

Homomorphic Encryption for Approximate Arithmetic

Homomorphic encryption is a cryptosystem that supports arithmetic operations between ciphertexts. This technology has great potentials in real-world applications since it allows us to securely outsource expensive computation on public (untrusted) server. The security of practical homomorphic encryption schemes is based on the hardness of ring learning with errors problem. The inherent limitation of these constructions is that they only support the arithmetic operations over modular spaces so the required size of parameter for real number operations (ie, no modular reduction over plaintext space) is too large to be practically implemented. On the other hand, real-world applications such as statistical testing, neural networks, and machine learning do not require absolute precision and they are all to a certain degree approximate.

Recently, Cheon et al presented a new homomorphic encryption scheme which supports an approximate arithmetic of encrypted messages [18]. The main idea is to consider an encryption noise (on the ciphertext for security) as a part of computation error occurring during approximate arithmetic. In other words, given a secret key sk , an encryption ct of a plaintext m satisfies the equation $\langle ct, sk \rangle = m + e \pmod{q}$ for some small error e .

We suppose that the underlying HE scheme is based on the RLWE assumption over the cyclotomic ring $R = \mathbb{Z}[X]/(X^N + 1)$ for being a power of two. Let us denote by $[\cdot]_q$ the reduction mod q into the interval $(-q/2, q/2] \cap \mathbb{Z}$ of the integer. We write $R_q = R/qR$ for the residue ring of R modulo an integer q . The following is a description of the homomorphic encryption scheme [18].

- $\text{ParamsGen}(\lambda)$: Given the security parameter λ , choose a power-of-two integer N , a modulus $Q = q^2$ and a discrete Gaussian distribution χ . The RLWE problem of parameter (N, Q, χ) should achieve at least λ bits of security level of cryptosystem. Output $params \leftarrow (N, q, \chi)$.
- $\text{KeyGen}(params)$: Generate a polynomial $s(X) \in R$ by sampling its coefficient vector randomly from a sparse distribution on $\{0, 1, -1\}^N$ and set the secret key as $sk \leftarrow (1, s(X))$. Sample a polynomial $a(X)$ uniformly at random from R_q and an error polynomial $e(X)$ from χ . Set the public key as $pk \leftarrow (b(X), a(X)) \in R_q \times R_q$ where $b(X) \leftarrow -a(X) \cdot s(X) + e(X) \pmod{q}$. Let $s'(X) \leftarrow s(X)^2$. Sample a polynomial $a'(X)$ uniformly at random from R_Q and $e'(X)$ from χ . Set the evaluation key as $evk \leftarrow (b'(X), a'(X)) \in R_Q \times R_Q$ where $b'(X) \leftarrow -a'(X) \cdot s(X) + e'(X) + q \cdot s'(X) \pmod{Q}$.

- $\text{Enc}_{pk}(m)$: Sample a small polynomial $v(X)$ (with 0,1,-1 coefficients) and two error polynomials $e_0(X), e_1(X)$ from χ . Output the ciphertext $ct \leftarrow v(X) \cdot pk + (m+e_0(X), e_1(X)) \in R_q \times R_q$.
- $\text{Dec}_{sk}(ct)$: For $ct = (c_0(X), c_1(X))$, output $m \leftarrow c_0(X) + c_1(X) \cdot s(X) \pmod{q}$.
- $\text{Add}(ct, ct')$: Add two ciphertexts and output $ct_{\text{add}} \leftarrow ct + ct' \pmod{q}$.
- $\text{Mult}(ct, ct')$: For two ciphertexts $ct = (c_0(X), c_1(X))$ and $ct' = (c_0'(X), c_1'(X))$, let $(d_0(X), d_1(X), d_2(X)) = (c_0(X) \cdot c_0'(X), c_0(X) \cdot c_1'(X) + c_1(X) \cdot c_0'(X), c_1(X) \cdot c_1'(X)) \pmod{q}$. Output the ciphertext $ct_{\text{mult}} \leftarrow (d_0(X), d_1(X)) + [q^{-1} \cdot d_2(X) \cdot \text{evk}] \pmod{q}$.

The native plaintext space is simply represented as the set of polynomials in R with coefficient less than q , but it can be understood as a vector of $(N/2)$ -dimensional complex vector using the encoding/decoding method described in [18] (each factor is called a plaintext slot). This technique enables us to parallelize both space and computation time. Addition and multiplication in R correspond the component-wise addition and multiplication on plaintext slots. In addition, we could get a shift of the plaintext slots, that is, if ct encrypts a plaintext vector (m_1, m_2, \dots, m_k) , we could obtain the encryption of the shifted vector (m_2, \dots, m_k, m_1) .

This homomorphic encryption scheme has its own distinct and unique characteristics represented by the following rescaling procedure.

- $\text{RS}(ct; r)$. For a ciphertext $ct \in R_q \times R_q$ and $r > 0$, output the ciphertext $ct' \leftarrow [r^{-1} \cdot ct] \in R_{q'} \times R_{q'}$ for $q' = r^{-1} \cdot q$.