

Họ tên: Nguyễn Việt Thanh Duy

MSSV: 19127378

Assignment_07

1. Diffie-Hellman:

❖ Các hàm quan trọng:

- `keyExchange()`: dùng để thực hiện trao đổi khoá, trả về khoá cuối cùng sau khi trao đổi
- `genPrime()`: dùng để sinh số nguyên tố p
- `primRoot(p)`: dùng để tìm g căn nguyên thủy modulo p

❖ Một số lưu ý:

- Các thư viện sử dụng: `socket`, `numpy`, `random`
- Trước khi chạy chương trình cần cấu hình địa chỉ IPv4 tương ứng của máy tại hàm `connect()`

```
def connect():  
    HOST = '192.168.1.13' # thiết lập ipv4 của máy tại đây  
    PORT = 80  
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
    s.bind((HOST, PORT))  
    s.listen(1)  
    print('Waiting to connect...')  
    client, addr = s.accept()  
    return client
```

- Demo thực hiện trên số các số nguyên 128 bit

❖ Các bước chạy demo:

- Chạy file Alice (đóng vai trò là server nên cần được chạy trước)
- Chạy file Bob
- Trên cả 2 file sẽ xuất hiện khoá cuối cùng sau khi trao đổi (128 bit)

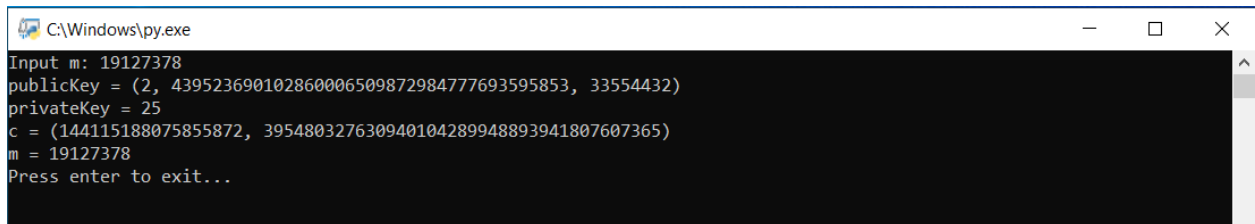


```
C:\Windows\py.exe  
Waiting to connect...  
key = 260353620507253940784925716164780910678  
Press enter to exit...  
  
C:\Windows\py.exe  
key = 260353620507253940784925716164780910678  
Press enter to exit...
```

2. ElGamal:

❖ Các hàm quan trọng:

- `genKey()`: trả về `publicKey` và `privateKey`
 - `encrypt(m,publicKey)`: thực hiện mã hoá bản rõ `m`, trả về bản mã (c_1, c_2)
 - `decrypt(c, publicKey, privateKey)`: thực hiện giải mã bản mã `c`, trả về bản rõ `m`
- ❖ Một số lưu ý:
- Các thư viện sử dụng: `numpy`, `random`
 - Demo thực hiện trên các số nguyên 128 bit
- ❖ Các bước chạy demo:
- Nhập `m` (độ lớn: 128 bit, cơ số 10)
 - Chương trình trả về lần lượt:
 - Khoá công khai `publicKey` dưới dạng (g, p, h)
 - Khoá bí mật `privateKey`
 - Bản mã `c` dưới dạng (c_1, c_2)
 - Bản rõ `m` (để kiểm tra)



```

C:\Windows\py.exe
Input m: 19127378
publicKey = (2, 439523690102860006509872984777693595853, 33554432)
privateKey = 25
c = (144115188075855872, 395480327630940104289948893941807607365)
m = 19127378
Press enter to exit...

```

3. Digital signature:

- ❖ Các hàm quan trọng:
- `genKey()`: trả ra `e, d, n`
 - `sign(m, d, n)`: trả về chữ kí `s`
 - `verify(m, s, e, d)`: dùng để xác minh chữ kí `s`
- ❖ Một số lưu ý:
- Các thư viện sử dụng: `random`
 - Demo thực hiện trên 2 số nguyên tố `p` và `q` 64 bit
- ❖ Các bước chạy chương trình:
- Nhập `m` (độ lớn: 128 bit, cơ số: 10)
 - Chương trình trả về lần lượt:
 - Khoá công khai `e`
 - Khoá bí mật `d`
 - Số tự nhiên `n`
 - Chữ kí `s`
 - Kết quả xác minh `verify`

```
C:\Windows\py.exe
Input m: 19127378
e = 22989602257424782891
d = 100312517088180826916573829334302090871
n = 720215832419502911303935724619650717993
sign = 498971714916127909709334705770781331565
verify = True
Press enter to exit...
```