3 주차

Ploblem : (x + 1)^2 * (X^2 + 2) and Rotated left 2.

CKKS – 숫자(복소수)의 벡터 (Z0, Z1, … , Z(N/2 - 1)를 한번에 암호화.

원하는 식을 어떤식으로 만드느냐에 따라 처리시간이 다르다. 만약 전개하여 풀어내면 x^4… 되어 곱셈연산을 많이 수행해야하므로 비효율적. 따라서 주어진 식을 그대로 계산하는 것이 계산에서 효율적이다.

Input : 1~8192
Polymodulus degree : 2^14 = 16384
Coff modulus : 60, 50, 50, 50, 50, 60

Q 는 각 비트에 근사한 소수로 결정된다. 에러를 줄이기 위해 마지막 60bit 는 곱셈시 잠시 모듈러스를 올렸다가 내린다.

CKKS 실수의 표현 ex) 3.14 * 1000 여기서 Delta = 1000 = scale 실수를 정수화
여기서는 3.14 * 16384 가 된다. => encode

- 실수를 정수화한 후 갈루아 필드로 매핑한다. 이후 암호화 진행

PQ = P * q0 * q1 * q2 * q3 * q4
- 초기 x 레벨 : 4

C = C1 * C2 라 하면

C1 = x^2 + 1 (x -> x^2 -> x^2 + 2)
- 한번의 곱셈 : LV 4 -> 3

C2 = (x + 1)^2 (x -> x + 1 -> (x + 1)^2)
- 한번의 곱셈 : LV 4 -> 3

- Q4 = q0q1q2q3q4 -> q0q1q2q3
- 제곱을 할 때 마다 재선형화와 재스케일링 진행

- 곱셈시에는 스케일 * 스케일이 되어버리기 때문에 스케일이 매우 커지기 때문에 재스케일링 작업을 거치며 레벨을 하나 소모했다고 한다.

C = C1 * C2

Lv : 2

```
------------------------------------------------------
make (x + 1)
scale of (x + 1) : 50bits
make (x + 1)^2 and relinearization
scale of (x + 1)^2 befor rescale 100bits
scale of (x + 1)^2 after rescale 50bits
------------------------------------------------------
make (x^2) and relinearization
scale of (x^2) befor rescale 100bits
scale of (x^2) after rescale 50bits
make (x^2 + 2)
scale of (x^2 + 2) scale 50bits
------------------------------------------------------
```

위 C1 과 C2 는 각각 한번의 곱셈이 진행되므로 레벨이 같으며 재선형화 및 재스케일링 해주었기 때문에 같은 스케일을 가지고 있다. 그러므로 따로 정규화해줄 필요 없이 곱셈이 가능하다.
곱셈이후에는 Q = q0q1q2 가 되어 레벨 2 가된다.

이후 갈루아 키를 통해 left 2 로테이션을 진행한다.

Q = q0q1q2

LV 2

이후 1~10 까지의 결과만 확인해본다.

```
------------------------------------------------------
True answer (x + 1)^2 * (x^2 + 2)
12 54 176 450 972 1862 3264 5346 8300 12342
------------------------------------------------------
Decode result
11.9062 53.8438 175.844 450.219 972.062 1862.12 3263.47 5346.34 8299.97 12341.6
------------------------------------------------------
Decode result Apply Rounding
12 54 176 450 972 1862 3263 5346 8300 12342
------------------------------------------------------
Rotate 2 steps left.
176 450 972 1863 3264 5346 8300 12342 17712 24674
------------------------------------------------------
```

```cpp
#include <iostream>
#include <seal/seal.h>
#include <cmath>
using namespace seal;
using namespace std;

int main() {
    cout << "----- (x + 1)^2 * (x^2 + 2) and left 2 ---- CKKS" << endl << endl;

    EncryptionParameters parms(scheme_type::ckks);
    size_t poly_modulus_degree = 16384;
    parms.set_poly_modulus_degree(poly_modulus_degree);
    parms.set_coeff_modulus(CoeffModulus::Create(poly_modulus_degree, { 60, 50,
50, 50, 50, 60 }));
    double scale = pow(2.0, 50);

    SEALContext context(parms);
    KeyGenerator keygen(context);
    auto secret_key = keygen.secret_key();
    PublicKey public_key;
    keygen.create_public_key(public_key);
    RelinKeys relin_keys;
    keygen.create_relin_keys(relin_keys);
    GaloisKeys gal_keys;
    keygen.create_galois_keys(gal_keys);
    Encryptor encryptor(context, public_key);
    Evaluator evaluator(context);
    Decryptor decryptor(context, secret_key);

    CKKSEncoder encoder(context);
    size_t slot_count = encoder.slot_count();
    cout << "Number of slots: " << slot_count << endl;
    cout << endl;
    vector<double> input;
    input.reserve(slot_count);

    for (size_t i = 0; i < slot_count; i++)
        input.push_back(i + 1);

    Plaintext plain_coeff, plain_coeff2, x_plain;
    encoder.encode(1.0, scale, plain_coeff);
    encoder.encode(2.0, scale, plain_coeff2);
    encoder.encode(input, scale, x_plain);


    Ciphertext x_plus_1_encrypted;
```

```
    encryptor.encrypt(x_plain, x_plus_1_encrypted); // 평문 x를 암호화하여 저장
    evaluator.add_plain_inplace(x_plus_1_encrypted, plain_coeff); // x + 1
    cout << "-----------------------------------------------------" << endl;
    cout << "make (x + 1)" << endl;
    cout << "scale of (x + 1) : " << log2(x_plus_1_encrypted.scale()) << "bits" <<
endl;
    cout << "make (x + 1)^2 and relinearization" << endl;

    Ciphertext x_plus_and_square_encrypted;
    evaluator.square(x_plus_1_encrypted, x_plus_and_square_encrypted);
    evaluator.relinearize_inplace(x_plus_and_square_encrypted, relin_keys);

    cout << "scale of (x + 1)^2 befor rescale " <<
log2(x_plus_and_square_encrypted.scale()) << "bits" << endl;
    evaluator.rescale_to_next_inplace(x_plus_and_square_encrypted);
    cout << "scale of (x + 1)^2 after rescale " <<
log2(x_plus_and_square_encrypted.scale()) << "bits" << endl;
    cout << "-----------------------------------------------------" << endl;

    cout << "make (x^2) and relinearization" << endl;

    Ciphertext x_square_and_plus_encrypted, x_encrypted;
    encryptor.encrypt(x_plain, x_encrypted);
    evaluator.square(x_encrypted, x_square_and_plus_encrypted);
    evaluator.relinearize_inplace(x_square_and_plus_encrypted, relin_keys);

    cout << "scale of (x^2) befor rescale " <<
log2(x_square_and_plus_encrypted.scale()) << "bits" << endl;
    evaluator.rescale_to_next_inplace(x_square_and_plus_encrypted);
    cout << "scale of (x^2) after rescale " <<
log2(x_square_and_plus_encrypted.scale()) << "bits" << endl;
        cout << "make (x^2 + 2)" << endl;

    x_square_and_plus_encrypted.scale() = pow(2.0, 50); // 곱하기를 했으므로
정규화
    evaluator.mod_switch_to_inplace(plain_coeff2,
x_square_and_plus_encrypted.parms_id()); // 더하기를 위해서 레벨 동일화

    evaluator.add_plain_inplace(x_square_and_plus_encrypted, plain_coeff2); // x +
2

    cout << "scale of (x^2 + 2) scale " <<
log2(x_square_and_plus_encrypted.scale()) << "bits" << endl;
    cout << "-----------------------------------------------------" << endl;

    Ciphertext result_encrypted;
```

```
    cout << "make result_encrypted and relinearization" << endl;

    evaluator.multiply_inplace(x_plus_and_square_encrypted,
x_square_and_plus_encrypted);
    evaluator.relinearize_inplace(x_plus_and_square_encrypted, relin_keys);

    cout << "scale of result before rescale " <<
log2(x_plus_and_square_encrypted.scale()) << "bits" << endl;
    evaluator.rescale_to_next_inplace(x_plus_and_square_encrypted);
    cout << "scale of result after rescale " <<
log2(x_plus_and_square_encrypted.scale()) << "bits" << endl;
    cout << "------------------------------------------------------" << endl;

    Plaintext p;

    decryptor.decrypt(x_plus_and_square_encrypted, p);

    vector<double> result;
    encoder.decode(p, result);



    cout << "True answer (x + 1)^2 * (x^2 + 2)" << endl;
    for (int i = 1; i <= 10; i++) {
        cout << (i + 1) * (i + 1) * (i * i + 2) << " ";
    }
    cout << endl << "------------------------------------------------------" <<
endl;

    cout << "Decode result " << endl;

    for (int i = 0; i < 10; i++) {
        cout << result[i] << " ";
    }
    cout << endl << "------------------------------------------------------" <<
endl;
    cout << "Decode result Apply Rounding" << endl;

    for (int i = 0; i < 10; i++) {
        cout << round(result[i]) << " ";
    }
    cout << endl << "------------------------------------------------------" <<
endl;

    Ciphertext rotated;
    cout << "Rotate 2 steps left." << endl;
    evaluator.rotate_vector(x_plus_and_square_encrypted, 2, gal_keys, rotated);
```

```
    decryptor.decrypt(rotated, p);
    vector<double> rotate_result;
    encoder.decode(p, rotate_result);
    for (int i = 0; i < 10; i++) {
        cout << round(rotate_result[i]) << " ";
    }
    cout << endl << "------------------------------------------------------" << 
endl;

    return 0;
}
```

```
----- (x + 1)^2 * (x^2 + 2) and left 2 ---- CKKS

Number of slots: 8192

------------------------------------------------------
make (x + 1)
scale of (x + 1) : 50bits
make (x + 1)^2 and relinearization
scale of (x + 1)^2 befor rescale 100bits
scale of (x + 1)^2 after rescale 50bits
------------------------------------------------------
make (x^2) and relinearization
scale of (x^2) befor rescale 100bits
scale of (x^2) after rescale 50bits
make (x^2 + 2)
scale of (x^2 + 2) scale 50bits
------------------------------------------------------
make result_encrypted and relinearization
scale of result before rescale 100bits
scale of result after rescale 50bits
------------------------------------------------------
True answer (x + 1)^2 * (x^2 + 2)
12 54 176 450 972 1862 3264 5346 8300 12342
------------------------------------------------------
Decode result
11.9062 53.8438 175.844 450.219 972.062 1862.12 3263.47 5346.34 8299.97 12341.6
------------------------------------------------------
Decode result Apply Rounding
12 54 176 450 972 1862 3263 5346 8300 12342
------------------------------------------------------
Rotate 2 steps left.
176 450 972 1863 3264 5346 8300 12342 17712 24674
------------------------------------------------------
```