ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

СИСТЕМИ КЕРУВАННЯ КЛЮЧАМИ ДЛЯ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ

Виконав: Новіков І.І.

Керівник: ст. викл. каф. ЕОМ Радченко В. О.

ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

2

Система керування ключами є однією з основних компонентів безпеки даних, що дозволяє зберігати, керувати та захищати ключі, які використовуються для захисту даних. СКК може використовуватися в багатьох галузях, включаючи фінанси, медицину, військову та наукову сфери.

Метою кваліфікаційної роботи є розробка та реалізація прототипу системи керування криптографічними ключами для забезпечення інформаційної безпеки, використовуючи мову програмування Python у середовищі Google Colab.

Завдання:

- проаналізувати основні принципи криптографії та управління ключами;
- дослідити сучасні криптографічні бібліотеки Python;
- розробити архітектуру системи керування ключами;
- реалізувати генерацію, зберігання, шифрування/дешифрування та знищення ключів у Google Colab;
- перевірити ефективність системи на прикладі шифрування даних.

3

ОСНОВИ УПРАВЛІННЯ КРИПТОГРАФІЧНИМИ КЛЮЧАМИ

4

СИМЕТРИЧНЕ ШИФРУВАННЯ

Симетричне шифрування - це метод шифрування, в якому використовується той самий ключ для шифрування та дешифрування даних. Цей ключ повинен бути відомий як відправнику, так і отримувачу даних.

5

АСИМЕТРИЧНЕ ШИФРУВАННЯ

Асиметричне шифрування - це метод шифрування, який використовує два ключі: публічний та приватний. Кожен користувач, що використовує асиметричне шифрування, має пару ключів: приватний ключ, який зберігається в таємниці, і публічний ключ, який розголошується.

6

ГОМОМОРФНЕ ШИФРУВАННЯ

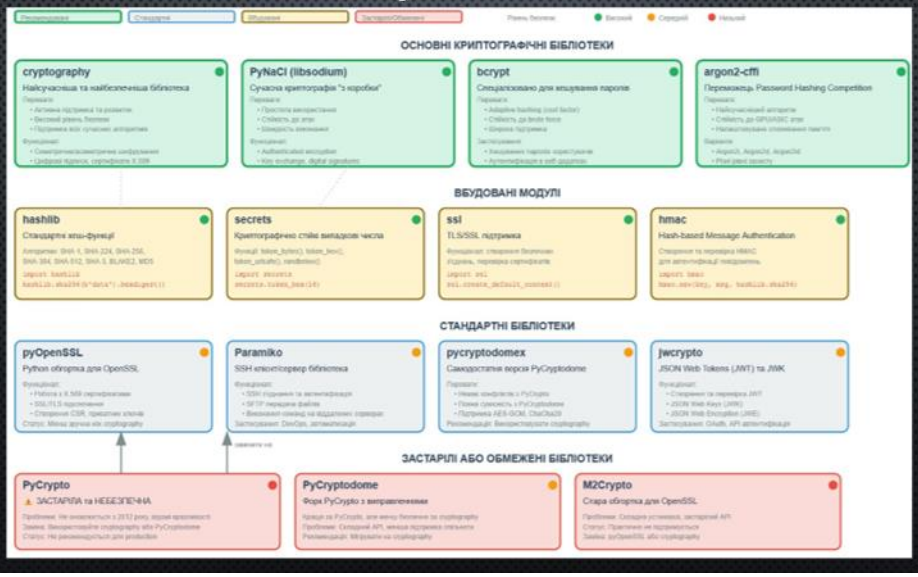
Гомоморфне шифрування - це метод шифрування, який дозволяє проводити операції з зашифрованими даними, не розшифровуючи їх. Іншими словами, гомоморфне шифрування дозволяє зашифровані дані залишати в зашифрованому вигляді, проводити з ними різні операції і отримувати результат у зашифрованому вигляді.

АНАЛІЗ ІСНУЮЧИХ СИСТЕМ КЕРУВАННЯ КЛЮЧАМИ

7

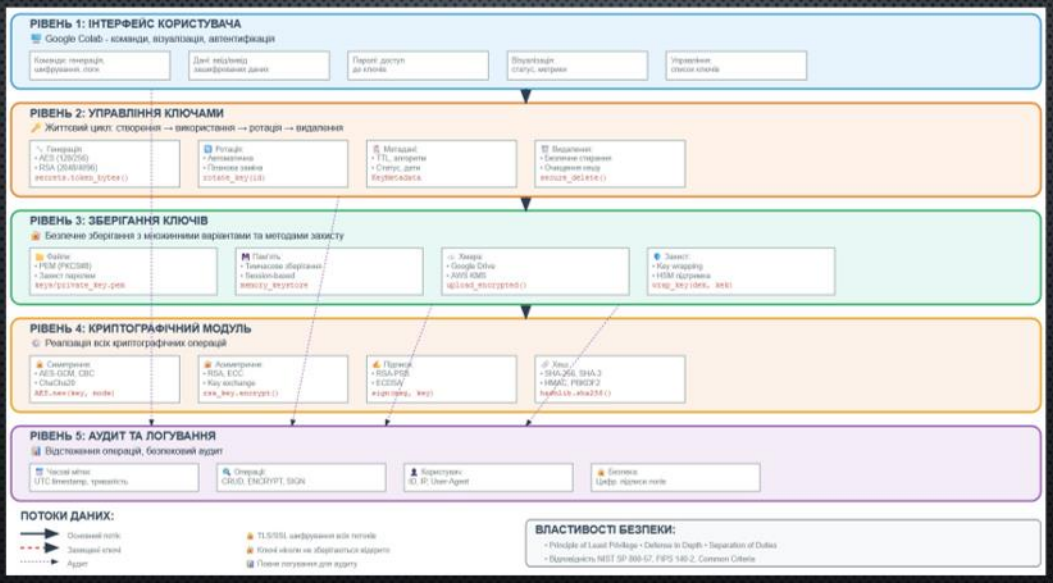
Огляд криптографічних бібліотек Python

8



Архітектура системи керування ключами

9



Реалізація системи

10

```
# Встановлення необхідної бібліотеки
!pip install cryptography
```

```
# Імпорти
from cryptography.hazmat.primitives.asymmetric import rsa, padding
from cryptography.hazmat.primitives import serialization, hashes
from cryptography.fernet import Fernet
import os
import datetime
```

```
# Створимо директорії
os.makedirs("keys", exist_ok=True)
os.makedirs("logs", exist_ok=True)
```

```
# Логування дій
def log_action(action):
    with open("logs/actions.log", "a") as log_file:
        log_file.write(f"{datetime.datetime.now()} - {action}\n")
```

```
# Генерація симетричного ключа
def generate_symmetric_key():
    key = Fernet.generate_key()
    with open("keys/symmetric.key", "wb") as f:
        f.write(key)
    log_action("Згенеровано симетричний ключ")
    print("Симетричний ключ збережено в keys/symmetric.key")
```

```
# Генерація RSA ключів
def generate_rsa_keys():
    private_key = rsa.generate_private_key(public_exponent=65537, key_size=2048)
    with open("keys/private_key.pem", "wb") as f:
        f.write(private_key.private_bytes(
            encoding=serialization.Encoding.PEM,
            format=serialization.PrivateFormat.PKCS8,
            encryption_algorithm=serialization.NoEncryption()
```

```
Симетричний ключ збережено в keys/symmetric.key
RSA ключі збережено в keys/private_key.pem та keys/public_key.pem
Зашифрований текст: b'gAAAAABoXlp2d1CVIJ3SLH6VqwodN_smXV8kKbk-dO-Z_iEgvtThkE9LInMSlx2KPcUsk7SOUfwu3GUJW81Z58_d1IKBwHZ6t3dPrpT8LC5DZdMT7xF37jXbJn-1nGsqygAGIl_4MbM0EgZoR1xp7Q_IiewaYTG7Vw=="
Розшифрований текст: Це конфіденційне повідомлення
```

Результати

11

```python
# Відображення логів
generate_symmetric_key()
generate_rsa_keys()
encrypt_with_symmetric("тест")
def show_logs():
    with open("logs/actions.log", "r") as f:
        lines = f.readlines()
    df = pd.DataFrame([line.strip().split(" - ") for line in lines], columns=["Час", "Дія"])
    return df

show_logs()
```

```
Симетричний ключ збережено в keys/symmetric.key
RSA ключі збережено в keys/private_key.pem та keys/public_key.pem
```

	Час	Дія
0	2025-06-27 08:51:45.931425	Згенеровано симетричний ключ
1	2025-06-27 08:51:46.024335	Згенеровано RSA ключі
2	2025-06-27 08:51:46.025248	Зашифровано текст симетричним ключем

```python
# Інтерактивне шифрування
def encrypt_interface():
    input_text = widgets.Text(description="Текст:")
    output_text = widgets.Textarea(description="Зашифровано:", layout={'height': '100px'})
    button = widgets.Button(description="Шифрувати")

    def on_click(b):
        encrypted = encrypt_with_symmetric(input_text.value)
        output_text.value = encrypted.decode()

    button.on_click(on_click)
    display(input_text, button, output_text)

encrypt_interface()
```

Текст:

Шифрувати

Зашифров...

ВИСНОВОК

12

У процесі виконання бакалаврської кваліфікаційної роботи розроблено прототип системи керування криптографічними ключами з використанням мови програмування Python у середовищі Google Colab. У ході роботи було проведено теоретичний аналіз фундаментальних принципів криптографії, зокрема конфіденційності, цілісності, автентифікації та незаперечності, що дало змогу визначити основні вимоги до ефективної реалізації систем управління ключами.

Під час роботи було опрацьовано та оцінено сучасні криптографічні бібліотеки Python, серед яких було обрано найдоцільніші для практичного використання в рамках розробленої системи. На основі аналізу сформовано архітектуру, що охоплює модулі генерації ключів, їх безпечного зберігання, шифрування і дешифрування даних, логування операцій та візуалізації процесів. Реалізована система забезпечує основні функції криптографічного захисту, включаючи створення як симетричних, так і асиметричних ключів, здійснення криптографічних операцій над текстовими повідомленнями, а також ведення журналу подій для забезпечення аудиту та контролю доступу.

Окрему увагу було приділено питанням зручності та наочності використання системи. З цією метою до реалізації було інтегровано інтерактивні елементи, що дозволяють виконувати шифрування безпосередньо з інтерфейсу Colab, а також візуалізацію структури системи у вигляді графічної діаграми.

У підсумку можна стверджувати, що створена система є функціональним прототипом, який може бути застосований для навчальних або дослідницьких цілей, а також адаптований до практичного використання з розширенням функціональності. Подальші напрямки вдосконалення можуть включати реалізацію повноцінної багатокористувацької автентифікації, автоматизовану ротацію ключів, інтеграцію з зовнішніми системами та розгортання на хмарних платформах.