Homomorphic Encryption through Learning With Errors (LWE)

May 12th, 2024

1 Introduction

Homomorphic encryption is an important cryptographic primitive that is necessary to support modern cryptographic protocols, such as anonymous voting. Some constructions of such a scheme are complicated and unintuitive. However, recent constructions relying on the Learning With Errors (LWE) problem and utilizing lattice structures provide concise and palatable examples.

This report will discuss the background and context for such a scheme, and walk through a sample implementation in Python.

1.1 Notation

Let $\mathcal{K}_\mathcal{S}, \mathcal{K}_\mathcal{P}, \mathcal{M}, \mathcal{C}$ represent the secret key space, public key space, message space, and ciphertext spaces of an encryption scheme, respectively.

We will refer to a *lattice* $L_{n,q}$ to be the set of n-dimensional vectors whose (integral) elements are in $\mathbb{Z}_q$ (sometimes written $\mathbb{Z}_q^n$).

The inner product of two vectors $\vec{a}, \vec{b}$ is written $\langle \vec{a}, \vec{b} \rangle$, corresponding to:[1]

$$\langle \vec{a}, \vec{b} \rangle = \sum_{i=0}^{n-1} a_i b_i \tag{1}$$

Where each vector is of size n.

When describing encryption schemes, they will each contain an encryption scheme Enc which takes as parameters a public key k_p and a message m, and decryption scheme Dec which takes as parameters a secret key k_s and a ciphertext c. An example illustrating the correctness of such a scheme would be:

$$\text{Dec}(k_s, \text{Enc}(k_p, m)) = m \tag{2}$$

2 Homomorphic Encryption

When we say that an encryption scheme is *homomorphic*, what we really mean is that we are able to perform some binary operation $S_C : (C, C) \to C$ on encrypted ciphertexts that corresponds to some operation $S_M : (M, M) \to M$ on the original messages.

An intuitive example could be a scheme where adding two ciphertexts $c_1 + c_2$ results in a ciphertext that is a valid encryption of the *sum* of the original messages m_1, m_2, such that only the sum $m_1 + m_2$

[1] I will be sticking to 0-indexed notation for this report, so that it matches more closely with the code.

is revealed when the quantity $c_1 + c_2$ is decrypted. Importantly, this cannot reveal *anything about the original ciphertexts themselves* (other than what is discernable from the operation itself).

A more precise description of a homomorphic encryption scheme is as follows:

Definition (Homomorphic Encryption). An encryption scheme $E = (\text{Enc}_E, \text{Dec}_E)$ consisting of encryption and decryption functions is *homomorphic* for an operator $S = (S_M, S_C)$ if for any two messages $m_1, m_2 \in \mathcal{M}$, public key $k_p \in \mathcal{K}_\mathcal{P}$, and secret key $k_s \in \mathcal{K}_\mathcal{S}$, the following is satisfied:

$$\text{Dec}_E\left(k_s, S_C(\text{Enc}_E(k_p, m_1), \text{Enc}_E(k_p, m_2))\right) = S_M(m_1, m_2) \tag{3}$$

An important note is that the operations S_C and S_M are not necessarily (and are generally not) the same. For instance, it is possible that $S_C(c_1, c_2) = c_1 \cdot c_2$ (multiplying the ciphertexts) whereas the effect on the messages is addition: $S_M(m_1, m_2) = m_1 + m_2$ (adding the underlying messages). In other schemes, including the LWE scheme that we will discuss later, these functions get more complicated.

This definition is somewhat specific to work with the system we will discuss later, as certain homomorphic schemes can require additional parameters to be passed to the homomorphic operation S_C, such as the public key of the scheme.

A more ideal version of this ciphertext malleability is to allow for computation of arbitrary functions (circuits) on any number of ciphertexts. This is generally referred to as *fully homomorphic encryption (FHE)* (CITATION). This seems like an incredibly difficult (or even impossible) task at first, but the following theorem helps us achieve this:

We will see that the scheme we focus on later in this report is in fact a FHE scheme, but with some limitations that we will also investigate.

2.1 Example: Diffie-Hellman

One illustrative example of homomorphic encryption is actually the Diffie-Hellman encryption scheme. If there are two encrypted messages $c_1 = (g^{r_1}, h^{r_1} m_1)$, and $c_2 = (g^{r_2}, h^{r_2} m_2)$, then we see that we get:

$$\begin{aligned} S_C(c_1, c_2) &= c_1 \cdot c_2 && (4)\\ &= (g^{r_1} \cdot g^{r_2}, h^{r_1} m_1 \cdot h^{r_2} m_2) && (5)\\ &= (g^{r_1 + r_2}, h^{r_1 + r_2} m_1 \cdot m_2) && (6) \end{aligned}$$

Note that we're basically just doing pointwise multiplication within the tuple ciphertext. Importantly, since r_1, r_2 are randomly chosen from $\mathbb{Z}_q$, the quantity $r_1 + r_2$ is also completely random. This means that this is the same as an encryption of the quantity $m_1 \cdot m_2$, the product of the original messages.

Unfortunately, this example does not suffice for *fully* homomorphic encryption, because it only supports a single operation on the message space (multiplication), so we cannot use this for arbitrary computation.

2.2 Applications of Homomorphic Encryption

In practice, homomorphic encryption is an incredibly versatile tool, which has a variety of use cases:

1. **Anonymous Voting**: Say some government or organization wanted to tally votes for an election
2. **Healthcare Statistics**:

3 Learning With Errors (LWE)

Learning with Errors generally refers to a cryptographic *assumption* about a problem that is difficult to compute in a lattice $L_{q,n}$. That problem can be described as follows:

Definition (Learning with Errors - Decisional). Let χ be a probability distribution. Consider the following distributions:

1. $\mathcal{R}$: Generate a random vector $\vec{a} \leftarrow L_{q,n}$, and a random element $b \leftarrow \mathbb{Z}_q$. Then, return the tuple $(\vec{a}, b)$.
2. $\mathcal{L}_{\vec{s}}$: Let s be a parameter of the distribution, where $\vec{s}$ is a random vector sampled from $L_{q,n}$ (this is the secret key of the distribution).

   Samples are generated as follows: select $\vec{a} \leftarrow L_{q,n}$ randomly, and then set $b = \langle \vec{a}, \vec{s} \rangle + e$, where e is sampled randomly from χ. Then, the tuple $(\vec{a}, b)$ is returned.

The (decisional) LWE problem is the following: Given m samples from either $\mathcal{R}$ or $\mathcal{L}_s$ (all samples from the same distribution), the problem is to guess which distribution the samples come from.

The LWE *assumption* is that the LWE problem is computationally infeasibly (outside of the quantum setting). That is, no polynomial time algorithm can differentiate from samples coming from the completely random distribution $\mathcal{R}$ and samples coming from the secret vector generated distribution $\mathcal{L}_{\vec{s}}$.

Note that clearly, the LWE assumption does *not* hold for all choices of χ. For instance, if $\chi = 0$ (returns 0 with probability 1), then there is no longer any randomness associated associated with b, and so an adversary can distinguish the distributions in this case.

Note that there is a similar version of this problem for *search*:

Definition (Learning with Errors - Search).

4 Homomorphic Encryption from LWE

5 Implementation

6 Conclusion