Learning with Errors

Foundations and Applications

Paul Galloway

UCSB, CS293G, Spring 2017

Abstract

In this paper we look at the basics of the "learning with errors" problem as applied to the engineering of cryptosystems. We are particularly interested in these types of cryptosystems because of their full homomorphic properties. Homomorphic properties are quite valuable and have applications in the arena of cloud computing. Machine learning computation on encrypted data can produce security/portability benefits for end users. We will start by exploring the high-level motivation for using the "learning with errors" problem. It is a promising candidate for the new "hard problem" in cryptography in general. The post-quantum benefits of such a scheme are of particular interest. A unifying theme in all this will be a reliance on lattice-based cryptographic concepts. We will give such background where necessary. Near the end, we mention some outstanding issues in this area. Finally, we will conclude with specific proposals for future work.

1 Introduction

The basic definition of a homomorphic encryption scheme is a scheme in which one can perform a function on multiple ciphertext, emit a result, and the decrypted version of this result will be the same as if the function were performed on the same set of plaintexts. This paper was prompted by one prominent example of homomorphic encryption applied to a machine learning task. The example is that of Numerai, which allows otherwise "siloed" hedge fund data to be shared publicly without revealing it's propreitary information. We are interest specifically in fully homomorphic encryption (FHE). A homomorphic encryption scheme is fully homomorphic if it can handle all functions, represented by addition and multiplication in finite fields. Some existing encryption schemes are partially homomorphic: RSA and ElGamal (multiplication) and Pailler (addition). Fully homomorphic schemes, on the other hand, can be represented by simple AND, OR, NOT, and XOR gates in base 2. One primary issue with FHE is that since it is built using these functions which can be easily enumerated, it has issues when there does not exist a priori knowledge of the lengths of the inputs. This is specifically an issue with regards to database systems utilizing homomorphic encryption.[14]

2 Motivation

One primary reason for wanting homomorphic encryption in a cloud setting is to maintain privacy while still operating in a ”public” cloud in which the cloud provider (presumably) has a theoretical ’backdoor’ into your application and data. A basic example would be the one shown in Figure 1. You might have a web app running on some cloud provider sending streaming event data every time it arrives from the outside world or internally. Either way, this data is now effectively visible to the cloud provider once it is in their system. Having another layer of encryption could prove useful in terms of data portability. In our case, the ”Event Data” (as shown) is assumed to have come into the web app already homomorphically encrypted. Thus, no opportunity is given for the infrastructure provider to see the method (and keys) used. But such encryption schemes are not a panacea and other information can still be snooped from the application. For example, the amount of rows queried in a database.[14] As an aside, it is worth noting that the example of Numerai is slightly simpler than the cloud example in that the encrypted data never has to be decrypted. It is used directly in a machine learning model and never decrypted.[3] Beyond ML models and/or cloud security, the utilization of homomorphic encryption ties in nicely with the future need for post-quantum cryptographic systems. The Learning with Errors (LWE) problem, specifically the Ring Learning with Errors variant which we will mostly focus on, may vary well be the new standard ’hard’ problem to base encryption schemes on. It is quantum-safe and is already gaining traction. It can be of great benefit in areas where privacy is key, medical/genomic applications being one example.[7]

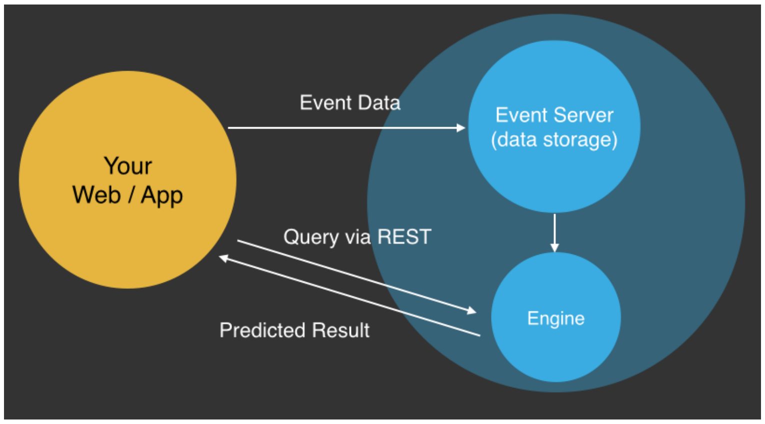

Figure 1: Cloud setup with an ML component. Credit: Apache Foundation

3 Learning with Errors

The Learning with Errors problem is a problem in the area of machine learning that is believed to be hard to solve. Regev introduced the application of this problem to cryptography in 2005.[9] In order to 'solve' the problem, an attacker must be able to 1.) Get some samples (x,y) where x belongs to the domain and y belongs to the range of the function and 2.) Find some secret noise function. If the attacker has both of these, they may be able to obtain $y = f(x)$, in general, with high probability.

3.1 Lattice-based Cryptography

An excellent overview of the application of Lattice-based cryptography can be found at [6]. Some highlights relevant to the LWE Problem are presented in the following sections.

Fundamentally, Ring Learning with Errors is a promising scheme because it is reducible to the NP-Hard Shortest Vector Problem (SVP) in a lattice.[13]

3.2 Shortest Vector Problem

In SVP, a basis of a vector space V and a norm N are given for a lattice L. Given this, one must find the shortest non-zero vector in V, as measured by N, in L. This is illustrated in euclidean space in the image below (although other vector spaces can be used).

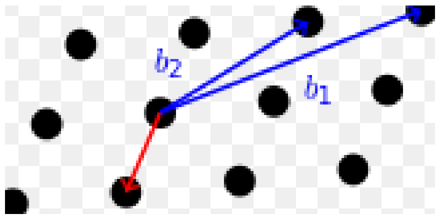

Figure 2: Shortest Vector in a lattice (basis in blue). Credit: Wikipedia

3.3 Ring Learning with Errors

The variant we will be looking at is the Ring variant of the LWE problem.[11] This variant is simply LWE over a ring. Other more elaborate techniques have come out recently, but this variant is still good to look at for introductory purposes. The benefits of this variant over the plain vanilla LWE problem are as follows: representing n vectors requires only $O(n)$ elements of Z_q, as opposed to $O(n^2)$ and using the fast Fourier transform, operations on such vectors can be significantly sped up. These results lead to cryptographic constructions that have smaller keys and are considerably faster.[10]

For Ring-LWE, the basic setup is as follows:

Let ring $R_q = Z_q[X]/(X^n + 1)$. Along with this, you have $*P$ (an error distribution on R_q), and a secret s that is drawn from the set R_q. The attacker is given pairs $(a, as + e)$ with

- a, uniformly random from R_q
- e, sampled from P

The task for the attacker is to find s.

$*P$ is often a discrete Gaussian (with small standard deviation).

3.4 Decision vs. Search

In the LWE problem, there are two separate versions of the problem. One of them is the *Decision* problem and the other is the *Search* problem. The *Search* problem consists of what was described above (namely, given a set of samples, try to derive the secret s given those samples and some noise). The *Decision* problem involves taking candidate samples and determining if they are "LWE samples" or uniformly random.

4 Implementations

One of the earliest examples of applying R-LWE is in the Key Exchange mechanism described in [8]. A more recent implementation is that being persued by Ducas, et. al in [4]. Google began experimenting with this approach recently: [2]. Below we talk about the setup of any practical implementation of an LWE scheme and briefly give two examples of usage: public-key and signatures.

- n, a prime number or power of 2. (n coefficients, max degree n-1)
- q, a prime number. (coefficients of the polynomials will be integers mod q and the arithmetic will be in R_q)
- $Phi(x)$, a cyclotomic polynomial. When n is a power of two $Phi(x) = (x^n + 1)$
- $a(x)$, a known fixed polynomial of degree less than n and with coefficients in R_q
- $s(x)$, a secret polynomial of degree less than n, with coefficients in R_q chosen according to a probability distribution
- $e(x)$, a error polynomial of degree less than n, with coefficients being in R_q and "small" in the integers and chosen according to an error probability distribution P.
- $b(x)$, a public polynomial equal to $b(x) = a(x) * s(x) + e(x)$.

$b(x)$ is the public key, $s(x)$ and $e(x)$ together constitute the private key. The parameters n, q, $Phi(x)$ and $a(x)$ along with the probability distributions for the coefficients of $s(x)$ and $e(x)$ are the system wide parameters.

4.1 Public-Key System

A scheme for the construction of a public key system based on the learning with errors problem was originally provided by Regev in [9]. A visualization of the Key generation and Encryption steps is shown below. The Decrypt function can naturally be derived from what is shown.

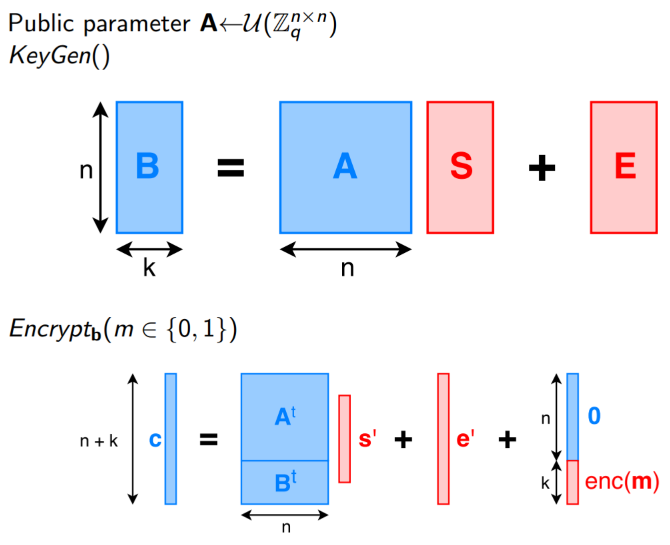
Public parameter $\mathbf{A} \leftarrow \mathcal{U}(\mathbb{Z}_q^{n\times n})$
KeyGen()
n
B
=
A
S
+
E
k
n
$Encrypt_{\mathbf{b}}(m \in \{0,1\})$
n + k
c
=
A^t
B^t
n
s'
+
e'
+
n
0
k
enc(m)

Figure 3: Generation Encryption for original LWE scheme. Credit: Leo Ducas

4.2 Signature

A scheme for signature generation is described in [5] and [12]. It goes as follows:

- Generate small polynomials $y_0(x)$, $y_1(x)$ (coefficients from $[-b$ to $b]$)
- Compute $w(x) = a(x) * y_0(x) + y_1(x)$
- Map $w(x)$ into a bit string W
- Compute c(x) $= PolyHash(W + m)$
  - c = polynomial with k non-zero coefficients
  - $+$ = concatenation of strings
- Compute $z_0(x) = s_0(x) * c(x) + y_0(x)$
- Compute $z_1(x) = s_1(x) * c(x) + y_1(x)$
- Until the infinity norms of $z_0(x)$ and $z_1(x) <= Beta$ go to step 1.
- The signature is the triple of polynomials $c(x)$, $z_0(x)$ and $z_1(x)$

At this point, you send the message $c(x)$. Additionally, you transmit $z_0(x)$ and $z_1(x)$ to the verifier. The process for verifying a message then consists of the following:

- Verify that the infinity norms of $z_0(x)$ and $z_1(x) <= Beta$, if not reject the signature.
- Compute $w'(x) = a(x) * z_0(x) + z_1(x) - t(x) * c(x)$
- Map w'(x) into a bit string W'
- Compute $c'(x) = PolyHash(W'|m)$

If c'(x) != c(x) reject the signature, otherwise accept the signature as valid.

5 Issues

We have presented the basic motivations for using LWE in cryptography, as well as given some examples of existing implementations. But naturally we should touch on some of the weaknesses that can exist in such a scheme, both from a mathematical point-of-view and from applications utilizing the homomorphic properties that LWE schemes possess.

Of course, any generic/naive attack on an RLWE scheme is exponential. But results have been given for specific weak instances. Lautner, et. al. have done work in formalizing the necessary properties number fields must have, in order to be susceptible.[7] Additionally, they have shown conditions necessary for a particular application of RLWE to be considered ”weak”. The implementations shown previously are naturally free of these weaknesses. For example, instances utilizing 2-power cyclotomic fields do not satisfy all the properties necessary for a sub-exponential attack. Such theoretical work naturally serves to boosts confidence, but many open questions remain.

Aside from theoretical concerns, there is also the ever-present issue of side-channel attacks. As mentioned early on in [14], homomorphic schemes can provide data privacy on the data itself, but certain information can still be inferred. Database accesses to this stored data can still yield information about the data. The application operating on the data must still be ”hardened” to avoid such side channel attacks. For example, by always querying some ”window” of data on each node. Aside from side channel attacks, a machine learning expert would obviously be concerned with how their models may be affected by operating on encrypted data. Previous work has been done in the area of evaluating such models, for example in [1]. Such concerns naturally apply to any homomorphic encryption scheme (not just LWE-based schemes). However, it is worth staying aware of such drawbacks when it comes to specific applications of LWE, such as making it a key component in the design of encrypted databases used for training machine learning models.

An underlying concern to all users of such LWE schemes would, of course, be the ”hardness” of the scheme being used. Though the RLWE problem on ideal lattices is believed to be hard, and commercial applications are beginning to spring up, the demonstration of hardness relies on some non-standard assumptions. The only way to reduce ”standard” problems to LWE is via a quantum reduction. Additionally, there are still some gaps in the understanding of lattice problems when applied to cryptosystems. [10]

6 Conclusion

Throughout our discussion, the overall focus has been on the application of the Learning with Errors problem to two separate but related problems. The application and further refinement of such approaches can and will present a huge set of possibilities. Not only do these approaches potentially allow for the construction of practical post-quantum cryptosystems, but they also prove to be a natural fit for applications with specific privacy requirements (that may or may not utilize a machine learning component). Depending on the adaptability of given infrastructures, such LWE cryptographic approaches may allow for increased application portability for secure workloads. It seems worthwhile to explore this area further. The post-quantum security guarantees combined with the homomorphic properties of such cryptosystems make their adoption a win-win.

An interesting exercise might be to take many of the existing implementations of LWE-based cryptographic systems and evaluate them with respect to various machine learning and cloud tasks (taking into account performance in terms of speed and security guarantees, as well as accuracy of ML models in terms of error metrics). The topic of error metrics may be worthy of more exploration in it's own right. Can such LWE schemes be strengthened further when certain assumptions are made about how they will be used. Additionally, how will the application of these schemes affect the work of data scientists. It would seem that such schemes will not necessarily be modular in design and might have side effects. There are many different issues that can be explored further which are separate from the underlying mathematics of any given implementation. A good starting point would be to perform such exercises starting with RLWE as the underlying implementation. If such exercises proved fruitful, they could be of great value to other developers who want to harden their applications. Relieving the burden on said developers would go a long way towards wider adoption. Thus, making many connected applications more secure.

6.1 Future Work

As previously stated, the study of these LWE schemes and how well they work in Cloud and/or Machine Learning environments (for various use cases) may deserve more exploration. In becoming more acquainted with the techniques used for LWE, and homomorphic encryption more generally, the author of this work plans on exploring this area more thoroughly. Any comments or suggestions are welcomed. This class paper is naturally not comprehensive and there is likely much more to explore in the LWE space, as well as in the homomorphic encryption space (specifically with respect to use cases for such encryption). Please direct all such correspondence to pgalloway@cs.ucsb.edu.

References

[1] P. Holmes C. Aslett L., Esperança. *A review of homomorphic encryption and software tools for encrypted statistical machine learning.* Oxford University, 2015.

[2] M. Braithwaite. *Experimenting with Post-Quantum Cryptography.* Google, 2016.

[3] R. Craib. *Encrypted Data For Efficient Markets.* Medium.com, User: Numerai, 2016.

[4] Ducas et. al. *Post-quantum Key Exchange - A New Hope.* IACR, 2015.

[5] Lyubashevsky V. Pöppelmann T. GüneysuVadim, T. *On the Design of Hardware Building Blocks for Modern Lattice-Based Encryption Schemes.* Springer, 2012.

[6] Ducas L. *NewHope, Frodo, in Between and Beyond Instantiating and Implementing Lattice based Cryptography.* CWI, Amsterdam, The Netherlands, 2017.

[7] K. Lauter. *Attacks on Ring Learning with Errors.* UCI, 2015.

[8] C. Peikert. *Public-Key Cryptosystems from the Worst-Case Shortest Vector Problem.* SRI International, 2009.

[9] O. Regev. *On Lattices, Learning with Errors, Random Linear Codes, and Cryptography.* Tel-Aviv University, 2005.

[10] O. Regev. *The Learning with Errors Problem.* New York University, 2010.

[11] P. Schwabe. *Post-Quantum Cryptography.* Radboud University, Nijmegen, The Netherlands, 2016.

[12] V. Singh. *A Practical Key Exchange for the Internet using Lattice Cryptography.* IACR, 2015.

[13] N.P. Smart. *Ring-LWE: An Efficient PQC Public Key Encryption Schemey.* University of Bristol, 2016.

[14] Agrawal D. Wang, S. and A. El Abbadi. *Is Homomorphic Encryption the Holy Grail for Database Queries on Encrypted Data?* UCSB, 2012.