동형 암호 소개 및 실습

https://youtu.be/owmlNMfDAao

한성대학교 HANSUNG UNIVERSITY

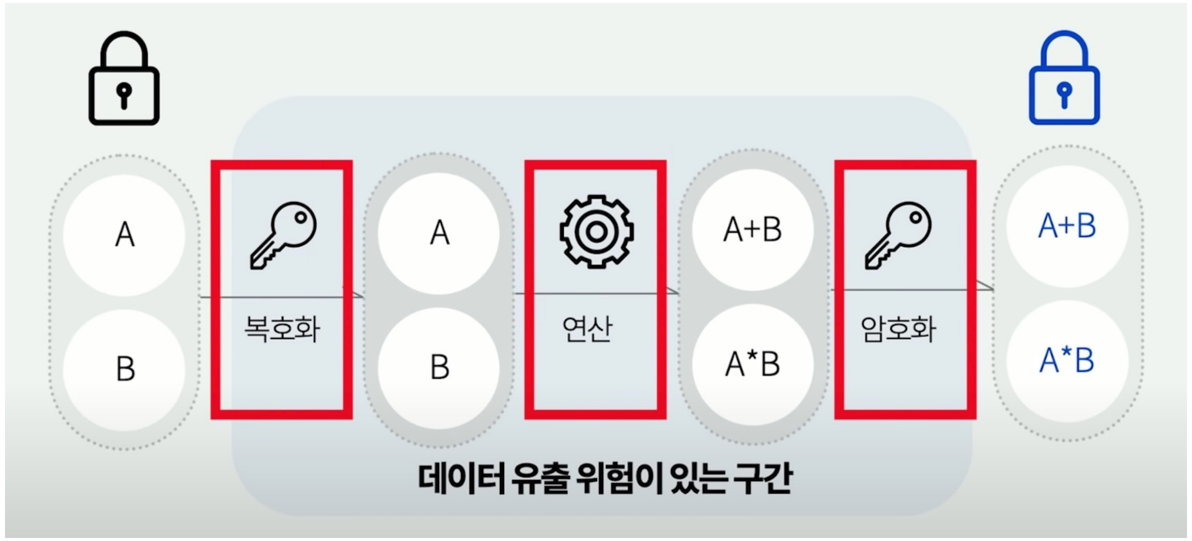
CryptoCraft LAB

1. 동형암호란

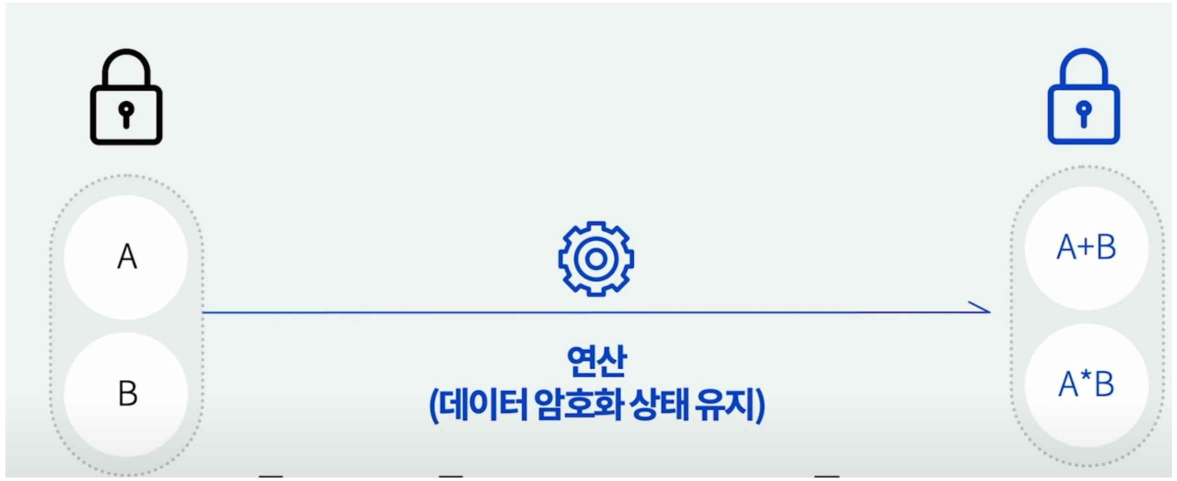
- Homomorphic Encryption
  - 평문과 암호문의 동형(Homomorphic) 성질로 인해 암호문 상태에서도 연산이 가능한 차세대 암호기술
  - 데이터를 기존 암호알고리즘을 통해 비식별화 하는 경우 클라우드에서 암호화된 데이터에 대한 연산이 불가능
  - 동형암호를 활용하면 민감 정보를 안전하게 보호하면서도 유용하게 데이터를 활용할 수 있음

1. 동형암호란

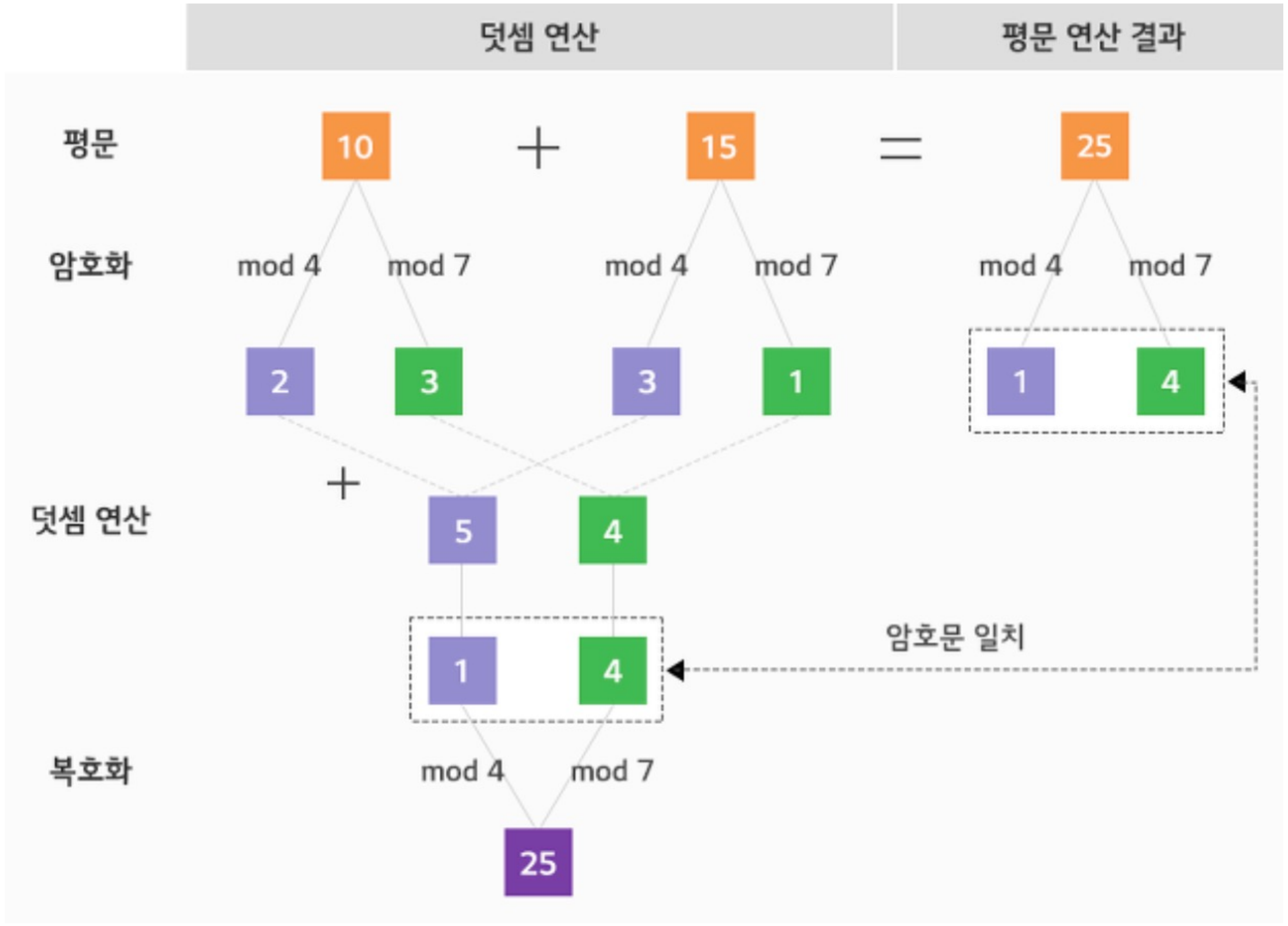
- 기존 암호알고리즘에서의 연산 과정
  - 복호화 → 연산 → 암호화



- 동형암호를 활용한 연산과정
  - 암호화된 상태에서 연산



1. 동형암호란



덧셈 연산
평문 연산 결과
평문
10
+
15
=
25
암호화
mod 4
mod 7
mod 4
mod 7
mod 4
mod 7
2
3
3
1
1
4
+
덧셈 연산
5
4
1
4
암호문 일치
복호화
mod 4
mod 7
25

2. 동형 암호 오픈소스 라이브러리

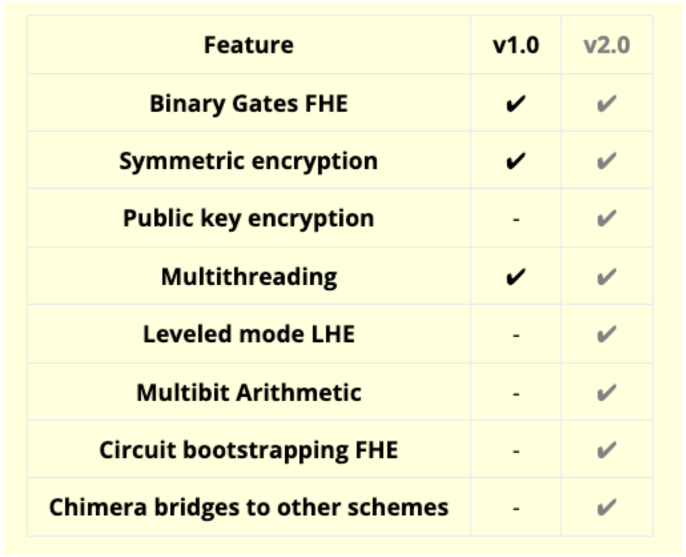
국내·외 동형암호 오픈소스 라이브러리 현황

국가	라이브러리명	개발년도	개발사/기관	비고
미국	HElib	2013	IBM	최초 동형암호 라이브러리 BGV 스킴 지원
	SEAL	2015	마이크로소프트	BFV 스킴 지원
	PALISADE 2	2017	MIT	격자 기반 암호 라이브러리
	cuHE	2017	WPI	GPU를 통한 고속화 라이브러리
	cuFHE	2018		
유럽	NFLlib	2016	Sorbonne	유럽 'H2020' HEAT 프로젝트 결과물
	TFHE	2017	KU Leuven	TFHE 스킴 지원 라이브러리
	Lattigo	2019	EPFL	다중 사용자용 라이브러리
한국	HeaAN	2017	서울대	근사 동형암호(CKKS) 스킴 지원 라이브러리

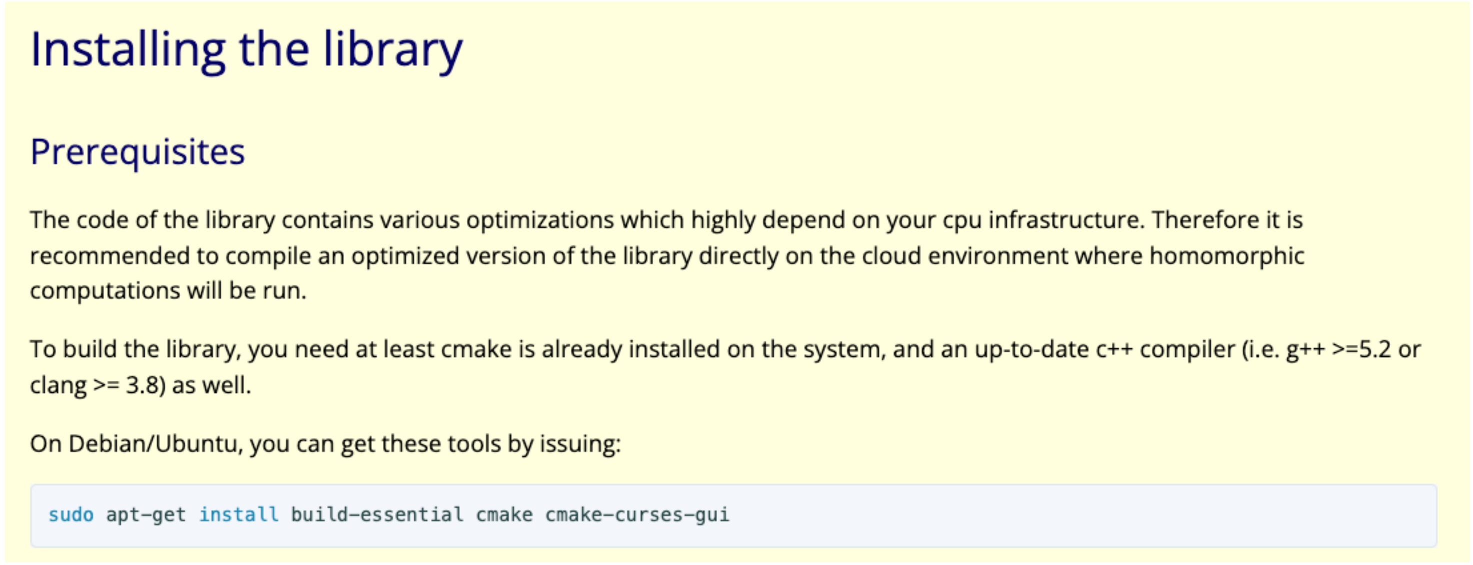
3. 동형 암호 실습

- TFHE

TFHE
A fast open-source library for fully homomorphic encryption.



Feature	v1.0	v2.0
Binary Gates FHE	✔	✔
Symmetric encryption	✔	✔
Public key encryption	-	✔
Multithreading	✔	✔
Leveled mode LHE	-	✔
Multibit Arithmetic	-	✔
Circuit bootstrapping FHE	-	✔
Chimera bridges to other schemes	-	✔



Installing the library

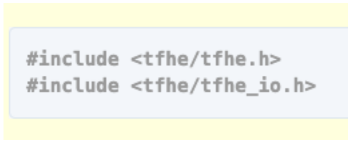
Prerequisites

The code of the library contains various optimizations which highly depend on your cpu infrastructure. Therefore it is recommended to compile an optimized version of the library directly on the cloud environment where homomorphic computations will be run.

To build the library, you need at least cmake is already installed on the system, and an up-to-date c++ compiler (i.e. g++ >=5.2 or clang >= 3.8) as well.

On Debian/Ubuntu, you can get these tools by issuing:

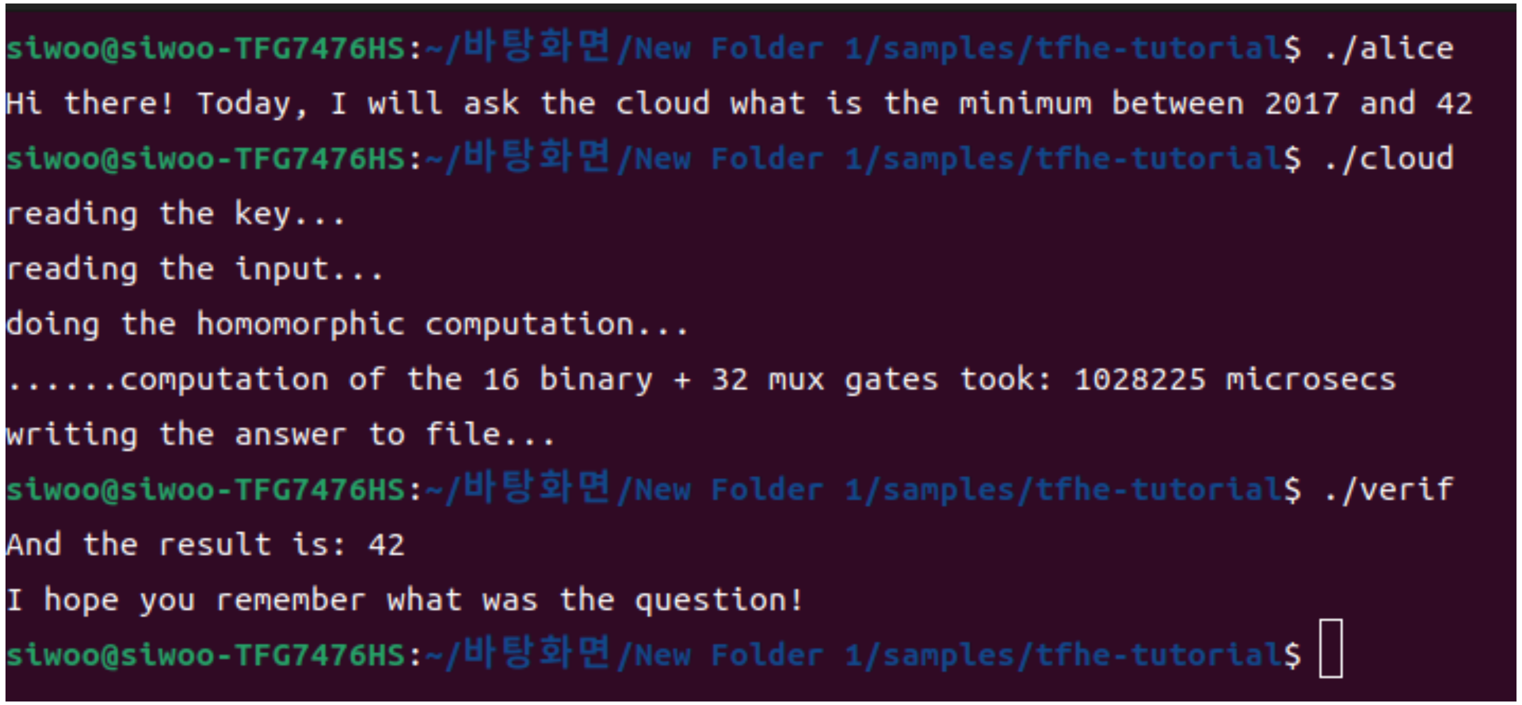
```
sudo apt-get install build-essential cmake cmake-curses-gui
```



```
#include <tfhe/tfhe.h>
#include <tfhe/tfhe_io.h>
```

3. 동형 암호 실습

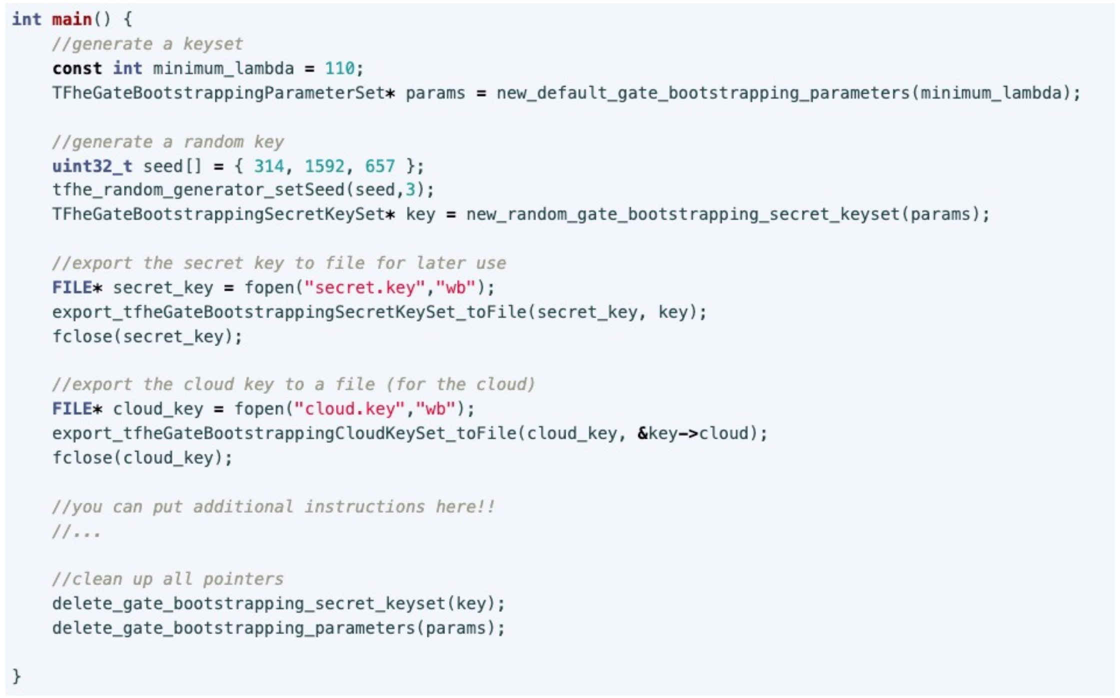
- Hello world 예제 코드 시나리오
  - 1. Alice가 키를 생성하고 두 개의 숫자를 암호화
  - 2. 클라우드에서는 동형 계산을 수행(두 숫자 중 최소값을 찾는 연산)
  - 3. Alice는 클라우드에서의 연산 결과를 복호화 하여 출력



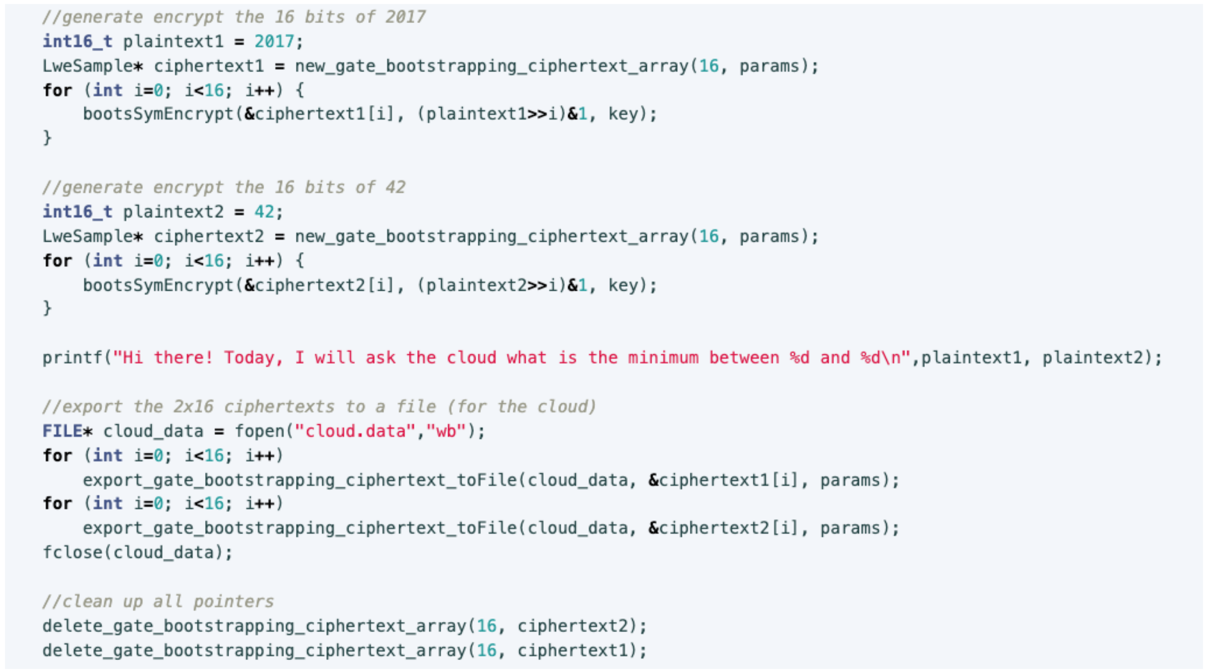
```
siwoo@siwoo-TFG7476HS:~/바탕화면/New Folder 1/samples/tfhe-tutorial$ ./alice
Hi there! Today, I will ask the cloud what is the minimum between 2017 and 42
siwoo@siwoo-TFG7476HS:~/바탕화면/New Folder 1/samples/tfhe-tutorial$ ./cloud
reading the key...
reading the input...
doing the homomorphic computation...
......computation of the 16 binary + 32 mux gates took: 1028225 microsecs
writing the answer to file...
siwoo@siwoo-TFG7476HS:~/바탕화면/New Folder 1/samples/tfhe-tutorial$ ./verif
And the result is: 42
I hope you remember what was the question!
siwoo@siwoo-TFG7476HS:~/바탕화면/New Folder 1/samples/tfhe-tutorial$ 
```

3. 동형 암호 실습

- 1. Alice가 키를 생성하고 두 개의 숫자를 암호화



```c
int main() {
    //generate a keyset
    const int minimum_lambda = 110;
    TFheGateBootstrappingParameterSet* params = new_default_gate_bootstrapping_parameters(minimum_lambda);

    //generate a random key
    uint32_t seed[] = { 314, 1592, 657 };
    tfhe_random_generator_setSeed(seed,3);
    TFheGateBootstrappingSecretKeySet* key = new_random_gate_bootstrapping_secret_keyset(params);

    //export the secret key to file for later use
    FILE* secret_key = fopen("secret.key","wb");
    export_tfheGateBootstrappingSecretKeySet_toFile(secret_key, key);
    fclose(secret_key);

    //export the cloud key to a file (for the cloud)
    FILE* cloud_key = fopen("cloud.key","wb");
    export_tfheGateBootstrappingCloudKeySet_toFile(cloud_key, &key->cloud);
    fclose(cloud_key);

    //you can put additional instructions here!!
    //...

    //clean up all pointers
    delete_gate_bootstrapping_secret_keyset(key);
    delete_gate_bootstrapping_parameters(params);

}
```

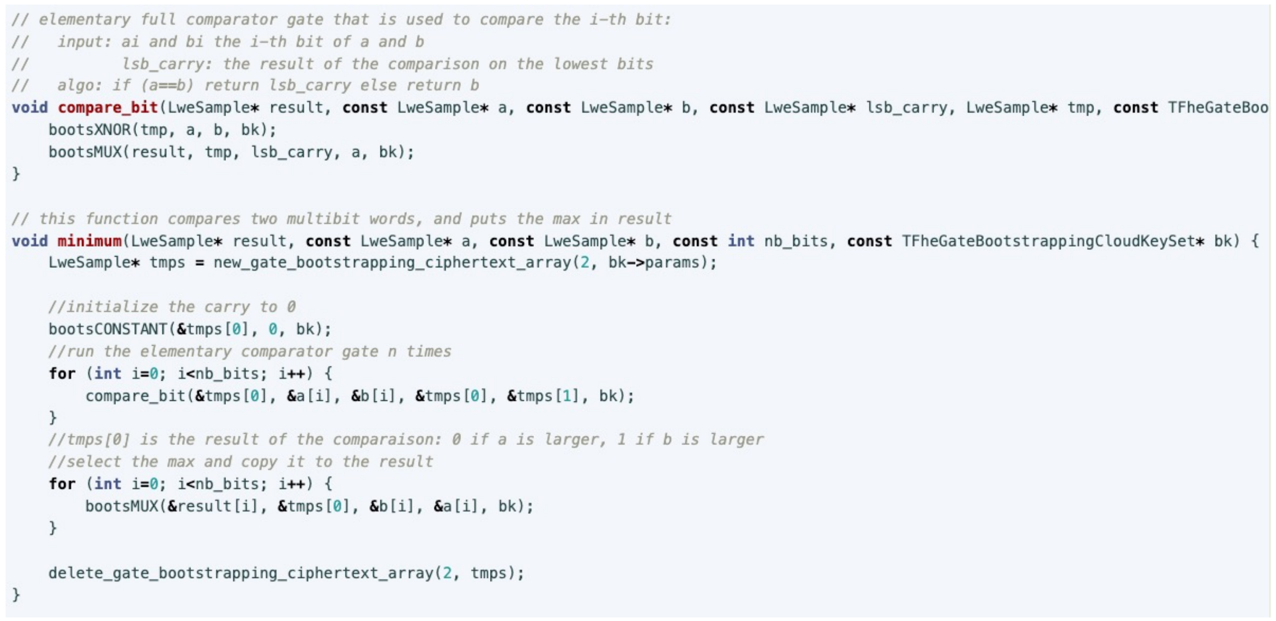


```c
    //generate encrypt the 16 bits of 2017
    int16_t plaintext1 = 2017;
    LweSample* ciphertext1 = new_gate_bootstrapping_ciphertext_array(16, params);
    for (int i=0; i<16; i++) {
        bootsSymEncrypt(&ciphertext1[i], (plaintext1>>i)&1, key);
    }

    //generate encrypt the 16 bits of 42
    int16_t plaintext2 = 42;
    LweSample* ciphertext2 = new_gate_bootstrapping_ciphertext_array(16, params);
    for (int i=0; i<16; i++) {
        bootsSymEncrypt(&ciphertext2[i], (plaintext2>>i)&1, key);
    }

    printf("Hi there! Today, I will ask the cloud what is the minimum between %d and %d\n",plaintext1, plaintext2);

    //export the 2x16 ciphertexts to a file (for the cloud)
    FILE* cloud_data = fopen("cloud.data","wb");
    for (int i=0; i<16; i++)
        export_gate_bootstrapping_ciphertext_toFile(cloud_data, &ciphertext1[i], params);
    for (int i=0; i<16; i++)
        export_gate_bootstrapping_ciphertext_toFile(cloud_data, &ciphertext2[i], params);
    fclose(cloud_data);

    //clean up all pointers
    delete_gate_bootstrapping_ciphertext_array(16, ciphertext2);
    delete_gate_bootstrapping_ciphertext_array(16, ciphertext1);
```

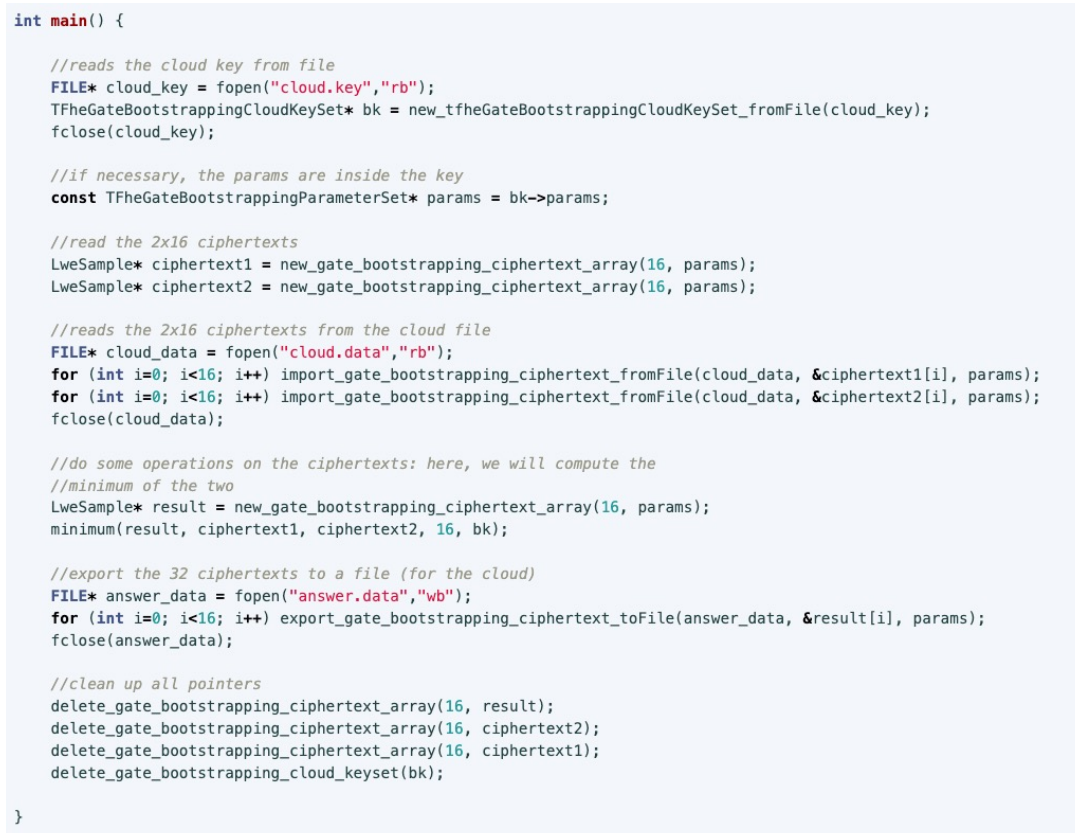
Secret.key size : 113.7MB
Cloud.key size : 113.7MB

3. 동형 암호 실습

- 2. 클라우드에서는 동형 계산을 수행(두 숫자 중 최소값을 찾는 연산)



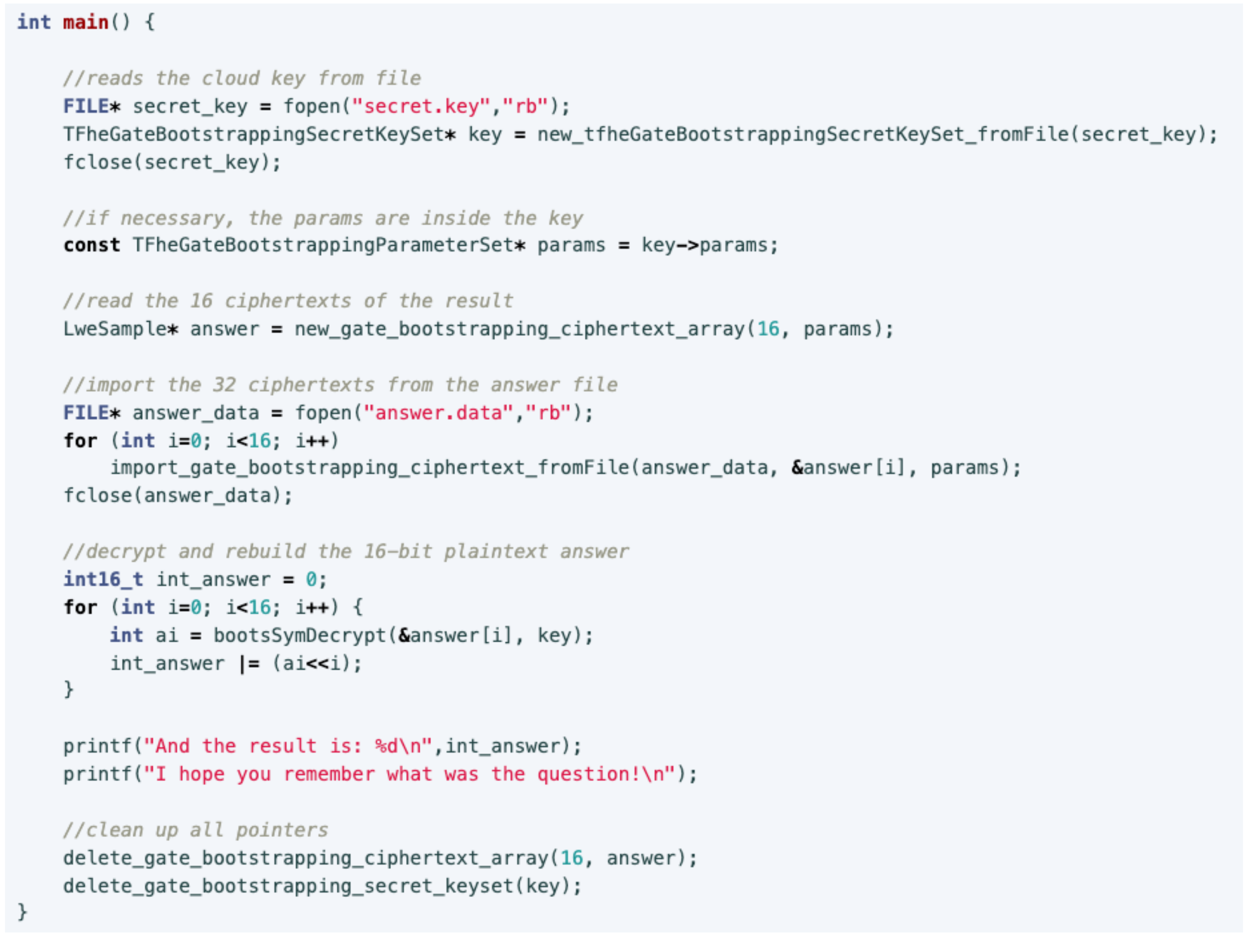
```
// elementary full comparator gate that is used to compare the i-th bit:
//   input: ai and bi the i-th bit of a and b
//          lsb_carry: the result of the comparison on the lowest bits
//   algo: if (a==b) return lsb_carry else return b
void compare_bit(LweSample* result, const LweSample* a, const LweSample* b, const LweSample* lsb_carry, LweSample* tmp, const TFheGateBoo
    bootsXNOR(tmp, a, b, bk);
    bootsMUX(result, tmp, lsb_carry, a, bk);
}

// this function compares two multibit words, and puts the max in result
void minimum(LweSample* result, const LweSample* a, const LweSample* b, const int nb_bits, const TFheGateBootstrappingCloudKeySet* bk) {
    LweSample* tmps = new_gate_bootstrapping_ciphertext_array(2, bk->params);

    //initialize the carry to 0
    bootsCONSTANT(&tmps[0], 0, bk);
    //run the elementary comparator gate n times
    for (int i=0; i<nb_bits; i++) {
        compare_bit(&tmps[0], &a[i], &b[i], &tmps[0], &tmps[1], bk);
    }
    //tmps[0] is the result of the comparaison: 0 if a is larger, 1 if b is larger
    //select the max and copy it to the result
    for (int i=0; i<nb_bits; i++) {
        bootsMUX(&result[i], &tmps[0], &b[i], &a[i], bk);
    }

    delete_gate_bootstrapping_ciphertext_array(2, tmps);
}
```



```
int main() {

    //reads the cloud key from file
    FILE* cloud_key = fopen("cloud.key","rb");
    TFheGateBootstrappingCloudKeySet* bk = new_tfheGateBootstrappingCloudKeySet_fromFile(cloud_key);
    fclose(cloud_key);

    //if necessary, the params are inside the key
    const TFheGateBootstrappingParameterSet* params = bk->params;

    //read the 2x16 ciphertexts
    LweSample* ciphertext1 = new_gate_bootstrapping_ciphertext_array(16, params);
    LweSample* ciphertext2 = new_gate_bootstrapping_ciphertext_array(16, params);

    //reads the 2x16 ciphertexts from the cloud file
    FILE* cloud_data = fopen("cloud.data","rb");
    for (int i=0; i<16; i++) import_gate_bootstrapping_ciphertext_fromFile(cloud_data, &ciphertext1[i], params);
    for (int i=0; i<16; i++) import_gate_bootstrapping_ciphertext_fromFile(cloud_data, &ciphertext2[i], params);
    fclose(cloud_data);

    //do some operations on the ciphertexts: here, we will compute the
    //minimum of the two
    LweSample* result = new_gate_bootstrapping_ciphertext_array(16, params);
    minimum(result, ciphertext1, ciphertext2, 16, bk);

    //export the 32 ciphertexts to a file (for the cloud)
    FILE* answer_data = fopen("answer.data","wb");
    for (int i=0; i<16; i++) export_gate_bootstrapping_ciphertext_toFile(answer_data, &result[i], params);
    fclose(answer_data);

    //clean up all pointers
    delete_gate_bootstrapping_ciphertext_array(16, result);
    delete_gate_bootstrapping_ciphertext_array(16, ciphertext2);
    delete_gate_bootstrapping_ciphertext_array(16, ciphertext1);
    delete_gate_bootstrapping_cloud_keyset(bk);

}
```

3. 동형 암호 실습

- 3. Alice는 클라우드에서의 연산 결과를 복호화 하여 출력



```c
int main() {

    //reads the cloud key from file
    FILE* secret_key = fopen("secret.key","rb");
    TFheGateBootstrappingSecretKeySet* key = new_tfheGateBootstrappingSecretKeySet_fromFile(secret_key);
    fclose(secret_key);

    //if necessary, the params are inside the key
    const TFheGateBootstrappingParameterSet* params = key->params;

    //read the 16 ciphertexts of the result
    LweSample* answer = new_gate_bootstrapping_ciphertext_array(16, params);

    //import the 32 ciphertexts from the answer file
    FILE* answer_data = fopen("answer.data","rb");
    for (int i=0; i<16; i++)
        import_gate_bootstrapping_ciphertext_fromFile(answer_data, &answer[i], params);
    fclose(answer_data);

    //decrypt and rebuild the 16-bit plaintext answer
    int16_t int_answer = 0;
    for (int i=0; i<16; i++) {
        int ai = bootsSymDecrypt(&answer[i], key);
        int_answer |= (ai<<i);
    }

    printf("And the result is: %d\n",int_answer);
    printf("I hope you remember what was the question!\n");

    //clean up all pointers
    delete_gate_bootstrapping_ciphertext_array(16, answer);
    delete_gate_bootstrapping_secret_keyset(key);
}
```

감 사 합 니 다