

Міністерство освіти і науки України  
Національний авіаційний університет  
Факультет кібербезпеки, комп'ютерної та програмної інженерії  
Кафедра комп'ютеризованих систем управління

Лабораторна робота № 2.2  
з дисципліни «Захист інформації в комп'ютерних системах»  
на тему «Застосування криптографічних програмних бібліотек»

Виконав:  
студент ФККПІ  
групи СП-425  
Клокун В. Д.  
Перевірила:  
Супрун О. М.

Київ 2019

## 1. МЕТА РОБОТИ

Ознайомлення з програмними бібліотеками криптографічних функцій.

## 2. ЗАВДАННЯ РОБОТИ

Використовуючи можливості криптографічних програмних бібліотек, створити програмний додаток, що виконує прості криптографічні операції.

## 3. ХІД РОБОТИ

Щоб виконати завдання лабораторної роботи, використаємо мову програмування Python і модуль `cryptography`. Перш за все, встановимо необхідний модуль. Для цього створюємо віртуальне середовище за допомогою такої команди:

```
python -m venv venv
```

В результаті з'явиться віртуальне середовище під назвою «`venv`». Тепер активуємо його за допомогою такої команди для операційної системи Windows:

```
venv\Scripts\activate.bat
```

або її аналога для GNU/Linux:

```
source venv/bin/activate
```

Тепер, знаходячись у віртуальному середовищі, встановлюємо необхідний модуль за допомогою такої команди:

```
pip install cryptography
```

Завершивши виконання, команда встановить необхідний модуль. Коли модуль встановлений, розробляємо програмний додаток, який обчислить значення хеш-суми за допомогою алгоритму SHA-256, а також зашифрує і розшифрує повідомлення за допомогою автентифікованого шифрування із додатковими даними за алгоритмом ChaCha20Poly1305 (лістинг A.2, A.1).

Запускаємо розроблений додаток, вводимо текст і спостерігаємо результат (рис. 1).

Як бачимо, розроблений програмний додаток працює справно і дає правдоподібний результат. Порівнюємо хеш-суму, отриману в розробленому додатку, із хеш-сумою, отриманою від реалізації в стандартній бібліотеці `hashlib` мови програмування Python і бачимо, що значення співпадають, отже алгоритм обчислення хеш-суми реалізований правильно.

```

C:\Windows\system32\cmd.exe
> python y04s01-infosec-lab-02-02-solution.py
Text: The quick brown fox jumps over the lazy dog
Plain text: The quick brown fox jumps over the lazy dog
SHA-256 sum: d7a8fbb307d7809469ca9abcb0082e4f8d5651e46d3cdb762d02d0bf37c9e592

Plaintext:      The quick brown fox jumps over the lazy dog
Ciphertext:     b'*\xe1\xfb0\x1e\x92\xb1h\x8a\x8fv\xca[\xe5\t\r\x85\xfd\x00\x0bm
\n\x80%\xcd\xddZZ\x07w\x1e\xd9|5\xfb\xcd&\xbe\xadj@ \x81\x02\xb6\xbb\xfe\xd2\xa8
\xa6\xd1\x86\xf1\xc4`\xbd\xb01\x10'
Decrypted text: b'The quick brown fox jumps over the lazy dog'

```

a)

```

C:\Windows\system32\cmd.exe
> python
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD6
4)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from hashlib import sha256
>>> fox = sha256()
>>> fox.update(b"The quick brown fox jumps over the lazy dog")
>>> fox.digest()
b'\xd7\xa8\xfb\xb3\x07\xd7\x80\x94i\xca\x9a\xbc\xb0\x08.0\x8dVQ\xe4m<\xdbv-\x02\
xd0\xbf7\xc9\xe5\x92'
>>> fox.hexdigest()
'd7a8fbb307d7809469ca9abcb0082e4f8d5651e46d3cdb762d02d0bf37c9e592'
>>> exit()

```

б)

Рис. 1: Результати обчислення хеш-сум, шифрування і розшифрування: а — розробленого програмного додатку, б — модуля hashlib

## 4. ВИСНОВОК

Виконуючи дану лабораторну роботу, ми знайомились з програмними бібліотеками криптографічних функцій.

### А. ПОЧАТКОВИЙ КОД ПРОГРАМНОЇ РЕАЛІЗАЦІЇ МОДУЛЯ

---

Лістинг А.1: Файл залежностей розробленого програмного додатку

---

```
1 cffi==1.13.2
2 cryptography==2.8
3 pycparser==2.19
4 six==1.13.0
```

---

---

Лістинг А.2: Початковий код програмного модуля для шифрування повідомлення

---

```
1 import os
2 from cryptography.hazmat.backends import default_backend
3 from cryptography.hazmat.primitives import hashes
4 from cryptography.hazmat.primitives.ciphers.aead import ChaCha20Poly1305
5
6
7 def run_compute_hash(text):
8     """Computes the SHA-256 sum of the given string."""
9
10    digest = hashes.Hash(hashes.SHA256(), backend=default_backend())
11    digest.update(text.encode())
12    res = digest.finalize()
13    print(
14        "Plain text:  {}\n"
15        "SHA-256 sum:  {}\n"
16        .format(
17            text,
18            res.hex(),
19        )
20    )
21
22
23 def run_aead_chacha(plaintext, associated_data=""):
24     """Encrypts and decrypts given plaintext using ChaCha20Poly1305 AEAD."""
25
26    data = plaintext.encode()
27    aad = associated_data.encode()
28    key = ChaCha20Poly1305.generate_key()
```

```

29     chacha = ChaCha20Poly1305(key)
30     nonce = os.urandom(12)
31     ciphertext = chacha.encrypt(nonce, data, aad)
32     decrypted_text = chacha.decrypt(nonce, ciphertext, aad)
33
34     print(
35         "Plaintext:      {}\n"
36         "Ciphertext:      {}\n"
37         "Decrypted text: {}".format(
38             plaintext,
39             ciphertext,
40             decrypted_text,
41         )
42     )
43
44
45
46 if __name__ == "__main__":
47     text = input("Text: ")
48     run_compute_hash(text)
49     run_aead_chacha(text)

```

---