Information Security and Cryptography Assignment 3

教授：游家牧

網站：https://login.stufinite.faith

林家賢、黃川哲、黃晟慶、陳勇瑜、许萌

1 FHE

1.1 Install HElib

使用 **Ubuntu 14.04 LTS**

What we need?

- GMP
  - 到 `https://gmplib.org/#DOWNLOAD` 選擇`lz`
  - 解壓縮：tar –lzip -xvf gmp-6.1.2.tar.lz
  - cd gmp-6.1.2
  - 檢查需要的 dependencies，產生 Makefile：./configure

```
Version:              GNU MP 6.1.2
Host type:            haswell-pc-linux-gnu
ABI:                  64
Install prefix:       /usr/local
Compiler:             gcc -std=gnu99
Static libraries:     yes
Shared libraries:     yes
```

  - make
    - 第一次報錯：缺 m4
      - sudo apt-get install m4
    - 第二次報錯：/mpn/m4-ccas: Permission Denied
      - chmod +x mpn/m4-ccas
    - 第三次報錯：missing makeinfo
      - sudo apt-get install texinfo
  - 不斷下載缺失的檔案直到沒有報錯後
  - sudo make install

```
 /bin/mkdir -p '/usr/local/include'
 /usr/bin/install -c -m 644 gmp.h '/usr/local/include'
make  install-data-hook
make[4]: Entering directory `/media/psf/Home/Documents/workspace/gmp-6.1.2'

+-------------------------------------------------------------+
| CAUTION:                                                    |
|                                                             |
| If you have not already run "make check", then we strongly  |
| recommend you do so.                                        |
|                                                             |
| GMP has been carefully tested by its authors, but compilers |
| are all too often released with serious bugs.  GMP tends to |
| explore interesting corners in compilers and has hit bugs   |
| on quite a few occasions.                                   |
|                                                             |
+-------------------------------------------------------------+

make[4]: Leaving directory `/media/psf/Home/Documents/workspace/gmp-6.1.2'
make[3]: Leaving directory `/media/psf/Home/Documents/workspace/gmp-6.1.2'
make[2]: Leaving directory `/media/psf/Home/Documents/workspace/gmp-6.1.2'
make[1]: Leaving directory `/media/psf/Home/Documents/workspace/gmp-6.1.2'
```

- NTL
  - 到 http://www.shoup.net/ntl/download.html：選擇 10.3.0
  - 解壓縮
  - cd /ntl/10.3.0
  - `./configure NTL_GMP_LIP=on`
  - make
    - 報錯：g++ no found.
      - sudo apt-get g++

```
ranlib ntl.a #LSTAT
make[1]: Leaving directory `/media/psf/Home/Documents/workspace/ntl-10.3.0/src'
touch all
```

  - sudo make install
    - 最後會在 /usr/local/include 找到

```
parallels@ubuntu:/usr/local/include$ ll
total 96
drwxr-xr-x  3 root root  4096 Dec 18 17:27 ./
drwxr-xr-x 10 root root  4096 Jul 23  2014 ../
-rw-r--r--  1 root root 83715 Dec 18 16:58 gmp.h
drwxr-xr-x  2 root root  4096 Dec 18 17:27 NTL/
```

- 安裝 HElib
  - git clone https://github.com/shaih/HElib.git
  - make
  - make check

```
AltCRT.cpp          DoubleCRT.h           FHE.o               misc                        permutations.h           Test_General.cpp
AltCRT.h            DoubleCRT.o           hypercube.cpp       multicore.h                 permutations.o           Test_IO.cpp
BenesNetwork.cpp    EncryptedArray.cpp    hypercube.h         MulTime.cpp                 polyEval.cpp             Test_LinPoly.cpp
BenesNetwork.o      EncryptedArray.h      hypercube.o         NumbTh.cpp                  polyEval.h               Test_matmul.cpp
bluestein.cpp       EncryptedArray.o      IndexMap.h          NumbTh.h                    polyEval.o               Test_OldEvalMap.cpp
bluestein.h         eqtesting.cpp         IndexSet.cpp        NumbTh.o                    powerful.cpp             Test_PAlgebra.cpp
bluestein.o         eqtesting.o           IndexSet.h          OldEvalMap.cpp              powerful.h               Test_Permutations.cpp
cgauss.cpp          EvalMap.cpp           IndexSet.o          OldEvalMap.h                powerful.o               Test_PolyEval.cpp
cloned_ptr.h        EvalMap.h             KeySwitching.cpp    OldEvalMap.o                recryption.cpp           Test_Powerful.cpp
CModulus.cpp        EvalMap.o             KeySwitching.o      OptimizePermutations.cpp    recryption.h             Test_Replicate.cpp
CModulus.h          extractDigits.cpp     LICENSE.TXT         OptimizePermutations.o      recryption.o             Test_Timing.cpp
CModulus.o          extractDigits.o       Makefile            PAlgebra.cpp                replicate.cpp            timing.cpp
Ctxt.cpp            fhe.a                 matching.cpp        PAlgebra.h                  replicate.h              timing.h
Ctxt.h              FHEContext.cpp        matching.h          PAlgebra.o                  replicate.o              timing.o
Ctxt.o              FHEContext.h          matching.o          params.cpp                  rotations.cpp            tlp.cpp
debugging.cpp       FHEContext.o          matrix.cpp          PermNetwork.cpp             Test_bootstrapping.cpp   tmm.cpp
debugging.o         FHE.cpp               matrix.h            PermNetwork.o               Test_EvalMap.cpp
```

1.2 HElib code

```
#include "FHE.h"
#include "EncryptedArray.h"
#include <NTL/lzz_pXFactoring.h>
#include <fstream>
#include <sstream>
#include <sys/time.h>

int main(int argc, char **argv)
{
    long m=0, p=257, r=1; // Native plaintext space
                         // Computations will be 'modulo p'
    long L=16;           // Levels
    long c=3;            // Columns in key switching matrix
    long w=64;           // Hamming weight of secret key
    long d=0;
    long security = 128;
    ZZX G;
    m = FindM(security,L,c,p, d, 0, 0);

    FHEcontext context(m, p, r);
    // initialize context
    buildModChain(context, L, c);
    // modify the context, adding primes to the modulus chain
    FHESecKey secretKey(context);
    // construct a secret key structure
    const FHEPubKey& publicKey = secretKey;
    // an "upcast": FHESecKey is a subclass of FHEPubKey
    G = context.alMod.getFactorsOverZZ()[0];

    secretKey.GenSecKey(w);
    // actually generate a secret key with Hamming weight w

    addSome1DMatrices(secretKey);
    cout << "Generated key" << endl;

    EncryptedArray ea(context, G);
    // constuct an Encrypted array object ea that is
    // associated with the given context and the polynomial G

    long nslots = ea.size();
    //cout<<"nslots: "<< nslots << endl;

```

- Include some needed header.
- We set some parameters for initialization, basically just copy and paste them.
- ”p” must be a prime number, we commonly hope it the larger the better.
- We use these to create a ”EncryptedArray” named ”ea”, and create ”nslot” equal to ea.

```
vector<long> v1;
v1.push_back(48);
for(int i = 0 ; i < nslots-1; i++) {
    v1.push_back(0);
}

Ctxt ct1(publicKey);


clock_t start = clock();
ea.encrypt(ct1, publicKey, v1);
cout << "v1 encrypt time = " << float(clock() - start)/CLOCKS_PER_SEC << endl;


vector<long> dv1;
start = clock();
ea.decrypt(ct1, secretKey, dv1);
cout << "v1 decrypt time = " << float(clock() - start)/CLOCKS_PER_SEC << endl;
for(int i=0; i<5; i++)
cout << dv1[i] << endl;


vector<long> v2;
Ctxt ct2(publicKey);
v2.push_back(234);
for(int i=0;i<nslots-1;i++)
 v2.push_back(0);
ea.encrypt(ct2, publicKey, v2);

```

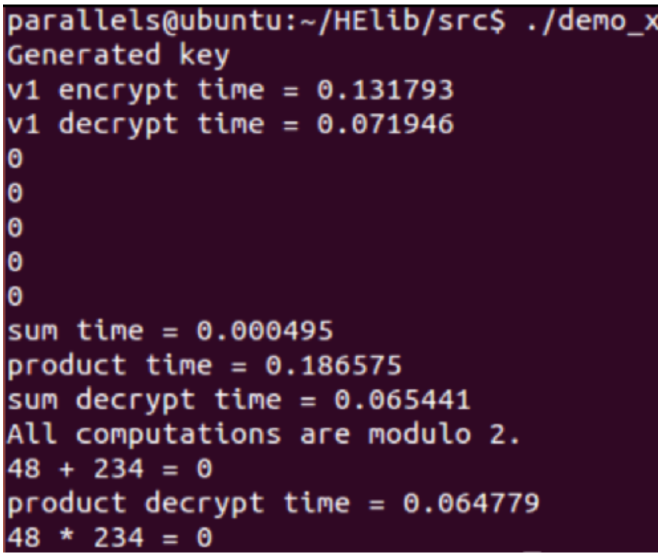
- "vector" is a data structure support by c++. Most of the time you can use it as a array.
- "push_back" can append something to a vector object. We set this vector as 48 and pad 0 until its length is equal to ea.
- Create a "Ctxt" object to store encrypted v1, and measure time Consumption.
- Create a "vector" dv1 to store decrypted ct1, and measure time Consumption.
- Create another "vector" and encrypt it as "Ctxt" named ct2.

```

Ctxt ctSum = ct1;
Ctxt ctProd = ct1;


start = clock();
ctSum += ct2;
cout << "sum time = " << float(clock() - start)/CLOCKS_PER_SEC << endl;

start = clock();
ctProd *= ct2;
cout << "product time = " << float(clock() - start)/CLOCKS_PER_SEC << endl;


 vector<long> res;
 start=clock();
 ea.decrypt(ctSum, secretKey, res);
 cout << "sum decrypt time = " << float(clock() - start)/CLOCKS_PER_SEC << endl;


 cout << "All computations are modulo " << p << "." << endl;

 cout << v1[0] << " + " << v2[0] << " = " << res[0] << endl;

 start = clock();
 ea.decrypt(ctProd, secretKey, res);
 cout << "product decrypt time = " << float(clock() - start)/CLOCKS_PER_SEC << endl;


 cout << v1[0] << " * " << v2[0] << " = " << res[0] << endl;


 return 0;
}

```

- Create two "Ctxt" named ctSum and CtProd.

- Just add and multiply ct1 and ct2.
- Then decrypt them.

1.3 Time



2 PHE

2.1 Install Paillier

Our environment

- Debian Jessie
- Python 2.7.9

Install paillier from Github using pip

- `$ pip install -e git://github.com/mikeivanov/paillier.git#egg=paillier`

2.2 Paillier code

```
#!/usr/bin/env python
from paillier import paillier

def generate_keypair(x):
    return paillier.generate_keypair(x)

def encrypt_x(pub, p):
    return paillier.encrypt(pub, p)

def encrypt_y(pub, p):
    return paillier.encrypt(pub, p)

def decrypt(priv, pub, z):
    return paillier.decrypt(priv, pub, cz)

def add(pub, x, y):
    return paillier.e_add(pub, x, y)


if __name__ == '__main__':
    priv, pub = generate_keypair(512)

    # plaintext用3,5
    x, y = 3, 5
    # 分别做加密
    cx, cy = encrypt_x(pub, x), encrypt_y(pub, y)
    # 取密文做加法
    cz = add(pub, cx, cy)
    # 解密
    z = decrypt(priv, pub, cz)
```

- Paillier is much easier. We only need to import it, and use its function.
- In our code, ”generate keypair” parameter x is the length of key pairs; pub is public key; priv is private key; p is plaintext; cz is ciphertext which we want to decrypt; ”add” parameter x and y are encrypted text.
- So we just use these functions to measure time Consumption, but paillier doesn’t support encrypted multiplication so we don’t need to test it.

2.3 Time

```
3.848sec on demo.py:2(<module>)
0.507sec on demo.py:4(generate_keypair)
1.648sec on demo.py:7(encrypt_x)
1.682sec on demo.py:10(encrypt_y)
0.003sec on demo.py:13(decrypt)
0.000sec on demo.py:16(add)
```