Homomorphic EncryptionUsingRNSTheorem

Shishir Kumar Choudhary andAnkit Gupta,
8th sem, CSE, YDIT, Bengaluru.
Email: shishirkundan@gmail.comand greamsx@gmail.com

ABSTRACT

The prospect of outsourcing an increasing amount of data storage and management to cloud services raises many new privacy concerns for individuals and businesses alike. Theprivacy concerns can be satisfactorily addressed if users encrypt the data they send to the cloud. If the encryptionscheme is homomorphic, the cloud can still perform meaningful computations on the data, even though it is encrypted.In fact, we now know a number of constructions of fully homomorphic encryption schemes that allow arbitrary computation on encrypted data. In the last two years, solutions forfully homomorphic encryption schemes have been proposedand improved upon, but it is hard to ignore the elephant inthe room, namely efficiency { can homomorphic encryption ever be efficient enough to be practical? Certainly, it seemsthat all known fully homomorphic encryption schemes havea long way to go before they can be used in practice. Given this state of affairs, our contribution is two-fold.First, we exhibit a number of real-world applications, in the medical, financial, and the advertising domains, whichrequire only that the encryption scheme is \somewhat" homomorphic. Somewhat homomorphic encryption schemes,which support a limited number of homomorphic operations,can be much faster, and more compact than fully homomorphic encryption schemes.Secondly, we show a proof-of-concept implementation of the recent somewhat homomorphic encryption scheme ofBrakerskiand Vaikuntanathan whose security relies on the ring learning with errors"(Ring LWE) problem. The scheme is very efficient, and has reasonably short ciphertexts. Our unoptimized implementation in magma enjoys comparable efficiency to even optimized pairing-based schemes with the same level of security and homomorphic capacity. We also show a number of application-specific optimizations to the encryption scheme, most notably the ability to efficiently convert between different message encodings in a cipher-text.

Keywords: FHE, cloud storage homomorphic encryption scheme, SHE

.

1. INTRODUCTION

The development of cloud storage and computing platforms allows users to outsource storage and computationson their data, and allows businesses to offload the task of maintaining data-centers. However,concerns over loss ofprivacy and business value of private data is an overwhelming barrier to the adoption of cloud services by consumersand businesses alike. An excellent way to assuage these privacy concerns is to store all data in the cloud encrypted, andperform computations on encrypted data. To this end, we need an encryption scheme that allows meaningful computation on encrypted data, namely a homomorphic encryption scheme. Homomorphic encryption schemes that allow simple computations on encrypted data have been known for a long time. For example, the encryption systems of Goldwasserand Micali [GM82], El Gamal [El-84] and Paillier [Pai99] support either adding or multiplying encrypted ciphertexts,but not both operations at the same time. Boneh, Goh and Nissim [3] were the first to construct a scheme capable of performing both operations at the same time {theirscheme handles an arbitrary number of additions and justone multiplication. More recently, in a breakthrough work,Gentry [Gen09, Gen10] constructed a fully homomorphic encryption scheme (FHE) capable of evaluating an arbitrary number of additions and multiplications (and thus, computeany function) on encrypted data.The main point of this paper is to show to what extent current schemes can actually be used to compute functionsof practical interest on encrypted data. Since the appearance of Gentry's scheme, there has been much informal discussion in the industry as to whether fully homomorphicencryption is implementable and practical. While the initial solution may not have been practical, subsequent developments produced other schemes [DGHV10, SV10, SS10] leading up to the most recent solutions of Brakerski and aikuntanathan [BV11b, BV11a], an implementation of which weconsider in this paper. The scheme is efficient and simple,produces short ciphertexts, and its security is based on thering learning with errors" (Ring LWE) problem [LPR10].While the

performance of the state-of-the art FHE implementations is itself a question of interest (and has indeedbeen considered recently in, e.g., [GH11, SV11]), our focus here is on describing concrete practicalapplications andconcrete useful functions to be computed, most of which require only a limited number of multiplications of ciphertexts(as well as a possibly very large number of additions of cipher texts). For these applications, it is enough to consideran implementation of a somewhat homomorphic encryption" (SHE) scheme, namely, one which allows a fixed number of multiplications of ciphertexts. These SHE schemesare building blocks for the FHE schemes of, e.g., [Gen09,DGHV10, BV11b, BV11a], and provide much better efficiency guarantees than their fully homomorphic counter-parts.

1.1 Practical Applications of Homomorphic Encryption

We describe a number of concrete applications and functions to be implemented to provide cloud services in themedical, financial, and advertising sectors. (We provide asketch of the applications here, and refer the reader to Section 2 for detailed descriptions).For a cloud service managing electronic medical records(EMR), consider a futuristic scenario where devices continuously collect vital health information, and stream them toa server who then computes some statistics (over these measurements, and over the course of time) and presumablydecides on the course of treatment (e.g., whether the dosageof medicine should be changed). The volume of the datainvolved is large, and thus, the patient presumably does notwant to store and manage all this data locally; she may prefer to use cloud storage and computation. To protect patientprivacy, all the data is uploaded in encrypted form, and thusthe cloud must perform operations on the encrypted data inorder to return (encrypted) alerts, predictions, or summariesof the results to the patient.We describe scenarios such as the above, which requirecomputing simple statistical functions such as the mean,standard deviation, as well as logistical regressions that aretypically used for prediction oflikehoods of certain desirable or undesirable outcomes. For these functions, it suffices to have a somewhat homomorphic encryption system whichcomputes many additions and a small number of multiplications on ciphertexts: for example, averages require no multiplications, standard deviation requires one multiplication,and predictive analysis such as logistical regression requiresa few multiplications (depending on the precision required).Other applications we describe in the financial and advertising sector use similar functions, except that in those sectors,the function itself may also be private or proprietary.

2. CLOUD SERVICES

Adoption of cloud services by consumers and businessesis limited by concerns over the loss of privacy or businessvalue of their private data. In this section we will describeconcrete and valuable applications of Fully HomomorphicEncryption which can help preserve customer privacy whileoutsourcing various kinds of computation to the cloud. Inall of these scenarios, we imagine a future of streaming datafrom multiple sources, uploaded in encrypted form to thecloud, and processed by the cloud to provide valuable services to the content owner. There are two aspects of thecomputation to consider: the data itself, and the functionto be computed on this data. We consider cases where oneor both of these are private or proprietary and should notbe shared with the cloud.In all of these applications, we consider a single content owner, who is the consumer for the cloud service. All datathat is encrypted and sent to the cloud is public-key encrypted to the content-owner's public key, using the semantically secure somewhat homomorphic encryption schemefrom [5] described later in this paper.

2.1 Medical Applications: Private data andPublic functions

In [6] , a private cloud medical records storage system (Patient Controlled Encryption) was proposed, in whichall data for a patient's medical record is encrypted by thehealthcare providers before being uploaded to the patient'srecord in the cloud storage system. The patient controlssharing and access to the record by sharing secret keys withspecific providers (features include a hierarchical structureof the record, ability to search the encrypted data, and various choices for how to handle key distribution). Howeverthis system does not provide for the cloud to do any computation other than search (exact keyword match, or possiblyconjunctive searches). With our FHE implementation, weadd the ability for the cloud to do computation on the encrypted data on behalf of the patient. Imagine a futurewhere monitors or other devices may be constantly streaming data on behalf of the patient to the cloud. With FHE,the cloud can compute functions on the encrypted data andsend the patient updates, alerts, or recommendations basedon the received data.The functions to be computed in this scenario may includeaverages, standard deviations or other statistical functionssuch as logistical regression which can help predict the likelihood of certain

dangerous health episodes. Encrypted inputto the functions could include blood pressure or heart monitor or blood sugar readings, for example, along with information about the patient such as age, weight, gender, andother risk factors. The functions computed may not need tobe private in this case since they may be a matter of publichealth and thus public.

2.2 Financial Applications: Private data andPrivate functions

In the financial industry there is a potential applicationscenario in which both the data and the function to be computed on the data is private and proprietary.As an example, data about corporations, their stock priceor their performance or inventory is often relevant to making investment decisions.

Figure 1 below show how hommomorphic encryption is done in banking system. Data may even be streamed on a continuous basis reflecting the most up-to-date information necessary for making decisions for trading purposes. Functions which do computations on this data may be proprietary, based on new predictive models for stock price performance and these models may be the product of costly research done by financial analysts, so a company may want to keep these models private to preserve their advantage and their investment. With FHE, some functions can be evaluated privately as follows. The customer uploads an encrypted version of the function to the cloud, for example a program where some of the evaluations involve encrypted inputs which are specified .The streaming data is encrypted to the customer's public key and uploaded to the cloud. The cloud service evaluates the private function by applying the encrypted description of the program to the encrypted inputs it receives. After processing, the cloud returns the encrypted output to the customer.

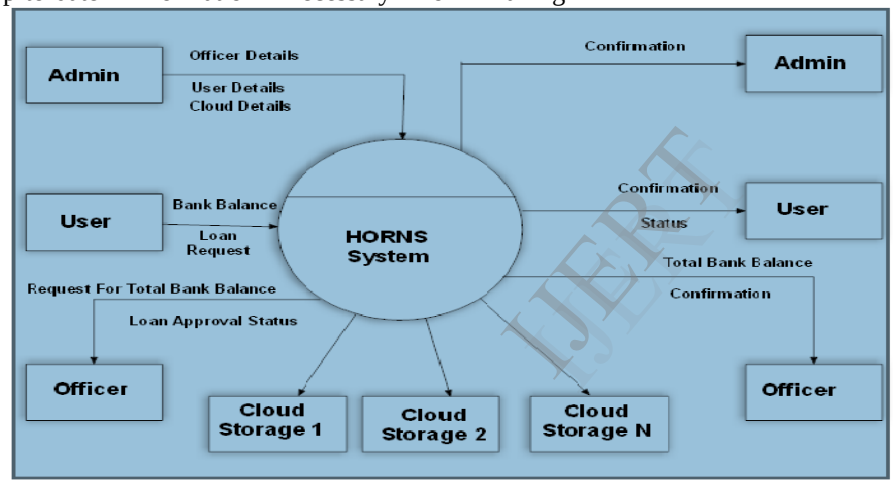

Figure 1 - High level diagram of homomorphic encryption for banking system

2.3 Advertising and Pricing

Imagine an advertiser, for example a cosmetics company,who wants to use contextual information to target advertising to potential customers. The consumer uses a mobilephone as a computing device, and the device constantly uploads contextual information about the consumer, includinglocation, the time of day, information from email or browsing activity such as keywords from email or browser searches.In the future, imagine that information is uploaded potentially constantly from video devices: either pictures of objects of interest such as brands or faces which are automatically identified, or from a video stream from a camera onthe body which is identifying context in the room (objects,people, workplace vs. home vs. store). When contextual information is uploaded to the cloud server and made accessible to the cosmetics company, the companycomputes somefunction of the contextual data and determine.This targeted advertisement to send back to the consumer's phone.Some examples of where context is important for advertising or providing targeted coupons: beer commercials duringsports events, or, you are near a Starbucks in the morning and a coffee discount coupon for the Starbucks nearby issent to your phone, or, cosmetics companies market differentproducts for different times of day (e.g. Friday night goingout vs. Sunday morning hanging out with the family), advisor coupons for shows if you are in New York near Broadwayin the evening. Other (private) contextual data might be:your income, your profession, your

purchasing history, yourtravel history, your address, etc.Encrypted version: The problem with these scenarios is theinvasion of privacy resulting from giving that much detailedinformation about the consumerto the server or to the advertising company. Now, imagine an encrypted version ofthis entire picture. All the contextual data is encryptedand then uploaded to the server; the advertiser uploads encrypted ads to the server; the server computes a functionon the encrypted inputs which determines which encryptedad to send to the consumer; this function could be eitherprivate/proprietary or not. All contextual data and all adsare encrypted to the consumer's public key. Then the cloudcan operate and compute on this data, and the consumercan decrypt the received ad. As long as the cloud serviceprovider does not collude with the advertisers, and semantically secure FHE encryption is employed, the cloud andthe advertisers don't learn anything about the consumer'sdata.

2.4 Functions to be computed with FHE

We can compute the following functions with a somewhathomomorphic encryption scheme:_ Average of n terms fcig: as a pair (Pi=1;:::n ci; n),where m =Pi=1;:::n cin is the average. Standard deviation:q(Pi=1;:::n ci□m)2n , returned as apair which is the numerator and denominator of theexpression, before taking the square root._ Logistical regression: x =Pi=1;:::n _ixi , where _i isthe weighting constant or regression coefficient for thevariable xi, and the prediction is f(x) = ex1+exA couple of remarks are in order. First, we set the parameter choices for the encryption system based on the expectednumber of multiplication operations to be done to computethe given functions. These parameter choices determine theefficiency and security of the system. Thus parameters forthe system need to be changed as the functions to be computed change.Secondly, so far we do not have a way to efficiently dodivisions of real numbers or square roots. Thus in the abovecomputations, numerators and denominators need to be returned as separate encryptions.3If the cloud and the advertiser collude, then the cloud maybe able to learn some information about whether the userlikes the ad or not, which reveals information about his preferences. This constitutes a form of CCA attack, which mightendanger the security of the FHE.

3. THE ENCRYPTION SCHEME

We describe the ring learning with errors (Ring LWE)assumption of [LPR10] in Section 3.1, and present the\some What" homomorphic encryption scheme of Brakerski andVaikuntanathan [5] based on Ring LWE in Section 3.2.We then report on an instantiation of the parameters, as wellas the running-times and sizes of the keys and ciphertextin Section 5 .

3.1 The Ring LWE Assumption

In this section, we describe a variant of the ring learning with errors assumption of Lyubaskevsky, Peikert and Regev In the RLWE assumption, we consider ringsfor some degree n integer polynomial and a prime integer the ring of degree‘n’ polynomials modulo f(x) with coefficients in Zq. Addition in these rings is done component-wise in their coefficients (thus, their additive group is isomorphic to Zn andZnqrespectively). Multiplication is simply polynomial multiplication modulo (and also q, in the case of the ringRq).Thus an element can be viewed as a degree‘n’polynomial over. One can represent such anelement using the vector of its coefficients. For an element we let note its `1 norm.The RLWEf ring assumption is parameterized by an integer polynomial of degree n (which defines thei), a prime integer q and an errordistribution over R, and is defined as follows. Let s $ Rqbe a uniformly random ring element. The assumption isthat given any polynomial number of samples of the form, where ai is uniformly random is drawn from the error distribution the bi's arecomputationally indistinguishable from uniform in Rq. Asshown in this is equivalent to a variantwhere the secret is sampled from the noise distribution rather than being uniform in Rq. It is also easy to see thatthe assumption is equivalent to a variant where the noise eiare multiples of some integer t that is relatively prime to We consider the RLWE problem for specific choices of thepolynomial f(x) and the error distribution _. Namely,we set f(x) to be the cyclotomic polynomial xn + 1for n a power of two. In addition to many other usefulproperties, the fact that means thatmultiplication of ring elements does not increase theirnorm by too much (see Lemma 3.2 below).The error distribution is the discrete Gaussian distribution for sample from thisdistribution defines a polynomial .We present some elementary facts about the Gaussian error distribution, and multiplication over the ring.The result fact bounds the(Euclidean and therefore, the `1)length of a vector drawfrom a discrete Gaussian of standard deviation byn. The second fact says that multiplication in the ring Z[x]= hxn + 1i increases the norm ofthe constituent elements only by a modest amount.

3.2 Somewhat Homomorphic Encryption

The somewhat homomorphic encryption scheme SHE =(SH:Keygen; SH:Enc; SH:Add; SH:Mult; SH:Dec) is associatedwith a number of parameters:

the dimension n, which is a power of 2, the cyclotomic polynomial f(x) = xn + 1, the modulus q, which is a prime such that q _ 1(mod 2n),Together, n; q and f(x) de_ne the rings R , Z[x]= hf(x)I and Rq , R=qR = Zq[x]= hf(x)i. the error parameter, which defines a discrete Gaussian error distribution with standard deviation a prime t < q, which defines the message space ofthe scheme as Rt = Zt[x]= hf(x)i, the ring of integerpolynomials modulo f(x) and t, and_ a number D >0, which defines a bound on the maximum number of multiplications that can be performedcorrectly using the scheme.These parameters will be chosen (depending on the securityparameter) in such a way as to guarantee correctness andsecurity of the scheme.

3.2.1 The Scheme

Keygen sample a ring element sand definethe secret key, Sample a uniformly random ringelement a1 Rq and an error e and compute thepublic key.Publish pk and keep secret. Recall that our message space is Rt. Namely,we encode our message as a degree npolynomial with coefficients in Zt.Given the public key pk = (a0; a1) and a message m 2 Rq,the encryption algorithm samples andcomputes the ciphertext. We now show how to computethe addition and multiplication operations homomorphically.To compute an arbitrary function f homomorphically, weconstruct an arithmetic circuit for f (made of addition andmultiplication operations over Zt), and then use Mult to iteratively compute f on encrypted inputs.Although the cipher texts reduced by SH:Enc containstwo ring elements, the homomorphic operations (in particular, multiplication) increases the number of ring elementsin the ciphertext. In general, the SH:Add and multoperations get as input two ciphertexts and The output of SH:Add containsmax) ring elements, whereas the output of SH:Multi contains ring elements. Assume that otherwise pad the shorter ciphertext with zeroes.Homomorphic addition is done by simple component-wiseaddition of the ciphertexts. Namely, compute and outputcteither of the ciphertexts with zero.Let v be a symbolic variable and consider the expression.

.3.2.2 Correctness and Security

We show the correctness of decryption and homomorphicevaluation in the following lemmas. The statement of thelemma also serves as the setting of the modulus q (in termsof _; t; n and D) that ensures that the scheme can performD multiplications and A additions.Lemma 3.3. The encryption scheme SHE is correct, andcan compute D multiplications followed by A additions, Proof First, note that the ciphertext canbe written in the polynomial where eachco-efficient has at most with over-whelming probability. This is because e; u; f; s and g areall polynomials whose co-efficient are drawn from a discrete. Gaussian with standard deviation and multiplyingtwo such polynomials (mod xn + 1) produces a polynomialwhose coefficients are of size at mostwith overwhelming probability (by the Central Limit theorem). Our experiments show that this number is in factsmaller, and is of the order of 2.Before we prove correctness of the homomorphic operations, we state an invariant that holds for all ciphertextsproduced either by the encryption algorithm, or as a resultof a homomorphic evaluation. The invariant is that for aciphertext where s is the secret keysmall errorand m is the message.Clearly, the invariant holds for a fresh ciphertext produced by SH:Enc (by the calculation above), assuming that. Furthermore, if the invariant holds, thenthe decryption algorithm succeeds. This is because the decryption algorithm outputs fct(s) (mod t) which is indeedthe message, assuming the bound on the error. The boundon the error essentially ensures that the quantity) that the decryption algorithm recovers does notwrap around mod q".Correctness of homomorphic addition is easy to see. Wehave two ciphertexts and that satisfy the invariants, the sum of the two ciphertextsis ctwhere the addition is done componentwise.This satisfies the invariant as well, assuming that the largererror t(e + e0) is smaller than q. In general, adding A ciphertexts with error at most each results in error at most In practice, as our experiments show, this is likely tobe smaller, namely of the order of 2pA.

3.2.3 An Optimization to Reduce Ciphertext Size

The homomorphic multiplication operation described aboveincreases the number of ring elements in a ciphertext. Brakerski and Vaikuntanathan [BV11a] describe a transformation called re-linearizationthat reduces theciphertextback to two ring elements. We describe this optimization below, implement it and report on the performance numbers.Essentially, the idea is the following: assume that we runSH:Mult on two ciphertexts (each containing two ring elements) produced by the encryption algorithm. The resulting ciphertext ctmlt contains three ring elements that satisfythe "invariant"fctmlt (s) = c2s2 + c1s + c0 = temult + mm0This is a quadratic equation in s, and thus, SH:Mult turnedtwo \linear ciphertexts" into a \quadratic ciphertexts". Thegoal of re-linearization is to bring this back down to a linearciphertext.To this end, we publish some homomorphism keys" toaid re-linearization. This could be thought of as part ofthe public key, but the homomorphism key is only used forre-linearization (following an SH:Multoperation).

The homomorphism key hk is computed. In a sense, quasi-encryptions these are They are not real encryptions since ti s2 may not lie in themessage space of the encryption scheme, namely Rt.The homomorphic multiplication generates a ciphertextctmlt = (c0; c1; c2), starting from two 2-element ciphertexts.Re-linearization is performed after every homomorphic multiplication, and proceeds as follows.On the one hand, re-linearization reduces the length of theciphertext considerably. On the ip side, the public parameters become much larger. They now consist of an additionallogt q ring elements, totaling to logt q _ n lg t = n(lg q)2= lg tbits. Re-linearization also affects the running time of the homomorphic multiplication. In particular, one needs to additionally perform roughly logt q polynomial multiplicationsand additions. These side effects are most pronouncedfor small t, where our experiments indicate considerableoverhead. Quite encouragingly, though, the benefits of re-linearization seem to dominate the side-effects for large t(see Section 5 for more details).The security of the encryption scheme given the homomorphism keys relies on the circular security of the encryptionscheme when encrypting quadratic functions of the secretkey.

4. MESSAGE ENCODING TECHNIQUES

The ease of performing homomorphic operations dependscrucially on the specific message-encoding used in the cipher texts. Consider the following two examples._ If we wish to compare two encrypted integers x; y 2 Zt homomorphically then it seems best to encrypt them4The summation runs from i = 0 to i = dlogt qe 1. Weomit the indices for brevity Bitwiserather than as an elements of Zt. The former approach translates to computing a polynomial ofdegree lg t over the encrypted bits, whereas the latterseems to require a polynomial of degree O(t)._ If we wish to compute the mean of k integers, thenit seems most natural to encode them as elements ofZt (for a large enough t). Computing the mean homomorphically then involves only cheap homomorphicadditions over Zt On the other hand, if the numbersare encrypted bit-wise, then addition requires computation of expensive \carry" operations that involve homomorphic multiplication over Z2.We describe two tricks for encoding messages. The first trickshows how to efficiently encode integers in a ciphertext so as to enable efficient computation of their sums and productsover the integers. This is useful in computing the mean, thestandard deviation and other private statistics efficiently.The second trick shows how to \pack" n encryptions of bitsinto a single encryption of the n-bit string. Some homomorphic operations, e.g., comparison of integers or privateinformation retrieval, seem to require bit-wise encryptionsof the input. Once the answers are computed, though, theycan be packed into a single encryption using this trick.

4.1 Efficient Encoding of Integers for ArithmeticOperations

Given a list of integers if our goal is tocompute their sum or product over the integers homomorphically, the obvious (and sub-optimal) choice is to encryptthem directly. Namely, for every m in the list, To ensure that we obtainPi mi over the integers (and notmod t), we are forced to choose t to be rather large, namelyt >Pi mi, which could be rather prohibitive.We show a method of encrypting integers more efficientlyby encoding them in the polynomial ring, in essence enablinga smaller choice of t and better efficiency. In particular, forsmall enough mi < 2n, we show that it suffices to choose in order to add ` integers. Being able to work with asmall t in turn enables us to choose other parameters, e.g.,q and n to be correspondingly smaller.The idea is very simple: break each m into (at mostn) bits (m(0); : : : ;m(n 1)), create a degree-(n-1) polynomialpm(x) =Pj m(j)i xj and encrypt m asAdding these encryptions now adds up the polynomials pmi(x)coefficient - wise. Note that each co-efficient was a single bitto start with, and a sum ` of them grows to at most `.As long as q > t this does not wrap around modulo t and upon decryption, we in fact get the polynomial. Now, the result is simplypmadd(2).Extending this idea to support multiplication is a bit trickier. The problem stems from the fact that multiplying thepolynomials pm(x) and pm0(x) increases their degree. Iftheir original degree was close to n to start with, we will onlybe able to obtain pm(x)pm0(x) (mod xn + 1) upon decryption, which loses information about the product. The solution is to encode the messages m as polynomials of degreeat most n=d, if we anticipate performing d multiplications.For our applications (e.g., computing standard deviations),this is an acceptable tradesince we only anticipate doinga single multiplication (or, at most a small number of themin the case of computing higher-order regression functions).

5. EXTENSIONS AND FUTURE WORK

Implementing, Fully Homomorphic Encryption. The somewhat homomorphic encryption scheme of [5] can beturned into a fully homomorphic encryption scheme usingthe re-linearization and the dimension reduction techniquesof [BV11a]. We leave the

problem of implementing the resulting fully homomorphic encryption scheme as animportant future work. Implementing bootstrapping could alsolead to a number of nice applications of homomorphic encryption, for example, to the problem of optimizing communication with the cloud described below.Optimizing communication with the cloud, we present asolution to help mitigate the problem of the large ciphertextsize for the Ring-LWE based FHE solution. In any of theabove applications, a client communicates with the cloudservice and uploads its data encrypted under a FHE scheme,and the cloud operates on this data and returns encryptedoutputs to the client. Each ciphertext has size n log(q), andfor functions requiring a large number of multiplications, qand n could be very large (see the implementation sectionfor sample choices of q and n).In short, all the communication over the network consistsof short, non-homomorphic ciphertexts. At the server's end,the cipher texts are first upgraded" to homomorphic cipher texts which are then computed on, and finally \downgraded" to short nonhomomorphic ciphertexts which are then sentto the client.

REFERENCES

[1] BennyApplebaum, David Cash,Chris Peikert, andAmit Sahai. "*Fast cryptographic primitives andcircular-secure encryption based on hard learningproblems. In Shai Halevi, editor, CRYPTO, volume5677 of Lecture Notes in Computer Science, pages595{618. Springer 2009".*

[2] Web Bosma, John Cannon and CatherinePlayoust. "*The Magma algebra system I: The userlanguage. J. Symbolic Compute, 24(3-4):235{265,1997. Computational algebra and number theory(London, 1993)"*

[3] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim."*Evaluating 2-DNFformulas, on ciphertexts. InTheory of Cryptography - TCC'05, volume 3378 ofLecture Notes in Computer Science, pages 325{341.Springer, 200"5*

[4] Zvika Brakerski,and Vinod Vaikuntanathan. *"Efficient fully homomorphic encryption from(standard) LWE, In Submission 2011".*

[5] Zvika BrakerskiandVinod Vaikuntanathan"*Fullyhomomorphic encryption from ring-LWE andsecurity for key dependent messages. ToAppear inCRYPTO 2011".*

[6] Melissa Chase, Kristin Lauter, Josh Benaloh, andEric Horvitz."*Patient-controlled encryption'*: