Homomorphic Encryption via Ring Learning With Errors (RLWE)

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 $\mathcal{K_S}, \mathcal{K_P}, \mathcal{M}, \mathcal{C}$ represent the secret key space, public key space, message space, and ciphertext spaces of an encryption scheme, respectively.

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 $m_1 + m_2$ 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 $k_p \in \mathcal{K_P}$, and secret key $k_s \in \mathcal{K_S}$, the following is satisfied:

$$\text{Dec}_E\left(k_s, S_C(\text{Enc}_E(k_p, m_1), \text{Enc}_E(k_p, m_2))\right) = S_M(m_1, m_2) \tag{1}$$

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 is because so long as we can multiply and add ciphertexts, we can compute any arithmetic function (by approximating it with a circuit of addition and multiplication operations). Note that we get

The variant that we will discuss later is a *somewhat homomorphic encryption (SHE)* scheme, which allows for a fixed number of operations to be computed on a given ciphertext. These schemes are less expressive than FHE schemes, but are much easier to implement.

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:

$$\begin{aligned} S_C(c_1, c_2) &= c_1 \cdot c_2 && (2)\\ &= (g^{r_1} \cdot g^{r_2}, h^{r_1} m_1 \cdot h^{r_2} m_2) && (3)\\ &= (g^{r_1 + r_2}, h^{r_1 + r_2} m_1 \cdot m_2) && (4) \end{aligned}$$

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 Ring Learning With Errors (RLWE)

The Ring Learning With Errors (RLWE) problem is a variant of the Learning With Errors (LWE) problem that is used to construct homomorphic encryption schemes. The constructs required for

these schemes are generally a security parameter n (which vaguely describes the hardness of the protocol, such that breaking the protocol is can only be achieved with 2^n operations), a modulus q (which is the modulus of the ring that we are working in).

For this scheme, we will rely on the rings of the form $R_q = \mathbb{Z}_q[x]/(x^m + 1)$, which represent polynomials with integer coefficients modulo q (aka the coefficients lie in $\mathbb{Z}_q$) modulo the polynomial $x^m + 1$ (roughly meaning that they have degree less than m). Although rings are outside the exact scope of this class (as well as the notion of modulo by ideals), these are details not directly relevant to the construction of the encryption scheme.

We also require a noise distribution χ which produces random element $e \in \mathbb{R}_q$ that is *small*, in the sense that has high probability of being less than some bound B, which we will discuss later.

The RLWE problem is then defined as follows :

4 Implementation

5 Conclusion