International Journal of Latest Trends in Engineering and Technology
IJLTET Special Issue- ICRACSC-2016 , pp.112-118
e-ISSN: 2278-621X

REDUCING NOISE LEVELS AT COMPUTATIONS ON ENCRYPTED DATA IN CLOUD COMPUTING THROUGH HOMOMORPHIC ENCRYPTION

V.Biksham[1] and D.Vasumathi[2]

Abstract- Homomorphic Encryption (HE) is a process of computing on encrypted data without decryption of cipher text. It is a method where we can perform specific types of calculations on cipher text, which is same result as applied on plaintext. Due to rapid development of cloud services, the customers are rely on cloud service providers (CSPs) for storing and retrieving the data from cloud servers. Frequent decrypt the cipher text and giving secret key to CSPs are probably exploiting the user data. In 2009 Craig Gentry proposes a method called Homomorphic encryption which provide the do the computations on encrypted data without decrypt the cipher text and enhance the security levels through the bootstrapping and using ideal lattices. In this paper we analyze noise levels of Partially HE & Fully HE and distinguished the noise levels in each method and propose a new method for decrease the noise levels at both SHE &FHE with various applications.

Keywords – Homomorphic Encryption, bootstrapping, lattices, noise.

I. INTRODUCTION

Cloud computing is the practice of using a network of remote servers hosted on the Internet to store, manage, and process data, rather than a local server or a personal computer. Cloud computing [1] is a general term for the delivery of hosted services over the Internet. It enables companies to consume compute resources as a utility -- just like electricity - rather than having to build and maintain computing infrastructures in-house. Cloud computing promises several attractive benefits for businesses and end users. Three of the main benefits of cloud computing includes: Self-service provisioning: End users can spin up computing resources for almost any type of workload on-demand related to applications of IT and Non IT products or services and so on. Elasticity: Companies can scale up as computing needs increase and then scale down again as demands decrease. Pay per use: Computing resources are measured at a granular level, allowing users to pay only for the resources and workloads they use. Cloud computing services can be private, public or hybrid [2]. Private cloud services are delivered from a business' data center to internal users. This model offers versatility and convenience, while preserving management, control and security as shown in figure (1). Internal customers may or may not be billed for services through IT chargeback. In the public cloud model, a third-party provider delivers the cloud service over the Internet. Public cloud services are sold on-demand, typically by the minute or the hour. Customers only pay for the CPU cycles, storage or bandwidth they consume. Leading public cloud providers include Amazon Web Services (AWS), Microsoft Azure, IBM/Soft Layer and Google Compute Engine. Hybrid cloud is a combination of public cloud services and on-premises private cloud – with orchestration and automation between the two. Companies can run mission-critical workloads or sensitive applications on the private cloud while using the public cloud for burst workloads that must scale on-demand. The goal of hybrid cloud is to create a unified, automated, scalable environment which takes advantage of all that a public cloud infrastructure can provide, while still maintaining control over mission-critical data.

[1] *Department of Computer Science and Engineering CMR Engineering College, Hyderabad, Telangana State, India*
[2] *Department of Computer Science and Engineering JNT University, Hyderabad, Telangana State, India*

Although cloud computing has changed over time, it has always been divided into three broad service categories: infrastructure as a service (IaaS), platform as a service (PaaS) and software as service (SaaS)[3]. IaaS providers such as AWS supply a virtual server instance and storage, as well as application program interfaces (APIs) that let users migrate workloads to a virtual machine (VM). Users have an allocated storage capacity and start, stop, access and configure the VM and storage as desired. IaaS providers offer small, medium, large, extra-large, and memory- or compute-optimized instances, in addition to customized instances, for various workload needs.

In the PaaS model, providers host development tools on their infrastructures. Users access those tools over the Internet using APIs, Web portals or gateway software. PaaS is used for general software development and many PaaS providers will host the software after it's developed. Common PaaS providers include Salesforce.com's Force.com, Amazon Elastic Beanstalk and Google App Engine. SaaS is a distribution model that delivers software applications over the Internet; these are often called Web services. Microsoft Office 365 is a SaaS offering for productivity software and email services. Users can access SaaS applications and services from any location using a computer or mobile device that has Internet access. Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics**,** three service models, and four deployment models.

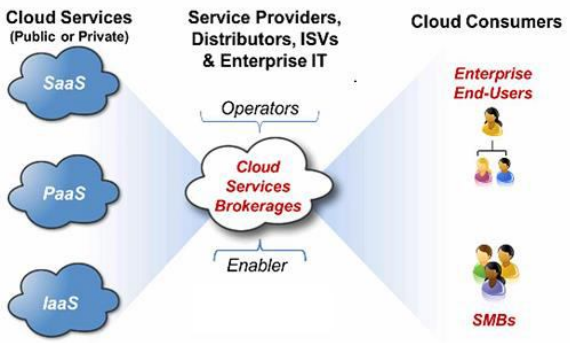

Figure 1: Cloud computing architecture

1.1. Essential Characteristics:

On-demand self-service: A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider. *Broad network access:* Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations). *Resource pooling:* The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, and network bandwidth. *Rapid elasticity:* Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.

1.2 Service Models:

Software as a Service (SaaS). The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

Platform as a Service (PaaS). The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming Languages, libraries, services, and tools supported by the provider.[3] The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

Infrastructure as a Service (IaaS). The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls).

1.3 Deployment Models:

Private cloud: The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.

Community cloud: The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.

Public cloud: The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.

Hybrid cloud: The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

1.4 Virtualization:

The cloud computing providers (CSP) come from huge digital stations called Data centers, using techniques based on Virtualization [4]. The virtualization is all the technical material software that can run on a single machine with multiple operating systems and/or multiple applications, separately from each other, as if they were working on separate physical machines. It can simplify the management of the server's side, by optimizing the use of resources and enabling high availability. The growth of the activity has irreparably the need to evolve the IT infrastructure. Adding a new servers for new applications at risk of under –use others. Administration costs are increasing and the structure loses flexibility and reliability. Among the reasons for adopting virtualization are server consolidation and infrastructure optimization. Briefly it can reduce the number of servers and the amount of hardware needed in data center. The types of virtualization techniques are Virtual Machine, Isolator, Full Hypervisor and Para virtualization.

II. RELATED WORK

In 1978, shortly after the invention of the RSA cryptosystem, Rivert, Adleman, and Dertouzos [RAD][5]came up with the idea of fully homomorphic encryption, which they called "Privacy Homomorphism". 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 homomorphism'; they form an interesting subset of arbitrary encryption schemes" .Despite the optimism of Rivert, Adleman, and Dertouzos [5], fully homomorphic encryption remained out of reach for many years.

Homomorphic encryption scheme that allow simple/complex computations (add, multiply, XOR) on encrypted data. For example, the encryption systems of Goldwasser and Micali[6],El Gamal[7], Cohen and Fischer, and Pailler[8] support either adding or multiplying encrypted ciphertexts, but not operations simultaneously. Boneh, Goh and Nissim [9] 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 [10] constructed a fully Homomorphic Encryption scheme (FHE) capable of evaluating an arbitrary number of additions and multiplications (and thus, compute and function) on encrypted data by using Ideal Lattices [11] and Bootstrapping [12]. Aside from Gentry's scheme Smart and Vercauteren [13] and an optimization by Stehle and Seinfeld, there are two other fully homomorphic encryption schemes.

III. HOMOMORPHIC ENCRYPTION

The word homomorphic has roots in Greek and loosely translates as "same shape" or "same form". In relation to cryptography, the concept is that operations can be performed on encrypted data without sharing the secret key needed to decrypt the data [14]. Homomorphic encryption has great utility in cloud computing, particularly for those that wish to house encrypted data on cloud provider's servers. Homomorphic Encryption: Homomorphic encryption is a form of encryption which allows specific or related types of computations/calculations to be performed on ciphertexts and generate an encrypted results which, when decrypted, matches the result of operations performed on the plaintext. This is most desirable and security feature of modern cloud computing domain.

The following figure(2) shown below is how Homomorophic Encryption is applied to the cloud computing, if x is an plaintext, by using public key(pk) performs encryption i.e.$Enc_{pk}(x)$ and generates a ciphertext(c).By using function(f),we can apply any arbitrary computations on C. Here y=Eval(f,Enc(x)) generates a result based on function(f) which yields the same operation performed on plain text. i.e. Enc(x) = Enc(0) where 0 is plaintext.

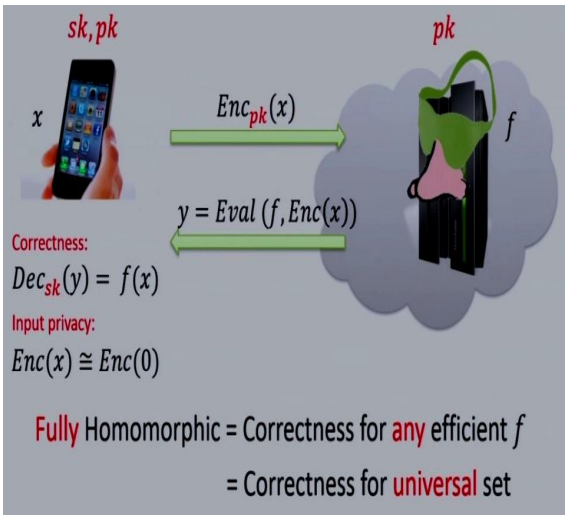

Figure 2: Homomorphic Encryption

IV. IMPLEMENTATION

Partially Homomorphic Encryption: A cryptosystem is considered partially homomorphic if it exhibits either additive or multiplicative homomorphism, but not both. Some examples of partially homomorphic cryptosystems are:RSA - multiplicative homomorphism,ElGamal -multiplicative homomorphism, Paillier -additive homomorphism

i. RSA - multiplicative homomorphism

RSA exhibits multiplicative homomorphism. By multiplying two (or more) RSA ciphertexts together, the decrypted result is equivalent to the multiplication of the two (or more) plaintext values.
Example 1. Consider an RSA key pair (d, e) and modulus n. Recall that the encryption procedure for a message m is $c=m^e \bmod n$ and the decryption procedure is $m=c^d \bmod n$ Given two plaintext messages, x_1 and x_2, the corresponding ciphertext is x_1^e and x_2^e respectively. Multiplying the ciphertext together yields $(x_1x_2)^e$. When decrypted, $((x_1x_2)^e)^d = x_1x_2 \bmod n$.

ii. ElGamal –multiplicative homomorphism

ElGamal exhibits multiplicative homomorphism. By multiplying each component of multiple ciphertexts with their corresponding respective components, the decrypted result is equivalent to the multiplication of the plaintext values. Consider an ElGamal public key (α,β,p) with private key a. Recall encryption of a plaintext message x with nonce k to be $€(x,k) = (y_1,y_2)$ where $y_1=\alpha^k \bmod p$ $y_2= x\beta^k \bmod p$. Given two plaintext messages x_1 and x_2 with nonces k_1 and k_2, the corresponding ciphertexts are: $€(x_1, k_1) = (y_1,y_2) = (\alpha^{k1} \bmod p, x\beta^{k1} \bmod p)$ $€(x_2, k_2) = (y_3,y_4) = (\alpha^{k2} \bmod p, x\beta^{k2} \bmod p)$ Multiplying the two ciphertexts together yields: $(y_1, y_2).(y_3, y_4) = (y_1.y_3, y_2. y_4) = (\alpha^{k1}\alpha^{k2}, x_1\beta^{k1}.x_2\beta^{k2}) = (\alpha^{k1+k2}, x_1x_2\ \beta^{k1+k2})$ Decrypting the resulting ciphertext yields: $d(\alpha^{k1+k2}; x_1x_2\beta^{k1+k2}) = x_1x_2$

iii. Paillier -additive homomorphism

Paillier exhibits additive homomorphism. By multiplying each component of multiple ciphertexts with their corresponding respective components, the decrypted result is equivalent to the addition of the plaintext values. The Paillier cryptosystem consists of the following values: Two large primes p and q and n = pq. We define $\lambda(n) = lcm(p - 1, q - 1)$. We choose some value g where $g \in Z^*_{n2}$ and $L(g^{\lambda} \bmod n^2)^{-1} \bmod n$ (known as μ) exists. The public key is (n,g) and the private key is (λ,μ). $L(u) = (u-1)/n$, Consider two plaintext message $x_1$and x_2 with corresponding ciphertexts: $€(x_1,r_1) = g^{x_1}r^{n}_1 \bmod n^2$,$€(x_2,r_2) = g^{x_2}r^{n}_2 \bmod n^2$.Multiplying the two ciphertexts together yields: $€(x_1,r_1) .€(x_2,r_2) = g^{x_1}r^{n}_1 . g^{x_2}r^{n}_2 \bmod n^2 = g^{x_1+x_2}(r1r2)^n \bmod n^2$.Decrypting the resulting ciphertext yields: $d(€(g^{x_1+x_2}(r1r2)^n) = x_1 + x_2 \bmod n$

V.FULLY HOMOMORPHIC ENCRYPTION

Principally, FHE allows for arbitrary computations on encrypted data. Computing on encrypted data means that if a user has a function f and want to obtain f(m1, . . . , mn) for some inputs m1, . . . , mn, it is possible to instead compute on encryptions of these inputs, c1, . . . , cn, obtaining a result which decrypts to f(m1, . . . , mn). In some cryptosystems the input messages (plaintexts) lie within some algebraic structure, often a group or a ring. In such cases the ciphertexts will often also lie within some related structure, which could be the same as that of the plaintexts. The function f in older homomorphic encryption schemes is typically restricted to be an algebraic operation associated with the structure of the plaintexts. For instance, consider ElGamal. If the plaintext space is a group G, then the ciphertext space is the product G×G, and f is restricted to the group operation on G. Indeed most schemes prior to 2009 fit such a structure. We can express the aim of fully homomorphic encryption to be to extend the function f to be any function. This aim can be achieved if the scheme is homomorphic[15] with respect to a functionally complete set of operations and it is possible to iterate operations from that set.

i. Lattices and bases

In this section we introduce some basic notions regarding lattice-based cryptography. For each notion, we will first formalize it, then provide an example to illustrate it. Individually, each section should not be hard to understand, the difficulty comes from the fact that there is *a lot* of them. However, we believe that if the reader calmly go through them, it should be accessible. In mathematics, a set of n independent vectors can be viewed as a basis of a vector space. For example, an identity matrix is a basis for the Euclidean space, where each column of the matrix is an independent vector (this is basically the space you use in any geometry class). Any point of this space is the result of a linear combination of those vectors. For example, stating that a point is at coordinates (1,3,2)(1,3,2) means that this point can be reach by adding the first basis vector once, the second thrice and the last twice.

ii. On the Learning With Error problem (LWE):

Given a basis, consider linear combinations of its vectors and add a small error. The problem of distinguishing the resulting linear combination with error from a completely random vector is called the Learning with Error problem. Stated differently, can we find the lattice vector closest to the vector with noise? Can we solve CVP for all basis and/or linear combinations? And the answer is: it depends. In order to be consistent with the rest of this paper, we will consider the case where our vector's coefficients are taken modulo some integer qq. You can see our space as a box, whenever a vector goes out of the box; it comes back from the other side (like in the game snake). Let us now take a look at a few examples (those examples are only here to get a better grip of the problem, they are in fact easy to solve with the right algorithms):

iii. On the Hardness of LWE:

As stated in the introductory section on LWE, the hardness of solving LWE is tied to the one of finding a better basis for a given lattice. The most widely used algorithm for basis reduction is LLL (we will not detail the algorithm, nor the other existing methods). This algorithm produces a reduced basis in a polynomial time, but at the cost of an approximation exponential in the number of dimensions. If the approximation is too important relatively to our space (modulo qq), solving CVP produces an error (i.e. fail).

iv. On rings and notation:

The scalar product of a matrix AA and a vector ss produces a vector consisting of the scalar product of each of the matrix row vector with ss. Moreover, the product of an integer pp with a vector ee result in a vector in which each coefficient is equal to the coefficient of ee multiplied by pp. However, when writting e+m1e+m1 where ee is an n-dimensional vector and m1m1 an integer, consider vector addition of ee and the n-dimensional vector [m1,0,...,0][m1,0,...,0]. Now that we have a hard problem at hand, we can start talking about cryptography: we will start by explaining the basis behind LWE-based SHE scheme. Then we will see how Gentry proposed to make it *fully* homomorphic. Finally we will talk a bit about Ring Learning With Error problem (RLWE).

VI. RESULTS & DISCUSSION

The following table shows the various schemes support to the Homomorphic Encryption property and there implemented year.

Scheme	Homomorphic Properties	Algorithm	Year
RSA	Multiplicative	Asymmetric	1977
Elgaml	Multiplicative	Asymmetric	1985
Bonaloh-Goh-Nissim	Multiplicative	Asymmetric	1994
Goldwasser Micali	XOR/ Additive	Asymmetric	1982
Pailler	Additive	Asymmetric	1999
Okamoto Uchiyama	Additive	Asymmetric	1998
Gentry	Fully	Asymmetric	2009

Table1: Homomorphic Encryption schemes

The noise levels of Somewhat Homomorphic Encryption (SHE) schemes using H.E.property.

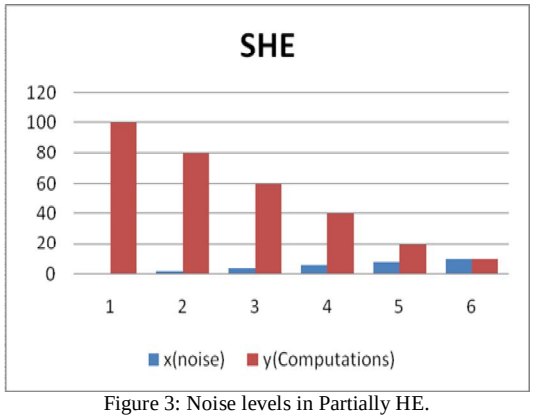

Figure 3: Noise levels in Partially HE.

The noise levels of Fully Homomorphic Encryption (FHE) schemes using Homompric Encryption property.

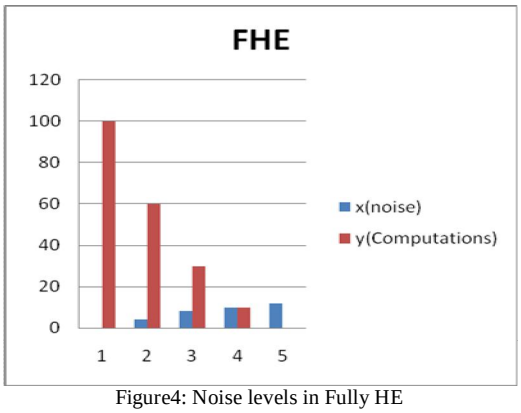

Figure4: Noise levels in Fully HE

The noise levels in fully homomorphic encryption is inversely directly proportional to the number of computations performed on ciphertext.

VII.CONCLUSION

To conclude with this post, FHE is a promising field in cryptography, with very interesting properties. However it is still quite limited regarding its computation capabilities. Moreover, transforming a complex application so that it supports encrypted data requires, if not a good understanding of homomorphic cryptography, a dialogue between developers and cryptographers.

REFERENCES

[1]. Dr.D.Durga Bhavani, "Implementations & analysis of open source computing on cloud based ERP method" IJARCCE, Volume5, issue 7, ISSN (online) 2278-1021, July 2017. http://www.ijarcce.com/upload/2016/july-16/IJARCCE%20152.pdf
[2] https://en.wikipedia.org/wiki/Cloud_computing
[3] Sumit Khurana, Anmol Gaurav Verma "Comparison of Cloud Computing Service Models: SaaS, PaaS, IaaS" IJECT Vol. 4, Issue Spl - 3, April - June 2013, ISSN : 2230-7109 (Online) | ISSN : 2230-9543 (Print)
[4] Maha TEBAA,Said EL HAJII "Secure Cloud.Computing through Homomorphic Encryption" International Journal of Advancements in Computing Technology (IACT),Volume5,Number16,December 2013
[5] R L.Revert, L.Adleman,and M.L.Dertouzos "On data banks and privacy homeomorphisms." In Foundations of secure computation, volume4, pages 169-180.New-York: Academic Press, 1978.
[6] Shruthi R 1 , Sumana P 2 , Anjan K Koundinya3 "Performance Analysis of Goldwasser-Micali Cryptosystem" International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 7, July 2013,ISSN 2278-1021(online).
[7] Czesław Koscielny, "A New Approach To The Elgamal Encryption Scheme" Int. J. Appl. Math. Comput. Sci., 2004, Vol. 14, No. 2, 265–267.
[8] http://security.hsr.ch/msevote/seminar-papers/HS09_Homomorphic_Tallying_with_Paillier.pdf
[9] Craig Gentry, Shai Halevi, Vinod Vaikuntanathan IBM T.J. Watson Research Center, NY, USA "A Simple BGN-type Cryptosystem from LWE" March 30, 2010, https://eprint.iacr.org/2010/182.pdf
[10] Craig Gentry's PhD Thesis - Stanford University https://crypto.stanford.edu/craig/
[11] Craig Gentry Stanford University and IBM Watson, "Fully Homomorphic Encryption Using Ideal Lattices" cgentry@cs.stanford.edu https://www.cs.cmu.edu/~odonnell/hits09/gentry-homomorphic-encryption.pdf
[12] vika Brakerski,Weizmann Institute of Science, Craig Gentry, Vinod Vaikuntanathan ,University of Toronto
IBM T.J. Watson Research Center"Fully Homomorphic Encryption without Bootstrapping" https://eprint.iacr.org/2011/277.p df
[13] Henning Perl, Michael Brenner and Matthew Smith Distributed Computing Security Group Gottfried Wilhelm Leibniz Universität Hannover, Germany {brenner,perl,smith}@dcsec.uni-hannover.de
"POSTER: An Implementation of the Fully Homomorphic Smart-Vercauteren Crypto-System"https://www.dcsec.uni-hannover.de/uploads/tx_tkpublikationen/ccsp121b.pdf
[14] Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjøsteen, Angela J¨aschke1, Christian A. Reuter1, and Martin Strand "A Guide to Fully Homomorphic Encryption" https://eprint.iacr.org/2015/1192.pdf
[15] http://blog.quarkslab.com/a-brief-survey-of-fully-homomorphic-encryption-computing-on-encrypted-data.html