TraceCipherText 구성

생성일	@2023년 8월 10일 오후 5:19
태그	

암호 디버그를 돕는 새로운 자료형 TraceCipherText

Class 구성

TraceCipherText 는 SEAL 라이브러리의 Replace 방식과 Return 방식을 모방하여 구현하였다.

상속의 방법은 계속 시도 하였으나 많은 어려움이 있어 클래스를 구성하는 방식을 계속하였다.

멤버변수

- Ciphertext ct
- Cyptocontext cc
- privatekey pk
- vector<double> trueValue :: 실제로 정답인 값을 갖고 있음
- errorStandard :: 어느정도로 값이 커지면 에러로 간주. 디폴트 값은 10만 사용자가 지정가능

메소드

- Getter, Setter
- show_Detaile : 현재 암호문의 레벨, 스케일, 암호문의 복호문, 원문을 출력하고 에러를 체크
- show_decode : 해독문을 출력
- decode : 해독한 평문을 리턴
- errorcheck : 한 원소가 설정 기준보다 높으면 경고
- replace_add(TraceCipherText &) : 들어온 추적 암호문과 더하여 현재 추적 암호문과 truevalue 값을 갱신
- replace_add(double &) : 들어온 소수를 현재 추적 암호문에 더하여 갱신
- replace_mul(TraceCipherText &) : 들어온 추적 암호문과 곱하여 현재 추적 암호문을 갱신
- add(double number) : replace와 달리 새로운 추적 암호문을 리턴함 현재 암호문은 그대로 유지됨
- add(TraceCiphertext &) : 위와 같음
- Mul(TraceCipherText &) : 위와 같음
- Rescale : 현재 암호문을 재스케일

class 원문

```
class TraceCipherText{
    private:
        vector<double> trueValue;
        Ciphertext<DCRTPoly> ct;
        CryptoContext<DCRTPoly> cc;
        PrivateKey<DCRTPoly> pk;
        double errorStandard = 100000;
    public:
        TraceCipherText(vector<double> tv, Ciphertext<DCRTPoly> a, CryptoContext<DCRTPoly> cc, PrivateKey<DCRTPoly> pk)
        : trueValue(tv), ct(a), cc(cc), pk(pk) {
            //cout << "암호문 생성" << endl;
        }

        void showDetatil(){
            cout << "scale : " << log2(ct->GetScalingFactor()) << endl;
            cout << "Level : " << ct->GetLevel() << endl;
            cout << "true value   : (";
            for(auto iter : trueValue){
                cout << iter << ", ";
            }
            cout << ")" << endl;
            cout << "decode value : ";
            show_decode();
            errorCheck();
            cout << endl;
        }

        Ciphertext<DCRTPoly> getCiphertext(){
            return ct;
        }
        vector<double> getTrueValue(){
            return trueValue;
        }

        void setError(double error){
            errorStandard = error;
        }

        void show_decode(){
            Plaintext plaintext;
            cc->Decrypt(pk, ct, &plaintext);
            plaintext->SetLength(8);
            cout << plaintext;
        }

        Plaintext decode(){
            Plaintext plaintext;
            cc->Decrypt(pk, ct, &plaintext);
            plaintext->SetLength(8);
            return plaintext;
        }
        void errorCheck(){
            for(int i = 0; i < trueValue.size(); i++){
                if((double)decode()->GetRealPackedValue()[i] >= errorStandard){ // 기준 에러보다 큼
                    cout << "WARNING ! 기준 에러 " << errorStandard << " 보다 큽니다." << endl;
                    cout << "index : " << i << endl;
                    cout << "value : " << (double)decode()->GetRealPackedValue()[i] << endl;
                    cout << endl;
                }
            }
        }
        void replace_add(TraceCipherText traceciphertext){
            ct = cc->EvalAdd(ct, traceciphertext.getCiphertext());
            for(int i = 0; i < trueValue.size(); i++){
                trueValue[i] += traceciphertext.getTrueValue()[i];
            }
            showDetatil();
        }

        void replace_add(double number){
            ct = cc->EvalAdd(ct, number);
            for(int i = 0; i < trueValue.size(); i++){
                trueValue[i] += number;
            }
            showDetatil();
        }
```

```
        void replace_Mul(TraceCipherText traceciphertext){
            ct = cc->EvalMult(ct, traceciphertext.getCiphertext());
            for(int i = 0; i < trueValue.size(); i++){
                trueValue[i] *= traceciphertext.getTrueValue()[i];
            }
            showDetatil();
        }

        TraceCipherText add(double number){
            Ciphertext<DCRTPoly> newciphertext = cc->EvalAdd(ct, number);
            vector<double> newVector(trueValue.size(), 0);

            for(int i = 0; i < trueValue.size(); i++){
                newVector[i] = trueValue[i] + number;
            }
            return TraceCipherText(newVector, newciphertext, cc, pk);
        }

        TraceCipherText add(TraceCipherText traceciphertext){
            Ciphertext<DCRTPoly> newciphertext = cc->EvalAdd(ct, traceciphertext.getCiphertext());

            vector<double> newVector(trueValue.size(), 0);

            for(int i = 0; i < trueValue.size(); i++){
                newVector[i] += traceciphertext.getTrueValue()[i];
            }
            //showDetatil();
            return TraceCipherText(newVector, newciphertext, cc, pk);
        }

        TraceCipherText Mul(TraceCipherText traceciphertext){
            Ciphertext<DCRTPoly> newciphertext = cc->EvalMult(ct, traceciphertext.getCiphertext());

            vector<double> newVector(trueValue.size(), 0);

            for(int i = 0; i < trueValue.size(); i++){
                newVector[i] = trueValue[i] * traceciphertext.getTrueValue()[i];
            }
            //showDetatil();
            return TraceCipherText(newVector, newciphertext, cc, pk);
        }

        void Rescale(){
            this->ct = cc->Rescale(ct);
        }

};
```

example

1. 추적 암호문 생성

```
Plaintext ptxt        = cc->MakeCKKSPackedPlaintext(x);
    std::cout << "Input x: " << ptxt << std::endl;

    TraceCipherText a(x, cc->Encrypt(keys.publicKey, ptxt),cc, keys.secretKey);
    TraceCipherText b(x, cc->Encrypt(keys.publicKey, ptxt),cc, keys.secretKey);
```

TraceCipherText는 생성자로 원문 x, 암호문, cc, privatekey를 받는다.

2. Replace 방식으로 (x + 1)^2 * (x^2 + 2) 계산

```
cout << "================== TRACE TEST ==================" << endl;

    a.replace_add(1.0); // x + 1
    a.replace_Mul(a); // (x + 1)^2
    b.replace_Mul(b); // x^2

    b.replace_add(2.0); // x^2 + 2
    a.replace_Mul(b); // (x + 1)^2 * (x^2 + 2)
    a.showDetatil();    // replace 방식
```

```
Input x: (1, 1.01, 1.02, 3, 1.04, 1.05, 1.06, 1.07,  ... ); Estimated precision: 50 bits

================== TRACE TEST ==================
scale : 69.584964
Level : 0
true value   : (2, 2.01, 2.02, 4, 2.04, 2.05, 2.06, 2.07, )
decode value : (2, 2.01, 2.02, 4, 2.04, 2.05, 2.06, 2.07,  ... ); Estimated precision: 43 bits

scale : 100
Level : 1
true value   : (4, 4.0401, 4.0804, 16, 4.1616, 4.2025, 4.2436, 4.2849, )
decode value : (4, 4.0401, 4.0804, 16, 4.1616, 4.2025, 4.2436, 4.2849,  ... ); Estimated precision: 41 bits

scale : 100
Level : 1
true value   : (1, 1.0201, 1.0404, 9, 1.0816, 1.1025, 1.1236, 1.1449, )
decode value : (1, 1.0201, 1.0404, 9, 1.0816, 1.1025, 1.1236, 1.1449,  ... ); Estimated precision: 41 bits

scale : 100
Level : 1
true value   : (3, 3.0201, 3.0404, 11, 3.0816, 3.1025, 3.1236, 3.1449, )
decode value : (3, 3.0201, 3.0404, 11, 3.0816, 3.1025, 3.1236, 3.1449,  ... ); Estimated precision: 41 bits

scale : 100
Level : 2
true value   : (12, 12.201506, 12.406048, 176, 12.824387, 13.038256, 13.255309, 13.475582, )
decode value : (12, 12.201506, 12.406048, 176, 12.824387, 13.038256, 13.255309, 13.475582,  ... ); Estimated precision: 37 bits

scale : 100
Level : 2
true value   : (12, 12.201506, 12.406048, 176, 12.824387, 13.038256, 13.255309, 13.475582, )
decode value : (12, 12.201506, 12.406048, 176, 12.824387, 13.038256, 13.255309, 13.475582,  ... ); Estimated precision: 37 bits
```

위에서 선언한 a,b 값이 계속 갱신되어간다. 새롭게 선언할 필요가 없어 간편하지만, 암호문을 재활용할 수는 없다. 또한, 에러를 일으킬 정도로 큰 수가 발견되지 않아 에러 메세지가 출력되지 않았다.

그리고 replace 함수들은 자동으로 showDetail을 출력하도록 구성하였다. 아무래도 같은 값을 갱신시키다보니 되돌리기 힘들어 정확해야한다고 생각하였다.

3. 새로운 암호문 반환 방식

```
    TraceCipherText c(x, cc->Encrypt(keys.publicKey, ptxt),cc, keys.secretKey);

    TraceCipherText x_plus_1 = c.add(1.0);      // x + 1
    TraceCipherText x_plus_1_sq = x_plus_1.Mul(x_plus_1);   // (x + 1)^2
    TraceCipherText x_sq = c.Mul(c); // x^2
    TraceCipherText x_sq_puls_2 = x_sq.add(2.0); // x^2 + 2
    TraceCipherText result = x_plus_1_sq.Mul(x_sq_puls_2); // (x + 1)^2 * (x^2 + 2)

    result.showDetatil(); // 새로운 값으로 리턴하는 방식
```

```
================return method ===================
scale : 100
Level : 2
true value   : (12, 12.201506, 12.406048, 176, 12.824387, 13.038256, 13.255309, 13.475582, )
decode value : (12, 12.201506, 12.406048, 176, 12.824387, 13.038256, 13.255309, 13.475582,  ... ); Estimated precision: 37 bits
```

새롭게 암호문을 반환하고 저장하는 방식이다. 반환형에는 showDetail이 포함되어있지 않다. 사용자가 보고 싶을 때만 보는 것이 나을 것이라 판단하였다.

4. 에러 경고 메세지

0번째 인덱스의 값을 크게 설정

```
std::vector<double> x = {1000.0, 1.01, 1.02, 3.0, 1.04, 1.05, 1.06, 1.07};
Plaintext ptxt        = cc->MakeCKKSPackedPlaintext(x);
std::cout << "Input x: " << ptxt << std::endl;
```

```
scale : 69.584964
Level : 0
true value   : (1001, 2.01, 2.02, 4, 2.04, 2.05, 2.06, 2.07, )
decode value : (1001, 2.01, 2.02, 4, 2.04, 2.05, 2.06, 2.07,  ... ); Estimated precision: 42 bits

scale : 100
Level : 1
true value   : (1002001, 4.0401, 4.0804, 16, 4.1616, 4.2025, 4.2436, 4.2849, )
decode value : (1002001, 4.0401, 4.0804, 16, 4.1616, 4.2025, 4.2436, 4.2849,  ... ); Estimated precision: 32 bits
WARNING ! 기준 에러 100000 보다 큽니다.
index : 0
value : 1002001


scale : 100
Level : 1
true value   : (1000000, 1.0201, 1.0404, 9, 1.0816, 1.1025, 1.1236, 1.1449, )
decode value : (1000000, 1.0201, 1.0404, 9, 1.0816, 1.1025, 1.1236, 1.1449,  ... ); Estimated precision: 32 bits
WARNING ! 기준 에러 100000 보다 큽니다.
index : 0
value : 1000000


scale : 100
Level : 1
true value   : (1000002, 3.0201, 3.0404, 11, 3.0816, 3.1025, 3.1236, 3.1449, )
decode value : (1000002, 3.0201, 3.0404, 11, 3.0816, 3.1025, 3.1236, 3.1449,  ... ); Estimated precision: 32 bits
WARNING ! 기준 에러 100000 보다 큽니다.
index : 0
value : 1000002


scale : 100
Level : 2
true value   : (1.002003e+12, 12.201506, 12.406048, 176, 12.824387, 13.038256, 13.255309, 13.475582, )
decode value : (1.002003e+12, 12.201009, 12.405675, 175.9999, 12.824402, 13.038273, 13.255275, 13.475783,  ... ); Estimated precision: 11 b
its
WARNING ! 기준 에러 100000 보다 큽니다.
index : 0
value : 1.002003e+12


scale : 100
Level : 2
true value   : (1.002003e+12, 12.201506, 12.406048, 176, 12.824387, 13.038256, 13.255309, 13.475582, )
decode value : (1.002003e+12, 12.201864, 12.405237, 175.99918, 12.824158, 13.038177, 13.254676, 13.475805,  ... ); Estimated precision: 11
bits
WARNING ! 기준 에러 100000 보다 큽니다.
index : 0
value : 1.002003e+12
```

디폴트 에러 검출값인 10만 이상을 포함하고 있는 암호문들에게 인덱스 위치와 값, 경고 메세지 출력 . 5번째 암호문 부터 결과에 오차가 발생한 것을 확인 할 수 있다.