HW5 B10615043

分工

- B10615013 四資工三甲 李聿鎧： `keygen , Command / File IO`
- B10615043 四資工三甲 何嘉峻： `sign , verify`

開發環境

- **IDE:** PyCharm 2019.2
- **Language:** Python 3.7

操作方式

產生金鑰

```
$ python DSA.py -keygen <bit-length> [<privatekey_name>] [<publickey_path>]
```

進行簽章

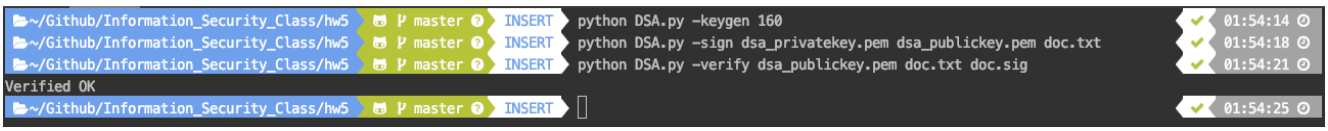
```
$ python DSA.py -sign <privatekey_path> <publickey_path> <message_path>
[<signature_name>]
```

進行驗章

```
$ python DSA.py -verify <publickey_path> <message_path> <signature_name>
```

執行結果

產生金鑰 → 進行簽章 → 進行驗章



`dsa_publickey.pem`

```
-----BEGIN PUBLIC KEY-----
7c53848f09726bbd65586fa61a596daec08b18f5753365dd00f76713a089463871b6844b7fa
e284bcb84b993f7ce1440f08c2afee941b3ff4c5
3f05acb1bb17584e07e1d217c0dfb24085374207f8d49ab6069d6ed36d35edf4bed2c8a7d11
7aedc0970b5ca80fe474c20394ea1acf50853a921bd66dbec2a9e0d6ee1e6f605e6c96673f9
-----END PUBLIC KEY-----
```

dsa_privatekey.pem

```
-----BEGIN DSA PRIVATE KEY-----
3348e157abd0397d6ef8c5b27971a1aec85bfb8c
-----END DSA PRIVATE KEY-----
```

doc.txt

```
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

doc.sig

```
-----BEGIN SIGNATURE-----
2df92602ddd2a270d846db74b6349735d003bc79
3aa289ebf7573399aa81e604b6be3411bfbbdfe6
-----END SIGNATURE-----
```

程式碼解說

DSA.py

- `elif sys.argv[1] == '-sign':`

  會檢查讀入的指令，如果使用者沒有自定義簽名檔案名稱會使用訊息檔案的檔名：

  - signature： `<filename>.sig`

先讀金鑰和訊息後，初始化 `sha1()` 並將訊息進行 hash。

之後產生 k_E : $0 < k_E < q$

計算 r, s 後呼叫 `write_signature()`
$r \equiv (a^{k_E} mod\ p)\ mod\ q$
$s \equiv (SHA(x) + \ d \times r) k_E^{-1}\ mod\ q$

- `elif sys.argv[1] == '-verify':`

先讀金鑰、簽名、訊息後，初始化 `sha1()` 並將訊息進行 hash。

計算 w, v 後檢查 v, r 是否相等
$w \equiv s^{-1} mod\ p$
$v_1 \equiv \alpha^{w \times SHA(x)\ mod\ q}\ mod\ p$
$v_2 \equiv \beta^{w \times r\ mod\ q}\ mod\ p$
$v \equiv (v_1 \times v_2\ mod\ p)\ mod\ q$

```
def main():

    # ~~~~~~~~~~ #

    elif sys.argv[1] == '-sign':
        # Error Handling
        if len(sys.argv) < 5:
            raise Exception(help_message)
        d = read_privatekey(sys.argv[2])
        p, q, a, b = read_publickey(sys.argv[3])
        message_file = open(sys.argv[4], 'r')
        signature_name = os.path.splitext(message_file.name)[0] + '.sig'
        try:
            signature_name = sys.argv[5]
        except IndexError:
            pass

        # Compute hashed message
        sha = hashlib.sha1()
        message = message_file.read().encode('utf-8')
        sha.update(message)
        hash_message = sha.hexdigest()

        # Generate ke, r, s
        ke = randrange(1, q)
        r = square_and_multiply(a, ke, p) % q
        s1 = (int(hash_message, 16) + d * r) % q
        s2 = modinv(ke, q) % q
        s = (s1 * s2) % q

        # Write into files
        write_signature(signature_name, r, s)

    elif sys.argv[1] == '-verify':
        # Error Handling
        if len(sys.argv) < 5:
            raise Exception(help_message)
        p, q, a, b = read_publickey(sys.argv[2])
        message_file = open(sys.argv[3], 'r')
        r, s = read_signature(sys.argv[4])

        # Compute hashed message
        sha = hashlib.sha1()
        message = message_file.read().encode('utf-8')
        sha.update(message)
        hash_message = sha.hexdigest()

        # Generate w, v
        w = modinv(s, q)
        v1 = square_and_multiply(a, (w * int(hash_message, 16)) % q, p)
        v2 = square_and_multiply(b, (w * r) % q, p)
        v = ((v1 * v2) % p) % q

        if r == v:
```

```
            print('Verified OK')
        else:
            print('Verification Failure')

    elif sys.argv[1] == '-help':
        print(help_message)
    else:
        raise Exception(help_message)

    exit()
```

遇到困難與心得

這次作業蠻簡單的，基本上照著上課簡報的公式做就沒有問題了，加上 Hash Function 可以直接用 library，又加上有上次 RSA 的 function 可以直接拿來用，在程式執行的方式跟聿鎧討論，我們最後決定參考類似 `openssl` 產生金鑰和簽驗章的模式，也多接觸了一些東西。