Information Security HW4

B10632026 吳苡瑄

執行環境

- Windows 10
- Python3

作業要求

- 1024 bit 的 RSA 加解密並使用以下方法
  - 產生大質數 (Miller-Rabin Test)
  - Square-and-Multiply
  - Chinese Remainder Theorem

執行方式

- init

```
python3 ./rsa.py init {bits}
```

- encrypt

```
python3 ./rsa.py -e {plaintext} {n} {e}
```

- decrypt

```
python3 ./rsa.py -d {ciphertext} {p} {q} {d}
```

程式碼解說

- generateLargePrime
  - 先用getrandbits生成所需bits長度的數字
  - 在用單一bit的or來做遮罩讓最高位和最低位都是1
  - 最後確認他是不是prime
- isPrime

    - 先用1000以內的質數去檢查
    - 最後再用Miller Rabin Test檢驗
    - 根據網路建議設定是做40次檢驗
- modInverse
    - 找模反元素
    - 利用Extended Euclidean algorithm
- chineseRemainderTheorem
    - 在decrypt的時候用到，作為加速
    - 需要d、p和q

心得

- 主要重點在於生成p、q上，只要成功生出p、q其他問題都很好解決
- 不確定要做幾次Miller Rabin所以上網查
- 生成的密文每個字元用'/'分隔開來
- 解密時把字串依據'/'切開一一運算回去
- 若需要用Chinese Remainder Theorem加速就要有p和q所以在輸入參數中多加了這兩個
- 其實照著講義上的步驟就能造出大部分的code了

執行結果

```
D:\school\InformationSecurity\Information_Security_Class\hw4\B10632026_吳苡瑄 (master -> origin)
$ python .\rsa.py init 1024
 p =  77555777348015101373994056749968116249056300215367171201840403507569250693147445788183080027794147998701896546715002505871972000827760723771
93059288993711
 q =  22103768960010295972750780785749267538541157119041837926358520616698232834897846523342381130548667198317647244129519331221010704973399339939939523527782779
 n =  17142749840145258275319839680104855556666415024587578632072130993696390480127967230457430200553624756607883339217865105975877280684058760576674873169949116173695634452600930353560771350131918950937617762665695179709560661313850113997777083408295254788837141398499133312488517371573800350458591839671790510286 9
 e =  7
 d =  24489642628778940393314056685864079366666307177982255188674472848137700685897096043510614572219463938011261913168378722822681829548655372252392675957070161696801378530600456197917321822177146580369862435443129322076377022568656703912381239056259888543767603735441790300757623643976135332739085894876441189483
```

```
D:\school\InformationSecurity\Information_Security_Class\hw4\B10632026_吳苡瑄 (master -> origin)
$ python .\rsa.py -e hello 17142749840145258275319839680104855556666415024587578632072130993696390480127967230457430200553624756607883339217865105975877280684058760576674873169949116173695634452600930353560771350131918950937617762665695179709560661313850113997777083408295254788837141398499133312488517371573800350458591839671790510286 9

Your ciphertext is :
131593177923584/107213535210701/171382426877952/171382426877952/207616015289871/

D:\school\InformationSecurity\Information_Security_Class\hw4\B10632026_吳苡瑄 (master -> origin)
$ python .\rsa.py -d 131593177923584/107213535210701/171382426877952/171382426877952/207616015289871/ 7755577734801510137399405674996811624905630021536717120184040350756925069314744578818308002779414799870189654671500250587197200082776072377193059288993711 22103768960010295972750780785749267538541157119041837926358520616698232834897846523342381130548667198317647244129519331221010704973399339939939523527782779 24489642628778940393314056685864079366666307177982255188674472848137700685897096043510614572219463938011261913168378722822681829548655372252392675957070161696801378530600456197917321822177146580369862435443129322076377022568656703912381239056259888543767603735441790300757623643976135332739085894876441189483

Your plaintext is :
hello
```