Outsourcing in Cloud Computing Using Homomorphic Encryption Potentials

Kishore.K[1], E.Madhusudhana Reddy[2]

[1]Scholar of Master of technology in Computer Science & Engineering, [2]Professor & Head, Department of Information Technology, [1, 2]Madanapalle Institute of Technology & Science, Madanapalle, Andhra Pradesh, India.

ABSTRACT: FHE was an amazing Breakthrough in theoretical cryptography and as such it deserves all the attention is has been given Outsourced computations, it enable better solutions towards practical problems that need major computations. Nevertheless, users' privacy remains as a major challenge, as the service provider can access users' data freely. It has been shown that fully homomorphic encryption scheme over the integers might be the perfect solution, the main appeal of this scheme (compared to Gentry's) is its conceptual simplicity and better in performance we prove that the scheme remains semantically secure. as it allows one party to process users' data homomorphically, without the necessity of knowing the corresponding secret keys and in the end come up with desired result in encrypted form this can then be sent to one or more parties who have access to the decryption key and they will learn the results in the clear. This shows that fully homomorphic encryption can be implemented using simple arithmetic Operations.

Keywords: cryptography, fully homomorphism encryption, Recrypt procedure, van Dijk, Gentry, Halevi and Vaikuntanathan (DGHV)

1. INTRODUCTION:

A couple of years ago, Craig Gentry Produced a break-Through result in cryptography: what researchers had been dreaming about for more than 25 years was finally possible FHE. It allows access to highly scalable, inexpensive, on-demand computing resources that can execute the code and store the data that are provided to them. This aspect, known as data outsourced computation is very attractive, as it alleviates most of the burden on IT services from the consumer (or data owner). Nevertheless, the adoption of data outsourced computation by business has a major obstacle, since the data owner does not want to allow the un trusted cloud provider to have access to the data being outsourced. Merely encrypting the data prior to storing it on the cloud is not a viable solution, since encrypted data cannot be further manipulated. This means that if the data owner would like to search for particular information, then the data would need to be retrieved and decrypted a very costly operation, which limits the usability of the cloud to merely be used as a data storage centre.

My turn, Homomorphic Encryption is a process by which complex calculations can be performed on data, and it does not matter that the data is encrypted.

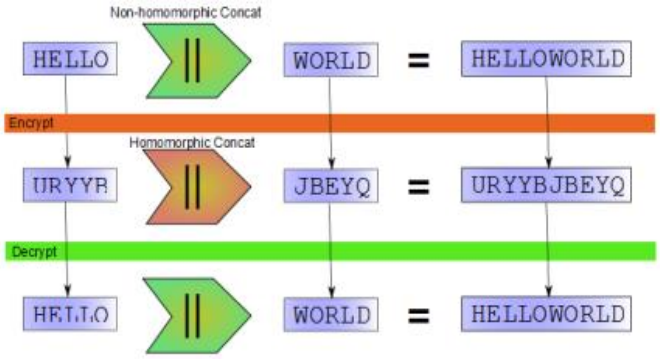

Fig 1. Scenario of Homomorphic Encryption

So a Fully Homomorphic Encryption scheme would seem to solve the security problem while still being compatible with a cloud deployment scenario. We can say that an encryption scheme is "fully homomorphic" if it supports enough homomorphic operations to implement any function we need. I'm being purposely imprecise here, because the details are a bit of a digression at this point. Although there are a number of encryption schemes with one homomorphic operation, until recently no one was really sure if fully homomorphic encryption was even possible.

Homomorphic encryption schemes are very useful in voting schemes, with the following structure: voters encrypt their votes, the homomorphic property is used to add all votes together, and the result is decrypted (with group decryption by a set of authorities who need to gather together, in a very public way, to perform a decryption). There are several homomorphic encryption schemes, some have been known for decades (e.g. El Gamal). They are efficient, and secure (as secure as asymmetric encryption can be). Note that homomorphic encryption solves the question of anonymous tallying, but that's only a small part of a proper voting scheme (e.g. the voter must also prove that he encrypted a 0 or a 1, not a 20-- otherwise, he could get 20 votes). Homomorphic encryption can also be used in digital cash systems, there again in order to ensure anonymity or some other properties.

Fully homomorphic encryption is a term which was coined when were first found encryption schemes which preserved *two* algebraic operations in a ring structure: namely, given *E(a)* and *E(b)*, you can compute *E(a+b)* and *E(ab)*. It turns out that with those two operations, you can compute just about everything. This is where the "cloud" gets into the picture: the cloud is powerful, but not trustworthy; hence, you could encrypt your data, send it to the cloud which performs the computation you want to do, and then decrypt the result.
Offloading computations to the cloud is, right now, a pure fantasy. The most efficient fully homomorphic encryption schemes currently known, based on a scheme by Gentry (published in 2009), are still very expensive, and the "arbitrary computation" part involves representing the computation as a circuit where *each* logic gate is emulated through its own homomorphic encryption. We are not talking about a 10x slowdown here; rather, we are talking about the whole Amazon EC2 cloud not being able, in a day, to perform homomorphically a computation which would take one second on a single iPhone. So while this is very interesting on a theoretical point of view, it will take a while before anything applicable in practice is discovered. Also, 2009 is quite recent; traditionally, we wait for at least 5 to 10 years before declaring that an asymmetric encryption scheme is "secure".

2. CLOUD SERVICES AND DEPLOYMENT MODELS

More and more companies are following the steps of early adopters of cloud computing and making strategies for cloud migration due to the list of enticing business benefits of the cloud that is only getting longer as the new ways for successful mitigation of risks involved with cloud adoption are emerging. As the technology itself is maturing, so are the ways to make the migration to the cloud more effective and its benefits more tangible. With a proper planning and the right choice of the cloud solution and solution provider, companies can truly capitalize on the benefits offered by the cloud in the form of reduced costs and capital expenditures, increased operational efficiencies, scalability, flexibility, immediate time to market, and so on.

Understanding of the key aspects of cloud computing and its core architectures is vital to pursuing the right cloud computing solution. Each company chooses a cloud service and a deployment model based on their specific business, operational, and technical requirements. Here we will in short introduce the three cloud service models (IaaS, PaaS, SaaS) which can serve as a foundation for further developing your understanding of cloud computing and its capabilities. You can read definitions of the primary Cloud Deployment Models (private cloud, community cloud, public cloud, and hybrid cloud).

- ***Software as a Service (SaaS).*** SaaS is a software model provided by the vendor through an online service. User doesn't have to install or maintain SaaS application. Software is running on a provider's cloud infrastructure and a user can access it via web browser. With SaaS, vendor makes the required software available to a business on subscription basis, and charges are based on the product usage. SaaS model can save companies the expense on buying hardware and software and it removes the maintenance costs.
- ***Platform as a Service (PaaS).*** PaaS enables companies to develop applications more quickly and efficiently in a cloud environment using programming languages and tools supported by the provider. The provider is responsible for maintenance and control of the underlying cloud infrastructure including network, servers, and operating systems. PaaS services provide a great deal of flexibility allowing companies to build PaaS environments on demand with no capital expenditures.
- ***Infrastructure as a Service (IaaS).*** With IaaS, a company can rent fundamental computing resources for deploying and running applications or storing data. It enables companies to deliver applications more efficiently by removing the complexities involved with managing their own infrastructure. IaaS enables fast deployment of applications, and improves the agility of IT services by instantly adding computing processing power and storage capacity when needed.
-

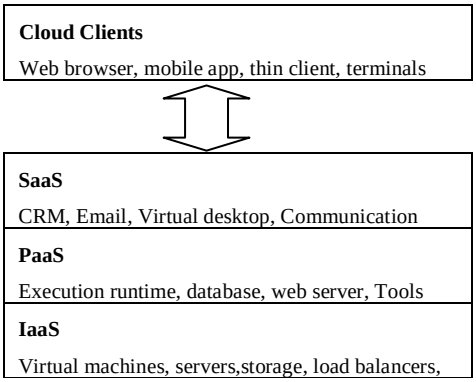

Fig 2. Architecture of Cloud Service models

Each company chooses a deployment model for a cloud computing solution based on their specific business, operational, and technical requirements. There are four primary cloud deployment models: private cloud, community cloud, public cloud, and hybrid cloud. Here is how each of the deployment models are defined:

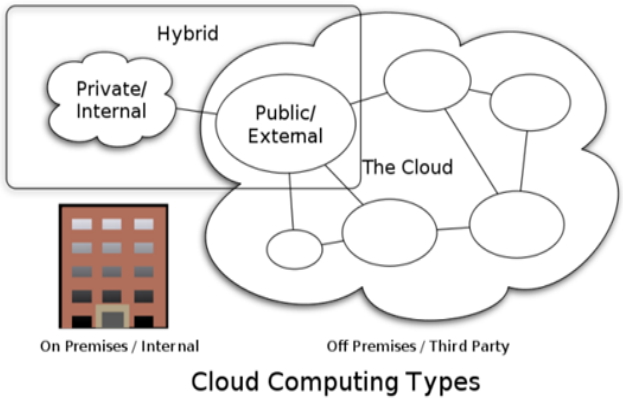

Fig 3. Cloud deployment model

Private cloud

The term *private cloud* has been described as neologisms, but the concept behind it pre-dates the term *cloud* by 40 years. Even within modern utility industries, hybrid models still exist despite the formation of reasonably well functioning markets and the ability to combine multiple providers. Some cloud vendors have used this term to describe offerings that emulate cloud computing on private networks. These offerings are able to deliver some benefits of cloud computing and at the same time mitigate some of the shortcomings. In the private cloud, the infrastructure is managed and operated exclusively for one company in order to keep a consistent level of security, privacy, and governance control. It can exist on or off premise, and can be managed by a company or a third party.

Community cloud

A *community cloud* refers to cloud computing environment shared and managed by several organizations that have similar requirements and are sharing the infrastructure in order to realize some of the benefits of cloud computing. With the costs spread over fewer users than a *public cloud* this option is more expensive but may offer a higher level of privacy, security and/or policy compliance. It may be managed by the company or a third party and can exist on or off premise.

Public cloud

Public cloud or *multi-tenant cloud*, describes cloud computing in the traditional mainstream sense. Resources are dynamically provisioned on a fine-grained, self-service basis over the Internet, via web services, from an off-site third-party provider who shares resources and bills on a fine-grained utility computing basis. The cloud infrastructure is owned by a cloud vendor, and is accessible to the general public or a large industry group.

Hybrid cloud

A *hybrid cloud* environment consists of multiple clouds (private, community, or public) and is the typical cloud deployment model for most enterprises. By integrating multiple cloud services users may be able to ease the transition to *public cloud* services while avoiding issues such as PCI compliance. In this cloud computing deployment model, an organization provides and manages some resources in-house and has others provided externally. The main benefit of the hybrid cloud is that it provides the scalability and low costs of a public cloud without exposing mission-critical applications and data to third-party.

3. FHE COMPUTATION OVER INTIGER AND ITS RESULTS

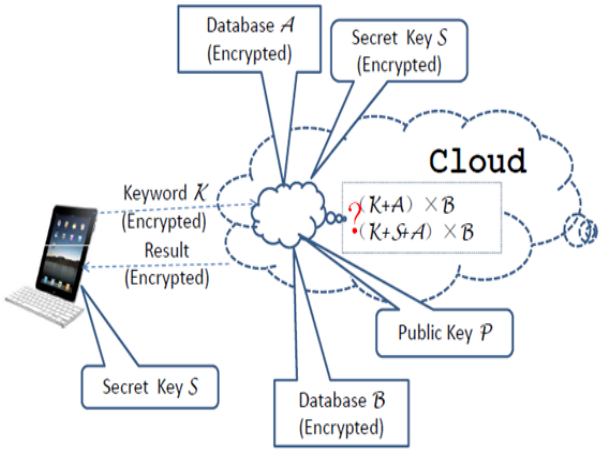

Fig 4. Fully homomorphic Encryption scenario

3.1 Fully Homomorphic Encryptions

The idea of fully homomorphic encryption was raised by Rivest, Adleman and Dertouzos [7], shortly after the invention of RSA [8]. A fully homomorphic encryption scheme consists of following four algorithms:

- **KeyGen(λ)**: Input a security parameter λ, it outputs public key **Pk**, secret key **Sk**.
- **Encrypt(m, Pk):** Input a message m and the public key **Pk**, it outputs a corresponding ciphertext c.
- **Decrypt(c, Sk):** Input a ciphertext c and the secret keys **Sk** it outputs a corresponding message.
- **Eval(Pk; c_1, c_2 …… c_n Cn):** Input a public key **Pk**, n ciphertext c_1, c_2 …… c_n and a permitted circuit C, it outputs Cn(c_1, c_2 …… c_n).

In this section we first recall the somewhat homomorphic encryption scheme published by van Dijk, Gentry, Halevi and Vaikuntanathan at Eurocrypt 2010 [4]. The scheme is based on a set of public integers: $x_i = p.q_i + r_i, 0 \le i \le r$ where the integer p is secret.

Notation: We use the same notation as in [4].for a real number ***x,*** we denote by [x],[x] and [x] the rounding of x up, down,or to the nearest integer. For a real z and an integer p we denote the reduction of z modulo p by $[z]_p$ with

$$-\frac{p}{2} < [z]_p \le \frac{p}{2}$$

We also denote $[z]_p$ $by\ z\ mod\ p$. We write

$f(\lambda) = o(g(\lambda))$ if

$$f(\lambda) = o(g(\lambda))\ log^k\ g(\lambda))\ for\ some\ k \epsilon N$$

The scheme parameters Given the security parameter λ, the following parameters are used:

- λ is the bit-length of the *xi*'s.
- η is the bit-length of secret key *p*.
- ρ is the bit-length of the noise r_i.
- τ is the number of x_i 's in the public key.
- ρ' s a secondary noise parameter used for encryption.

For a specific η −bit odd integer p, we use the following distribution over $\Upsilon -^{\iota}$ bit integers:

$$\mathcal{D}_{\gamma,\rho}(p) = \{Choose\ q \leftarrow \mathrm{Z} \cap \left[\frac{0,2^{\tau}}{p}\right), r \leftarrow \mathrm{Z} \cap (-2^{\rho}, 2^{\rho}):$$

$$Output\ x = q.p + r$$

$\boldsymbol{KeyGen}\ (\mathbf{1}^{\lambda})$: Generate a random odd integer p of size η bits. For $0 \le i \le r$ sample $x_i \leftarrow \mathcal{D}_{\gamma,\rho}(p)$. Relabel so that x_0 is the largest. Restart unless x_0 is odd and $\lfloor x_0 \rceil_p$ is even. Let $p^k = (x_0, x_1, \ldots . x_r) and\ s_k = p.$

$Encrypt\ (p^k, m\ \epsilon\{0,1\})$.Choose a random subset $S \subseteq \{1,2, \ldots, \tau\}$ and a random integer $r\ in\ (-2^{\rho'}, 2^{\rho'})$ and output the ciphertext:

$$c = [m + 2r + 2\sum_{i \epsilon S} x_i]$$

$\boldsymbol{Evaluate}\ (\boldsymbol{p^k}, \boldsymbol{C}, \boldsymbol{c_1}, \ldots, \boldsymbol{c_t})$:Given the circuit C with t inputs bits, and t ciphertexts c_i, apply the addition and multiplication gates of C to the ciphertexts, perfporming all the additions and multiplications over the integers, and return the resulting integer.

$\boldsymbol{Decrypt}(\boldsymbol{s_k}, \boldsymbol{c})$: Output $m \leftarrow (c \bmod p) \bmod 2$

. Note that since $c \bmod p = c - p.\left[\frac{c}{p}\right]$ and p is odd, one can compute instead: $m \leftarrow [c]_2 \oplus [\left[\frac{c}{p}\right]]_2$

This completes the description of the scheme. It is shown in [4] that the scheme is a somewhat Homomorphic scheme and that it is semantically secure under the approximate-GCD assumption.

Definition 3.1 (Approximate GCD). The (ρ, η, γ) approximate-GCD problem is: For a random η −bit odd integer p, given polynomially many samples from $\mathcal{D}_{\gamma,\rho}\ (p)$, output p.

We recall the constraints on the scheme parameters [4]:

- $\rho = \omega(\log \lambda)$ to avoid brute force attack on the noise.
- $\eta \geq \rho.\theta(\lambda\ log^L \lambda)$ In order to support homomorphic operations for evaluating the "squashed decryption circuit".
- $\gamma = \omega(\eta^2.\log \lambda)$ in order to the wart lattice-based attacks.
- $\tau \geq \gamma + \omega(\log \lambda)$ for the reduction to approximate GCD.
- $\rho^{'} = \rho + \omega(\log \lambda)$ for the secondary noise parameter.

To satisfy these constraints the following parameter set is suggested in [4]: $\rho = \lambda, \rho^{'} = 2\lambda, \eta = \tilde{\vartheta}(\lambda^2)$, $\gamma = \vartheta(\lambda^5)$ and $\tau = \gamma + \lambda$

The public key size is then $\tilde{\vartheta}(\lambda^{10})$ In practice the size of the xi's should be at least $\gamma = 2^{23}$ bits to prevent lattice attacks. The public key size is then at least 2^{46} bits, which is too large for any practical system.

3.2 Implementation of the Fully Homomorphic Scheme:

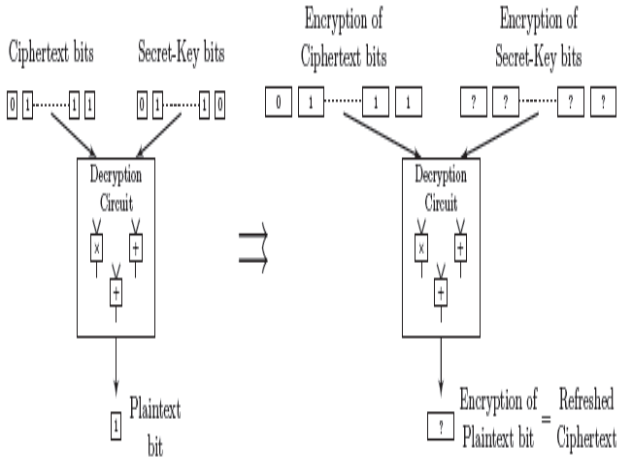

Fig 5. Gentry's key idea for achieving fully homomorphic encryption

Recryption: Now that decryption can be expressed as an arithmetic circuit of low depth, it is possible to achieve bootstrapping, i.e. to publicly evaluate the decryption circuit homomorphically on a ciphertext, which produces another ciphertext for the same message, but with possibly less noise (a "recryption"). This process, which is Gentry's key idea [6] for achieving fully homomorphic encryption, is illustrated in Figure 5. The procedure that evaluates the decryption circuit homomorphically, called **Recrypt,** takes as input encryptions of the ciphertext bits, and encryptions of the secret key bits.

In the case of the DGHV scheme or of our variant, 0 and 1 are valid encryptions of themselves, so the ciphertext bits can be passed as is to the decryption circuit. However, encryptions of the secret key bits should also be made available publicly, so the key generation from ξ 5.1 should be modified to include encryptions σ_i of the $S_i^{'}s$ into the public key $p^k = \left(p^{k^*}, y, \sigma\right)$ Then the **Recrypt** procedure is simply obtained by applying the decryption circuit to the cipher texts bits and the encrypted secret key bits.

Note that such cipher texts σ_i are normally generated using the $x_{i,b}^{'}s$ from the public key, leading to σ_i's with noise of size $\rho^{'}$. However since we are at key generation phase we can do better and let

$\sigma_i = s_i + 2r_i^{'} + p.q_i^{'} \, mod \, x_0$ for $1 \leq i \leq \Theta$,
for random integers $q_i^{'}$ modulo q_0 and random integers $r_i^{'}$ in $(-2^{\rho}, 2^{\rho})$.The resulting cipher texts σ_i have the same distribution as regular cipher texts but with noise ρ instead of $\rho^{'}$.Since $\rho < \rho^{'}$ this enables to reduce the size η of ρ required to achieve bootstrappability.

For the refreshed cipher text to decrypt correctly, its noise must be small enough, and in fact small enough that a multiplication operation between refreshed cipher texts still decrypts correctly. The cipher text bits are noise-free encryptions of themselves and the encrypted secret key bits contain ρ bits of noise, so one must have $d.\rho < \frac{\eta}{2}$, where d is the degree of the decryption circuit discussed in the previous section (or in fact, only half that degree, because only the degree with respect to the secret key bits matters; the contribution in the cipher text bits z_i can be ignored as they are used directly and without noise).

4. Fully Homomorphic Scheme completes Description:

For completeness we provide a complete description of the fully homomorphic scheme.

$\boldsymbol{KeyGen}\,(\mathbf{1}^{\boldsymbol{\lambda}})$: Generate a random odd integer ρ of size η bits. Pick a number $q_0 \in \left[0, \frac{2^r}{p}\right)$ chosen as a product of random λ^2 bit primes, and let $x_0 = q_0 . p$. Generate β pairs $x_{i,0}, x_{i,1}$ of integers with for $1 \leq i \leq \beta$;

$$x_{i,b} = p.q_{i,b} + r_{i,b}\,, \quad 1 \leq i \leq \beta\,, 0 \leq b \leq 1$$

Where $q_{i,b}$ are random integers in $[0, q_0)$ and $r_{i,b}$ are integers in $(-2^{\rho}, 2^{\rho})$.

Let $p^{k^*} = \left(x_0, x_{1,0}, x_{1,1}, \ldots \ldots x_{\beta,0}, x_{\beta,1}\right)$.

Additionally generate random bit vectors $s^{(0)}$ and $s^{(1)}$ of length $\lceil\sqrt{\Theta}\rceil$, subject to the conditions that $k \in \left[0, \sqrt{\theta}\right)$ and $b = 0,1$, there is at most one nonzero bit among the
$S_i^{(b)}, S, k\lfloor\sqrt{B}\rfloor < i \leq (k+1)\lfloor\sqrt{B}\rfloor$, with $B = \frac{\Theta}{\theta}$, and that $S = \{\,(i,j): s_i^{(0)}.s_i^{(1)} = 1\}$ Contains exactly θ elements.

Initialize a system-wide pseudo-random number generator *f* with a random seed se, and use *f(se)* to generate integers
$u_{i,j} \in [0, 2^{k+1})$ for $1 \leq i,j \leq \lceil\sqrt{\Theta}\rceil, (i,j) \neq (1,1)$.
Then, set $u_{1,1}$ such that

$$\sum_{(i,j)\in s} u_{i,j} = x_p \, mod \, 2^{k+1}$$

Where $x_p \leftarrow \lfloor 2^k / p \rfloor$.

Compute encryptions $\sigma^{(b)}$ of the vectors $s^{(b)}$, by picking, for each $i \in \left[1, \lceil\sqrt{\Theta}\rceil\right]$ and $b = 0,1$, random integers $u_{i,j} \in [0, 2^{k+1})$ for $1 \leq i,j \leq \lceil\sqrt{\Theta}\rceil, (i,j) \neq (1,1)$. and setting:

$$\sigma_i^{(b)} = s_i^{(b)} + 2r_{i,b}^{'} + p.q_{i,b}^{'} \, mod \, x_0$$

Finally, output the secret key as $s^k = \left(s^{(0)}, s^{(1)}\right)$, and the public key as

$$p^k = \left(p^k, se, u_{1,1}, \sigma^{(0)}, \sigma^{(1)}\right).$$

Encrypt$\left(\boldsymbol{p^k}, \boldsymbol{m} \in \{\mathbf{0},\mathbf{1}\}\right)$ **:** Choose a random integer matrix $b = \left(b_{ij}\right) 1 \leq i,j \leq \beta \in [0, 2^{\alpha)^{\beta\times\beta}}$ and a random integer *r* in $\left(-2^{\rho^{'}}, 2^{\rho^{'}}\right)$.

Output the cipher text:

$$c^* = m + 2r + 2 \sum_{1\leq i,j\leq\beta} b_{ij} \, . \, x_{i,0} \, . \, x_{j,1} \, mod \, x_0$$

Add$\left(\boldsymbol{p^k}, \boldsymbol{c_1^*}, \boldsymbol{c_2^*}\right)$: *Output* $c_1^* + c_2^* \, mod \, x_0$.

Mult$\left(\boldsymbol{p^k}, \boldsymbol{c_1^*}, \boldsymbol{c_2^*}\right)$: *Output* $c_1^* \, . \, c_2^* \, mod \, x_0$.

Expand $\left(\boldsymbol{p^k}, \boldsymbol{c^*}\right)$ **:** This procedure, cipher text expansion, takes a cipher text c^* and computes the associated *z*-matrix. We can think of it as separate from either encryption or decryption, as it can be executed publicly given the cipher text and public data. To do so, for every $1 \leq i,j \leq \sqrt{\Theta}$ First compute the random integer $u_{i,j}$ using the seeded pseudo-random number generator *f(se)*, then let $y_{i,j} = \frac{u_{i,j}}{2^k}$
and compute $z_{i,j}$ given by:

$$z_{i,j} = \left[c^* . y_{i,j}\right]2$$

Keeping only $\eta = \lceil \log_2(\theta + 1) \rceil$ bits of precision after the binary point. Define the matrix $\mathcal{Z} = \left(\mathcal{Z}_{i,j}\right)$.

Output the expanded cipher text $c = (c^*, \mathcal{Z})$.
Decrypt $(s^k, c^*, \mathcal{Z})$:

Output $m \leftarrow \left[c^* - \left\lfloor \sum_{i,j} s_i^{(0)} . s_j^{(1)} . \mathcal{Z}_{i,j} \right\rfloor \right] 2$

Recrypt $(p^k, c^*, \mathcal{Z})$: Apply the decryption circuit to the expanded cipher text z and the encrypted secret key bits $\sigma_i^{(b)}$. Output the result as a refreshed cipher text c^*_{new} .

5. CONCLUSION

Fully homomorphic encryption is a hot research topic, currently only two schemes known Gentry's Scheme based on Ideal lattices, DGHV scheme over the integers, Our implementation of DGHV shows that it's possible to have simple fully homomorphic encryption with better performance as Gentry's Scheme. It has been shown that fully homomorphic encryption scheme over the outsourced data in the cloud (integers) might be the perfect solution, the main appeal of this scheme (compared to Gentry's) it's conceptual simplicity and better in performance we prove that the scheme remains semantically secure.

6. REFERENCES:

[1] E. Bach, How to generate factored random numbers.
SIAM J. Comput., vol. 17, 1988, pp. 179{193.
[2]. J. Boyar, R. Peralta and D. Pochuev, On the multiplicative complexity of boolean functions over the basis Theor. Comput. Sci., vol. 235(1), 2000, pp. 43{57.
[3]. M. J. Coster, A. Joux, B. A. LaMacchia, A. M. Odlyzko, C.-P. Schnorr and J. Stern, Improved low-density subset sum algorithms. Computational Complexity, vol. 2, 1992, pp. 111{128.
[4]. M. van Dijk, C. Gentry, S. Halevi and V. Vaikuntanathan, Fully homomorphic encryption over the integers. In H. Gilbert (Ed.), EUROCRYPT 2010, LNCS, vol. 6110, Springer, 2010, pp. 24{43.
[5]. X. Pujol, D. Stehl_e et al., fplll lattice reduction library, http://perso.ens-lyon.fr/xavier.pujol/fplll/.
[6]. C. Gentry, A fully homomorphic encryption scheme. Ph.D. thesis, Stanford University, 2009. Available at http://crypto.stanford.edu/craig.
[7]. N. P. Smart and F. Vercauteren, Fully homomorphic encryption with relatively small key and ciphertext sizes. In P. Q. Nguyen and D. Pointcheval (Eds.), PKC 2010, LNCS, vol. 6056, Springer, 2010, pp. 420{443.
[8]. D. Stehl_e and R. Steinfeld, Faster fully homomorphic encryption. In M. Abe (Ed.), ASIACRYPT 2010, LNCS, vol. 6477, Springer, 2010, pp. 377{394.
[9]. D. Stehl_e and P. Zimmermann, A binary recursive GCD algorithm. In D. A. Buell, ANTS-VI, LNCS, vol. 3076, Springer, 2004, pp. 411{425.
[10]. M.N. Wegman and J.L. Carter, New hash functions and their use in authentication and set equality, Journal of Computer and System Sciences, vol. 22(3), 1981, pp. 265-279.
[11] W. Du and M. J. Atallah, "Secure multi-party computation problems and their applications: a review and open problems," in *Proc. of New Security Paradigms Workshop (NSPW)*, 2001, pp. 13–22.
[12] J. Li and M. J. Atallah, "Secure and private collaborative linear programming," in *Proc. of Collaboration*, Nov. 2006.
[13] "Mixed-Integer Nonlinear Programming", Michael R. Bussieck Armin Pruessner_February 19, 2003
[14] E. Madhusudhana Reddy & M. Padmavathamma; Need for strong public key cryptography in electronic commerce, International journal of Computer Applications in Engineering Technology and Sciences. IJ-CA-ETS, Pages 58-68, March 2009.

[15] E. Madhusudhana Reddy & M. Padmavathamma; An information security model for E-Business, The Technology World Quarterly Journal, Volume V, Issue I, Pages 203-206, December 2009.

[16] E. Madhusudhana Reddy & M. Padmavathamma; A Middleware E-Commerce Security Solution using $RMPJ_K$ –RSA Cryptosystem and $RMPJ_K$ –RSA Digital Signature , International journal of knowledge Management and e-Learning, Volume 1, No 2, Pages 71-76, December 2009.

[17] *AUTHORS DESCRIPTION*

Kishore.K, received BSc (Computer Science) from S.V.University, Tirupati & MCA from JNTU, Hyderabd.Currently he is doing M.Tech Academic Project from MITS, Madanapalli, A.P, India.Previously he was worked as Asst.Professor in Dept.of CSE at Dr.KVSRIT, Kurnool, A.P, India.His Research interests lie in the areas of Cloud Computing, Data & Network Security and Cryptography

[18] *AUTHORS DESCRIPTION*

Dr.E.Madhusudhana Reddy received M.C.A from S.V. University, Tirupati, M.Tech (IT) from Punjabi University, Patiala, M.Phil (Computer Science) from Madurai Kamaraj University, Madurai, and PhD from S.V. University, Tirupati. Currently he is working as Head, Department of Information Technology, Madanapalle Institute of Technology and Science, Madanapalle, Andhra Pradesh, India. His research interests lie in the areas of E-Commerce, Cryptography, Network Security and Data Mining. He has published 27 research papers in national/International journals and conferences. He is a Research Supervisor for Periyar University, Salem, India guiding for M.Phil. Also he is life member of Indian Society of Technical Education of India (ISTE), life member of Cryptography Research Society of India (CRSI) and Fellow in ISET, India.