and calculations.

Performance: This scheme made milestone in noise management technique under three considerations. First the technique controlled the noise level by constraining it to increasing linearly with multiplication instead of exponentially. Secondly it had allowed the L-level arithmetic circuit to be evaluated with $\tilde{O}(\lambda.L^3)$ per-gate computation or instead of $\Omega'(\lambda^4)$ which is large polynomial in the security parameter. Thirdly the removal of bootstrapping technique resulted on real cost reduction as the cost of bootstrapping in only $\Theta(\lambda)$ time was $\Omega'(\lambda^4)$, hence allowing evaluation of deeper circuits at a lower cost.

Security:The security of this scheme is based on the hardness of lattice problems with quasi-polynomial approximation factors. The security level has not improved from original Fully Homomorphic Encryption Scheme as it remains 2λ time against known lattice attacks.

II. METHODOLOGY

In order to design the Addition-Composition fully homomorphic scheme, there is need to define and prove mathematical concept around it. It is the developed mathematical concept that was used to generate an algorithm, which was later tested. We started by deriving mathematical theory to support the new scheme.

2.1 Mathematical Background

Let R = Z[x]/ $\phi(x)$ where $\phi(x)$ is irreducible over Q[x] but factor modulo t. If ϕ splits exactly into r distinct irreducible factors of degree g = n/r = $\varphi(d)/r$ i.e. $\phi(x) = \prod_{j=1}^{r} j_j(x)$.

then by Chinese Remainder Theorem, the following $Rt \cong Z_t[x]/ (f_1(x) \otimes \ldots\ldots \otimes Z_t[x]/f_r(x))$ is a natural isomorphism which is obtained.

In particular this helps to add and multiply in parallel. In our new technique we obtain the addition-composition via tensor products.

Definition 2.1

Let f(x) and g(x) be two polynomials such that $f(x):A \to B$ and $g(x):B \to C$ where A, B, C are arbitrary rings. Then the composition of f(x) and g(x) is well defined and given by:

$g(x) \circ f(x) = (g \circ f)(x) = g(f(x)) \quad \forall x \in A.$

Example

Let f(x) = 3x+1 and g(x) =x-2. Clearly, $(g \circ f)(x) = 3x-1$.

Definition 2.2

Consider two ciphers θ and τ.

We define addition-composition homomorphic encryption by:

$\eta(\psi \oplus \varphi) = \eta(\psi) \circ \eta(\varphi)$ where $\circ$ is a composite function and $\oplus$ is an additive function.

Lemma 2.3

Generally,

$$\eta(\sum_{i=1}^{m} \chi_i) := \Theta\eta(\chi_i) \qquad \forall m \in N = Z^+$$

Proof

Definition is well posed and defined. To see this consider θ and τ and $z \in Z^+$ where z=Z.

So $\theta = \lambda^x \circ \alpha_1^y \bmod n^2$

$\tau = \lambda^k \circ \alpha_2^y \bmod n^2$

$\theta \circ \tau = \lambda^x \alpha_1^y \circ \lambda^k \alpha_2 \bmod n^2$

$= \lambda^{x+k} \circ (\alpha_1 \alpha_2)^y \bmod n^2$

So $\theta_1 \circ \theta_2 \circ \ldots\ldots\ldots\ldots \circ \theta_m = \lambda^{(x_1+x_2+\ldots\ldots\ldots+x_m)} \circ ((\lambda_1\lambda_2\ldots\ldots\ldots\lambda_m)^y \bmod n^m$ and this completes proof.

The addition-multiplication homomorphic encryption is to the advantage of being usable in real time. The composite function element comes in handy since the result of the composite function f(x) is encrypted and

malicious hosts cannot know the results of the function. For instance in the mobile phone technology, the owner of the function gets the encrypted result through the function g(x). For instance Richard is the owner of function h(x) he wants to calculate the input c of Bernard but he doesn't want to expose himself as the owner of the function. So he chooses g(x) and creates f(x) then sends it to Bernard. Bernard hence calculates the result through f(x) using his input x and sends the result to Richard. Bernard cannot calculate h(x) because what he can see is just f(x). Only Richard can get the real result of h(x) through adding f(x) into inverse function i.e. $h(x) = g^{-1}(f(x))$. For this reason, we develop some theoretic results based on the addition-composition homomorphic encryption[21].

III. SCHEME TESTING AND RESULTS

To run the program run
> javac acfhes.java
> java acfhes 10 20

The above runs the program specifying values of 10 and 20 for m_1 and m_2 a brief explanation of the outputs, this can be found in the code

$ java acfches 10 20
 # Specify 10 and 20 as inputs
10 composed to 34 # (10 + 7) * 2
20 composed to 54 # (20 + 7) * 2

34 reverse composed to 10 # (34 / 2) - 7
54 reverse composed to 20 # (54 / 2) - 7

Input 10 was encrypted to
914230298129780563011715681158340346654543602699977594450413796356184635162358093323230424
681696171637738663970932013303240313090330397882701683852919774558090980556153806745895351
887029329584210683602643550613641944502069410962257148973647432913024456315152143971912392
189220967152295791165785868413596188 27

Input 20 was encrypted to
195871832380398390712064736903081228410982010530497359491304890546381955178006401516941397
235860351701415195879993524751311621904227470186880716468621690370012492329637488592526514
419657178063328558900523651203645487805449824562590341630887921851927793128324256451209785
330169637290096333335826406282444416 80

Composed 34 was encrypted to
435979835190856069575731053492627477693122994147643478666481631966106300080928605072941717
361105448847316024008073488076464116457390131962519865940279770694936674882194902603300653
368128233844131406381093790852058588239611074466185351965911407413382081930993606892729547
251819250277113091651660555878385 8404

Composed 54 was encrypted to
562234489019579959780500888969326350546292803602895504092061574348688071282624629945470971
345564002577314785691081471268095671752902222216274678842140014299124832954267865103647902
544118836854787492471316016565033986749963234007657859580660106797755236895691903430025641
551273220145721529090888669525302299 58

First 10 input decrypted to 10
Second 20 input decrypted to 20

First composed 34 decrypted to 34
Second composed 54 decrypted to 54

Above shows **acfches** successful encryption and decryption

Original sum: 30
 # Manual sum of 10 + 20

Composed sum: 88
 # Manual sum of 34 + 54

Below are now homomorphic

```
/* test homomorphic properties -> D(E(m1)*E(m2) mod n^2) = (m1 + m2) mod n */
```

Decrypted original sum: 30
*# Decrypt(E(10) * E(20) mod n^2)*
Decrypted composed sum: 88
*# Decrypt(E(34) * E(54) mod n^2)*
Reversed decrypted composed sum: 37
 # Applying ReverseComposition on Sum

```
/* test homomorphic properties -> D(E(m1)^m2 mod n^2) = (m1*m2) mod n */
```

Original product: 200
*# Manual product 10 * 20*
Composed product: 1836
*# Manual product 34 * 54*

Decrypted product: 200
 # (E(10) to power 20) mod n
Decrypted composed product: 1836
 # Sum as above
Reversed decrypted composed product: 911
 # Result from reverse composing theproduct.

IV. CONCLUSIONS

The study focused on the design and analysis of addition-composition fully homomorphic encryption technique that enhances data security on cloud computing. We have demonstrated that this is a possibility. A new scheme was developed that has foundations on mathematical proofs on validity and reliability. The study also demonstrated that on the tested metrics, addition-composition fully homomorphic scheme performs better than the previous schemes. The scheme is complex for easy adversarial attack, the speed of encryption faster and the resource consumption are manageable.

REFERENCES

[1] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres and Maik Lindner. A Break in the Clouds: Towards a Cloud Definition. ACM SIGCOMM Computer Communications Review, Vol. 39, No. 1, 2009.
[2] Peter Mell and Tim Grance. The NIST Definition of Cloud Computing. NIST Special Publication 800-145, Vol. 15, 2011.
[3] Joe Weinmann. Axiomatic Cloud Theory. Communications, Media and Entertainment Industry for Hewlett-Packard, 2011.
[4] NIST. US National Institute of Standards and Technology. Special Publications 800 – 145, 2011.
[5] Rittinghouse J. W., and Ransome J. F. Cloud Computing: Implementation, Management and Security. (CRC Press, 2010)
[6] Gnanavelu D. and Gunasekaran G. Survey on Security Issues and Solutions in Cloud Computing. International Journal of Computer Trends and Technology. Vol. 8 No. 3. Pp 126 – 130. ISSN: 2231-2803, 2014.
[7] Tiwari P. K. and Mishra B. Cloud Computing Security Issues, Challenges and Solution. International Journal of Emerging Technology and Advanced Engineering. Vol. 2 Issue 8 ISSN: 2250-2459, 2012.
[8] Kumar K. S., Anjaneyulu N. and Venkanna. Security Issues in Cloud Computing and Study on Encryption Method. International Journal of Emerging Trends and Technology in Computer Science. Vol. 2 Issues 3, pp 442 – 446, ISSN: 2278-6856, 2013.
[9] Chakraborty Nilotpal. Cloud Security using Homomorphic Encryption. National Conference on Advances in Computing, Networking and Security, 2013.
[10] Rivest R., Adleman L., and Dertouzos M. On Data Banks and Privacy Homomorphisms. Foundations of Secure Computation, Academic Press, pp. 169 – 177, 1978.

[11] Amna Ahmed Ali. A Comparative Study of Fully Homomorphic Schemes for Cloud Computing. International Journal for Emerging Technology and Advanced Engineering. Vol. 3 Special Issue 4. 2013.
[12] Boneh D., Goh E., and Nissim K. Evaluating 2-DNF Formulas on Ciphertexts. In Proceedings of Theory of Cryptography (TCC) '05, LNCS 3378, pp. 325-341, 2005.
[13] Gentry C. A Fully Homomorphic Encryption Scheme. PhD Thesis in Computer Science, Stanford University, California, USA, 2009.
[14] Marten Van Dijk, C. Gentry, S. Halevi and V. Vaikuntanathan. Fully Homomorphic Encryption Over Integers. Advance Cryptology – EUROCRYPT'10, Vol. 6110. Springer, 2010, pp. 24- 43.
[15] Leviel E., and Naccache D. Cryptographic Test Correction. 2008.
[16] Cohen Bram. Simple Public Key Encryption. 1998.
[17] Brakerski Z., Gentry C., and Vaikuntanathan. Fully Homomorphic Encryption without Bootstrapping. Proceedings of the 3rd Innovations in Theoritical Computer Science Conference (ITCS'12), pp. 309 – 325, ACM Press, New York, NY, USA, 2011.
[18] Lauter K., Naehrig M., and Vaikuntanathan V. Can Homomorphic Encryption be Practical? CCSW'11 Proceedings of the 3rd ACM Workshop Cloud Computing Security Workshop, pp 113 – 124, 2011, ISBN: 978-1-4503-1004-8, ACM New York, NY, USA. 978-1-4503-1004-8, ACM New York, NY, USA.
[19] Hu Y. and Wang F. An Attack on a Fully Homomorphic Encryption Scheme. Proceedings PKC2010, International Association for Cryptologic Research. 2010.
[20] ChungSheng Gu. Attack on Fully Homomorphic Encryption over Integers. Cryptology ePrint Archive Report 2012/157. 2012.
[21] Omollo R. O. and Okelo B. N. On Theory and Applications of Mathematics to Security in Cloud Computing: A Case of Addition-Composition Fully Homomorphic Encryption Scheme. International Journal of Academic Studies. Vol. 2 No. 4, 2016.

Omollo R.O Enhancing Data Security in Cloud Computation Using Addition-Composition Fully Homomorphic Encryption Scheme." IOSR Journal of Engineering (IOSRJEN) , vol. 7, no. 10, 2017, pp. 20-29.