НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Звіт з виконання комп'ютерного практикума

РЕАЛІЗАЦІЯ АЛГОРИТМІВ ГЕНЕРАЦІЇ КЛЮЧІВ ГІБРИДНИХ КРИПТОСИСТЕМ

Виконали студентки
групи ФІ-32мн
Зацаренко А. Ю.
Футурська О. В.

Перевірила:
Селюх П. В.

Київ — 2023

ЗВІТ

1.1 Мета роботи

Дослідження алгоритмів генерації псевдовипадкових послідовностей, тестування простоти чисел та генерації простих чисел з точки зору їх ефективності за часом та можливості використання для гененерації ключів асиметричних криптосистем.

1.2 Завдання на лабораторну роботу

Провести аналіз стійкості реалізацій ПВЧ та генераторів ключів для бібліотеки OpenSSL під Windows платформу. Надати опис функції генерації ПСП та ключів бібліотеки з описом алгоритму, вхідних та вихідних даних, кодів повернення. Контрольний приклад роботи з функціями.

1.3 Необхідні теоретичні відомості

Функція os.urandom() в Python використовує системний генератор випадкових чисел, доступний на більшості операційних систем, для генерації псевдовипадкової послідовності байтів.

Основний принцип генерації полягає в тому, що вона використовує непередбачувані величини (такі як температура, шуми від периферійних пристроїв тощо), які генеруються операційною системою, для створення псевдовипадкових даних.

Цей метод генерує послідовність байтів, що вважається криптографічно стійким, тобто біти у цій послідовності виглядають як випадкові та не кореляційні між собою. Вона може використовуватися для різноманітних криптографічних операцій, генерації ключів, створення токенів чи для будь-яких інших потреб, де необхідна випадковість.

Функція os.urandom(length) в Python призначена для генерації псевдовипадкової послідовності байтів заданої довжини length. Ось короткий опис цієї функції:

os.urandom(length): Ця функція з модуля os генерує псевдовипадкову послідовність байтів. Приймає аргумент length, який вказує кількість байтів, які потрібно згенерувати.

Генерація псевдовипадкових даних: Функція використовує вбудований генератор псевдовипадкових чисел, який залежить від операційної системи. Вона створює послідовність байтів, які вважаються криптографічно стійкими, що можуть бути використані для забезпечення випадковості в криптографічних операціях чи будь-яких інших випадкових потребах в програмуванні.

З використанням бібліотеки OpenSSL для Python можна використовувати cryptography для створення ключів та їх запису у файли.

1.4 Програмна реалізація

Алгоритм генерації ключів, який використовується у цій програмі, базується на алгоритмі RSA (Rivest-Shamir-Adleman), одному з найпоширеніших асиметричних криптографічних алгоритмів.

Ця програма має на меті генерацію пари ключів (приватний та публічний ключі) за допомогою бібліотеки cryptography в Python.

Ось опис функцій, використаних у програмі:

1) generate_key_pair(length): Ця функція генерує пару ключів з використанням алгоритму RSA. Параметр key_size визначає розмір ключа (за замовчуванням 2048 бітів). Функція rsa.generate_private_key() створює приватний ключ заданого розміру, а потім через private_key.public_key() отримує публічний ключ, пов'язаний з цим приватним ключем.

2) save_private_key(private_key, filename='private_key.pem'): Ця функція записує приватний ключ у файл. Вона використовує метод private_key.private_bytes() для серіалізації приватного ключа у формат PEM і запису його у файл з вказаним ім'ям.

3) save_public_key(public_key, filename='public_key.pem'): Ця функція записує публічний ключ у файл. Вона використовує метод

public_key.public_bytes() для серіалізації публічного ключа у формат PEM і запису його у файл з вказаним ім'ям.

1.5 Отримані результати

Реалізувавши основні необхідні функції, було зроблено наступне:

❄ Генерація псевдовипадкової послідовності довжини 256 байтів (2048 бітів) (рис. 1.1)

```
b'5\x8e\x8a\xbbt\xc6(z0\x0c\xf6\xb5\x8d\x8a\xa9.\xb6\x80\xcc
\xd7d\x84M\xb8\xc4|\xf2\x90\x13x\x02\xc9b\xaf\xe2\xa2iVTQ`\xca
\x98\xa3\xd0+\xdb\x8cZ)\xaa\xb5\xbaZr\xef\xb0Z&m4f\xe6/
\x93\xbf\x89\xff\xebY\xaf\x8cG\xfb\x9e\x89\xfc\xe8+HDS
\xbf4\xc2;\xe9&\x05Nm+&\xa5\x08\xf8\xf4\xd7[\x9c\x0e\x99b
\xe8\x82{\x07\x94e %g\x1a?)\x84pw\xdf\xff\x83jPqM
\xa3";u*0\x867N4\x19,,\xa0\xd0\xc0\xac3\xb7\xda\xc8\xfe
\xc9\xbd\xf1\xf5\x1b~\x94o?"\x06\xec\x90f\xcf\\\xf6\x18Yf\xad
\x91\xa1\xd3\xc47i&G\x08\xd4\x9b\xe01\xfd\xf1\xda
\x84\x89\xe4\xcb-\x17\x8a\xa0\xe5\x86QJf\xe6*J\xcau\x83r1@\x8b
\xd5\xb6\x06\x8c\xef\x9bO\xea\xa1\xc1\x17W\xf3\xe0=\xab\x12UCU
\x99Rt(5\xa1\x84\x9eN5\xefF\xbd>,y\xbc\xab\x97B\x0c
\xd8\xb4\x9b\xba\xa2\xb2\xb0`'
```

Рисунок 1.1 – Отримана послдовність

❄ Генерація ключів довжини 2048 бітів (рис. 1.2)

```
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQC66A0vFzvFqXs0
czoeSdsTWer3yRd2Z8x8kkAk9zIldgsFlFP1ZXjiWOSLis1jBvh+It6z2sBzTDKA
Xqu+BpmDfgSdCHbRnnteS1SsYUqPPr+pVBfhe1lHYoWJU1zx1MQG9IW8mAViFhCK
yAYs66fGBYthS8fiKzqHm8Nl/dAbHw4AQCC8e2fc9fVf5iY9uQ76/HCVtkDGxCka
34yH5/Wg0sZC2lFOkN4IWSjSYyOcjfRTpOpFIeWxikZwbv91VUBr+/UL3aoBwlnM
KAn++dhpxdNQ92QSeobFTlI4+wsTJPIwaHDSlGOrBDobUSdF9/6SuvTKuNAcDVmg
12zUrjSBAgMBAAECggEAK4zfaY4R1NxJQ27n51/RJqCogcT+7Q2BlbMkNBjPO/BV
4B5wM7lJogARfLxCYCUjKIzqdGDvv83M5KFqhhQWmA94drieqvgxCccmuyXOUDNQ
SF26VMU/h8UalWVxvj2T0YUPVmLbuj+m6hdybOwaAyeOkwSr1fS5phhyvqSV5YOA
dfJiwI7ZmgK59W4j5tNon3Tbg9s8wQe7koEl5vgB1dGcvejsjb6luGZ1xPS8rmbK
AkYz0IIPxvei9Ol813vlkzZuDTDQcRsVW5boC7p4ft+x2dusrvsHGsAIbEnLBXQX
M9KMN3rB68U4+Q7RHaKcf79Vp+BXTSpf5a6znzgaIwKBgQDeJ4i4sVB7MZSDyw2d
C1xc4vOG9+1htBM09OPNoZWnglSb8g/4VAv4qvSJyeMTqKHO7cnDhbo1jvgurqww
QGYgBf5cH3RhnOMvxmbdyTLMMjwrsJZj3Ecma5gHlgLASIsFIav/Hyq4Q++drMqM
hr4g9qmjZqA5B2+ue3MwRjVSCwKBgQDXYcakUh7+dXG47Yk1uB6gZJs3LEtN9vaV
LKKYLjM+fUETMnpKP49KWjrd5H2FLNrL5nxBWWM9ngZYsVl/+HYJYzMDNRqOIG5M
fLF2VNVNUutJXjfNPtj5ItToop09YlmsjL/hU2dduX3cqN6H1Ec7pKupGTn9eiEk
U4I0wy8XIwKBgDuybfgLle/0687aosbfxaskRYKwXvbRWBPz9r9MO5Xu3OOnEYmO
tETXU60vNjkTKKAdoAlNhzNbPu/0ekrhE1Bpq+d/gFu2uAofi4bvyRm8KhUnd9JP
4EK/yPPI6OZY2BrB6KE+89DeWKaLR0ap8q+S+RTyv/1zNRyzCcogI7XvAoGBAIY6
JjzD/RLiCISSbJ9w5o1ZBa6FyUyGa7geABrNOBxqnQKkw7fAEsHop6Ufv+YeRSEr
vCZR2dpNwHDH3ho1swI4s1L9ZlZK+dJrJ4Gybthoo7+umyIrV7c0MPHmXg4iW/aA
JRj9DH9JmZnKGB0Ye2g9QMn6sbw5v6vO62eMeK13AoGBAJdgYdMzln9Gev/DZOHA
nC5ZLgVeNsqDa0IB/e3DyDnsYGfmXZA24+pL6iU1NUagA1x73x/kybYnQNjw4Nrl
dHVMuqnjX+qklvXRrcSyzIG/NaCUkZK4B3fDeenGOjfMB5YCmC1oWAkGNuel/vZJ
g4Cz2g5Ul6p+2n+31nKZyiEM
-----END PRIVATE KEY-----
```

(а) private_key.pem

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAuugNLxc7xal7NHM6Hknb
E1nq98kXdmfMfJJAJPcyJXYLBZRT9WV44ljki4rNYwb4fiLes9rAc0wygF6rvgaZ
g34EnQh20Z57XktUrGFKjz6/qVQX4XtZR2KFiVNc8dTEBvSFvJgFYhYQisgGLOun
XgWLYUvH4is6h5vDZf3QGx8OAEAgvHtn3PX1X+YmPbkO+vxwlbZAxsQpGt+Mh+f1
oNLGQtpRTpDeCFko0mMjnI30U6TqRSHlsYpGcG7/dVVAa/v1C92qAcJZzCgJ/vnY
acXTUPdkEnqGxU5SOPsLEyTyMGhw0pRjqwQ6G1EnRff+krr0yrjQHA1ZoNds1K40
gQIDAQAB
-----END PUBLIC KEY-----
```

(б) public_key.pem

Рисунок 1.2 – Згенеровані ключі

❄ Вимірювання часу роботи – було отримано, що середній час для генерації пар ключів при 1000 повторах становить $0,796$ секунд.

✏ Посилання на код програми

ВИСНОВКИ

У даній лабораторній роботі було розглянуто системний генератор псевдовипадкових чисел в Python та доведено його коректність та ефективність застосування для генерації ключів асиметричних криптосистем.

Python, будучи універсальною мовою програмування, має кілька бібліотек і модулів, які взаємодіють з OpenSSL і надають криптографічні можливості. Найвідомішою бібліотекою для криптографічних операцій у Python є бібліотека cryptography.

Таким чином, OpenSSL і бібліотеки криптографії в Python пов'язані в тому сенсі, що бібліотеки криптографії в Python часто використовують OpenSSL як базову реалізацію для забезпечення криптографічної функціональності. Бібліотека cryptography, зокрема, є популярним вибором для криптографічних операцій у Python і використовує OpenSSL для певних криптографічних функцій.

Саме тому у даній роботі було застосовано бібліотеку cryptography з вбудованими вже функціями для RSA. Ці функції спрощують процес генерації та збереження ключів, дозволяючи легко створювати та зберігати ключі для подальшого використання у криптографічних операціях.