Openfhe를 사용하여 ckks 방식으로 암호화한 암호문을 연산하는 코드 작성.

Key Swiching : 암호문을 연산하다보면 곱셈시 암호의 키가 s 였다면 s^2을 키로 갖고 있는 것처럼 보이는 경우가 발생함 -> 부자연스러움 다시 키를 s로 바꿔주는 작업

Hoisting : dnum이 같으므로 암호문의 decomposition 작업을 선행하고 other 작업을 수행.

과제 : (x^2 + 2) * (x + 1)^2

아래 소스코드

```
#include <iostream>
#include <openfhe.h>

#define PROFILE

using namespace std;
using namespace lbcrypto;

void AutomaticRescaleDemo(ScalingTechnique scalTech);
void ManualRescaleDemo(ScalingTechnique scalTech);

int main(){
    AutomaticRescaleDemo(FLEXIBLEAUTO);
    AutomaticRescaleDemo(FIXEDAUTO);
    ManualRescaleDemo(FIXEDAUTO);
}

void ManualRescaleDemo(ScalingTechnique scalTech){
    std::cout << "\n\n\n ===== FixedManualDemo ============== " <<
std::endl;

    uint32_t batchSize = 8;
    CCParams<CryptoContextCKKSRNS> parameters;
    parameters.SetMultiplicativeDepth(5);
    parameters.SetScalingModSize(50);
    parameters.SetBatchSize(batchSize);

    CryptoContext<DCRTPoly> cc = GenCryptoContext(parameters);

    std::cout << "CKKS scheme is using ring dimension " << cc-
>GetRingDimension() << std::endl << std::endl;

    cc->Enable(PKE);
```

```
    cc->Enable(KEYSWITCH);
    cc->Enable(LEVELEDSHE);

    auto keys = cc->KeyGen();
    cc->EvalMultKeyGen(keys.secretKey);

    // Input
    std::vector<double> x = {1.0, 1.01, 1.02, 1.03, 1.04, 1.05, 1.06, 1.07};
    Plaintext ptxt        = cc->MakeCKKSPackedPlaintext(x);

    std::cout << "Input x: " << ptxt << std::endl;

    auto c = cc->Encrypt(keys.publicKey, ptxt);

    auto c_sq = cc->EvalMult(c,c);
    auto c_sq1 = cc->Rescale(c_sq); // x^2

    auto c_sq_plus = cc->EvalAdd(c_sq1, 2.0);
    auto c_sq_plus1 = cc->Rescale(c_sq_plus); // x^2 + 2

    auto c_plus_sq = cc->EvalMult(cc->EvalAdd(c, 1.0), cc->EvalAdd(c, 1.0));
    auto c_plus_sq1 = cc->Rescale(c_plus_sq);

    auto result = cc->EvalMult(c_sq_plus1, c_plus_sq1);
    auto result1 = cc->Rescale(result);

    Plaintext plainresult;
    std::cout.precision(8);

    cc->Decrypt(keys.secretKey, result1, &plainresult);
    plainresult->SetLength(batchSize);
    std::cout << "(x + 1)^2 * (x^2 + 2) : " << plainresult << std::endl;

    TimeVar t;
    uint32_t dnum = 2;

    std::cout << "- Using HYBRID key switching with " << dnum << " digits" <<
std::endl << std::endl;

    cc->EvalRotateKeyGen(keys.secretKey, {2});

    TIC(t);
    auto rotationRsult = cc->EvalRotate(result1, 2);
    double time2digit = TOC(t);

    Plaintext RotationResult;
    cout.precision(8);
```

```
    cc->Decrypt(keys.secretKey, rotationRsult, &RotationResult);
    RotationResult->SetLength(batchSize);
    cout << "Rotation Result : " << RotationResult << endl;
    cout << "2 rotations with HYBRID (2 digits) took " << time2digit << "ms" <<
endl;


}

void AutomaticRescaleDemo(ScalingTechnique scalTech){
    if (scalTech == FLEXIBLEAUTO) {
        std::cout << std::endl << std::endl << std::endl << " =====
FlexibleAutoDemo ============= " << std::endl;
    }
    else {
        std::cout << std::endl << std::endl << std::endl << " =====
FixedAutoDemo ============= " << std::endl;
    }

    uint32_t batchSize = 8;
    CCParams<CryptoContextCKKSRNS> parameters;
    parameters.SetMultiplicativeDepth(3); // 몇번곱할 수 있는가? 깊이 설정
    parameters.SetScalingModSize(50); // 스케일 2^50
    parameters.SetScalingTechnique(scalTech);
    parameters.SetBatchSize(batchSize);

    CryptoContext<DCRTPoly> cc = GenCryptoContext(parameters); // 설정 파라미
터로 생성

    std::cout << "CKKS scheme is using ring dimension " << cc-
>GetRingDimension() << std::endl << std::endl;

    cc->Enable(PKE);
    cc->Enable(KEYSWITCH);
    cc->Enable(LEVELEDSHE);

    auto keys = cc->KeyGen();
    cc->EvalMultKeyGen(keys.secretKey);

    // Input
    std::vector<double> x = {1.0, 1.01, 1.02, 1.03, 1.04, 1.05, 1.06, 1.07};
    Plaintext ptxt        = cc->MakeCKKSPackedPlaintext(x);

    std::cout << "Input x: " << ptxt << std::endl;

    auto c = cc->Encrypt(ptxt, keys.publicKey);
```

```
    auto c_plus_1      = cc->EvalAdd(c, 1.0);                    // x + 1
    auto c_plus_1_sq   = cc->EvalMult(c_plus_1, c_plus_1);       // (x + 1)^2
    auto c_sq          = cc->EvalMult(c, c);                     // x^2
    auto c_sq_plus_2   = cc->EvalAdd(c_sq, 2.0);                 // x^2 + 2
    auto ccresult      = cc->EvalMult(c_plus_1_sq, c_sq_plus_2);

    Plaintext result;
    std::cout.precision(8);

    cc->Decrypt(ccresult, keys.secretKey, &result);
    result->SetLength(batchSize);
    std::cout << "(x + 1)^2 * (x^2 + 2) : " << result << std::endl;



    TimeVar t;
    uint32_t dnum = 2;

    std::cout << "- Using HYBRID key switching with " << dnum << " digits" <<
std::endl << std::endl;

    cc->EvalRotateKeyGen(keys.secretKey, {2});

    TIC(t);
    auto rotationRsult = cc->EvalRotate(ccresult, 2);
    double time2digit = TOC(t);

    Plaintext RotationResult;
    cout.precision(8);

    cc->Decrypt(keys.secretKey, rotationRsult, &RotationResult);
    RotationResult->SetLength(batchSize);
    cout << "Rotation Result : " << RotationResult << endl;
    cout << "2 rotations with HYBRID (2 digits) took " << time2digit << "ms" <<
endl;
}
```

결과

```
 ===== FlexibleAutoDemo =============
CKKS scheme is using ring dimension 16384

Input x: (1, 1.01, 1.02, 1.03, 1.04, 1.05, 1.06, 1.07,  ... ); Estimated precision: 50 bits

(x + 1)^2 * (x^2 + 2) : (12, 12.201506, 12.406048, 12.613663, 12.824387, 13.038256, 13.255309, 13.475582,  ... ); Estimated precision: 34 bits

- Using HYBRID key switching with 2 digits

Rotation Result : (12.406048, 12.613663, 12.824387, 13.038256, 13.255309, 13.475582, 12, 12.201506,  ... ); Estimated precision: 34 bits

2 rotations with HYBRID (2 digits) took 0ms


 ===== FixedAutoDemo =============
CKKS scheme is using ring dimension 16384

Input x: (1, 1.01, 1.02, 1.03, 1.04, 1.05, 1.06, 1.07,  ... ); Estimated precision: 50 bits

(x + 1)^2 * (x^2 + 2) : (12, 12.201506, 12.406048, 12.613663, 12.824387, 13.038256, 13.255309, 13.475582,  ... ); Estimated precision: 35 bits

- Using HYBRID key switching with 2 digits

Rotation Result : (12.406048, 12.613663, 12.824387, 13.038256, 13.255309, 13.475582, 12, 12.201506,  ... ); Estimated precision: 35 bits

2 rotations with HYBRID (2 digits) took 0ms


 ===== FixedManualDemo =============
CKKS scheme is using ring dimension 32768

Input x: (1, 1.01, 1.02, 1.03, 1.04, 1.05, 1.06, 1.07,  ... ); Estimated precision: 50 bits

(x + 1)^2 * (x^2 + 2) : (12, 12.201506, 12.406048, 12.613663, 12.824387, 13.038256, 13.255309, 13.475582,  ... ); Estimated precision: 39 bits

- Using HYBRID key switching with 2 digits

Rotation Result : (12.406048, 12.613663, 12.824387, 13.038256, 13.255309, 13.475582, 12, 12.201506,  ... ); Estimated precision: 39 bits

2 rotations with HYBRID (2 digits) took 0ms
```