BÀI 5: ỨNG DỤNG BẢO MẬT

Trong bài thực hành này, chúng ta sẽ tập trung vào những khái niệm cơ bản liên quan đến Base64 - một phương pháp mã hóa dùng để chuyển đổi dữ liệu nhị phân thành chuỗi văn bản, SSL (Secure Sockets Layer) - giao thức bảo mật giúp thiết lập kết nối mạng an toàn, Blockchain - công nghệ cho phép lưu trữ và xác minh thông tin một cách an toàn trong mạng phân tán, và Steganography - kỹ thuật giấu tin trong các phương tiện truyền thông như hình ảnh, video, văn bản, hoặc âm thanh mà khó bị phát hiện. Chúng ta sẽ tìm hiểu cách thức hoạt động, đặc điểm của từng phương pháp và ứng dụng của chúng trong việc bảo vệ thông tin và dữ liệu.

5.1 BASE64

Base64 là một chuẩn mã hóa cơ bản giúp biểu diễn dữ liệu nhị phân dưới dạng văn bản ASCII. Phương pháp này được sử dụng để chuyển đổi các dữ liệu nhị phân không thể đọc thành dạng văn bản có thể hiểu được và ngược lại, từ đó giúp việc truyền tải và lưu trữ thông tin trong các hệ thống không hỗ trợ dữ liệu nhị phân trở nên dễ dàng hơn.

Quá trình mã hóa Base64 chia dữ liệu nhị phân thành các nhóm 6 bit, sau đó mã hóa mỗi nhóm thành một ký tự ASCII tương ứng. Base64 thường được áp dụng trong các trường hợp như: đính kèm email, mã hóa URL và lưu trữ dữ liệu nhị phân trong các định dạng văn bản.

Mặc dù mã hóa Base64 không phải là một phương pháp bảo mật, vì nó không mã hóa dữ liệu mà chỉ chuyển đổi dữ liệu từ định dạng này sang định dạng khác, nhưng nó rất hữu ích trong việc bảo vệ dữ liệu khỏi việc mất mát trong quá trình truyền tải, cũng như trong việc lưu trữ dữ liệu nhị phân trong các hệ thống không hỗ trợ trực tiếp.

5.2 SSL

SSL (Secure Sockets Layer) là công nghệ mã hóa dữ liệu nhằm đảm bảo an toàn cho thông tin khi được truyền tải qua internet. Công nghệ này được phát triển để

cung cấp lớp bảo mật bổ sung cho các kết nối mạng, đặc biệt là trong việc bảo vệ liên kết giữa máy khách và máy chủ trên các ứng dụng web.

SSL hoạt động bằng cách mã hóa dữ liệu trong quá trình truyền qua internet, thường sử dụng mã hóa đối xứng như RSA để mã hóa dữ liệu trước khi gửi và giải mã khi dữ liệu đến đích. Quá trình này đóng vai trò quan trọng trong việc ngăn chặn các cuộc tấn công, giúp bảo vệ thông tin khỏi bị đọc hoặc can thiệp khi đang truyền tải.

SSL đã được nâng cấp thành TLS (Transport Layer Security), một phiên bản tiên tiến hơn của SSL. TLS hiện nay được áp dụng phổ biến trong ngành công nghiệp công nghệ thông tin. Khi người dùng truy cập vào một trang web có giao thức "https://" ("s" ➔ "secure"), điều này chứng tỏ rằng trang web đó đang sử dụng SSL hoặc TLS để đảm bảo an toàn cho giao tiếp giữa máy khách (client) và máy chủ (server).

Cơ chế hoạt động của SSL (và TLS) bao gồm các bước chính sau:

- **Bắt đầu phiên kết nối (Handshake):** Khi một máy khách kết nối đến một máy chủ thông qua giao thức bảo mật SSL, quá trình bắt đầu bằng một bước gọi là "handshake".
- **Xác thực (Authentication):** Máy chủ gửi chứng chỉ số công khai của mình đến máy khách để xác minh danh tính của mình. Máy khách kiểm tra chứng chỉ để đảm bảo rằng máy chủ được xác thực và có thể tin cậy.
- **Trao đổi khóa (Key Exchange):** Máy khách và máy chủ sử dụng một loạt các thuật toán để thống nhất và trao đổi thông tin về khóa mã hóa.
- **Mã hóa và Giải mã (Encryption and Decryption):** Sau khi các thông tin khóa được trao đổi, máy khách và máy chủ sử dụng thông tin này để mã hóa và giải mã dữ liệu được truyền qua kết nối.
- **Kiểm tra Tính toàn vẹn (Integrity Check):** SSL cũng cung cấp kiểm tra tính toàn vẹn dữ liệu. Mỗi gói tin được gửi đi đều được đính kèm mã xác thực để ngăn chặn việc dữ liệu bị sửa đổi khi truyền qua mạng.

Quá trình này cung cấp một kết nối bảo mật giữa máy khách và máy chủ, đảm bảo rằng dữ liệu được truyền qua internet không bị lộ thông tin và không bị thay đổi trên đường truyền.

5.3 BLOCKCHAIN

Blockchain là một công nghệ lưu trữ và truyền tải thông tin phi tập trung, được thiết kế để ghi lại các giao dịch và dữ liệu một cách an toàn, không thể thay đổi và minh bạch. Công nghệ này không chỉ được áp dụng trong lĩnh vực tài chính mà còn có thể được sử dụng trong nhiều lĩnh vực khác như y tế, chuỗi cung ứng, quản lý tài sản, bầu cử điện tử và nhiều lĩnh vực khác. Blockchain mang lại sự minh bạch và an toàn trong quá trình giao dịch cũng như lưu trữ dữ liệu. Các đặc điểm quan trọng của blockchain bao gồm:

- Phi tập trung (Decentralization): Thay vì có một trung tâm kiểm soát duy nhất, dữ liệu trên blockchain được phân tán trên nhiều nút (nodes) khác nhau trên mạng.
- Mã hoá và bảo mật: Dữ liệu trên blockchain được bảo vệ bằng mã hoá, giúp đảm bảo tính toàn vẹn và an toàn của thông tin.
- Giao dịch phi tập trung và minh bạch: Blockchain cho phép các giao dịch diễn ra một cách minh bạch và không cần đến sự can thiệp của bên thứ ba.
- Smart contracts (Hợp đồng thông minh): Blockchain cũng có khả năng thực hiện các hợp đồng thông minh tự động dựa trên các điều khoản đã được định sẵn, giúp tiết kiệm thời gian và chi phí trong việc thực hiện các giao dịch.

5.4 STEGANOGRAPHY

Steganography là một phương pháp ẩn tin và che giấu thông tin bên trong dữ liệu khác một cách không dễ dàng để phát hiện. Khác với mã hóa thông tin, Steganography không chỉ mã hóa dữ liệu mà còn ẩn nó vào các phương tiện khác một cách khá tinh vi, chẳng hạn như ẩn tin nhắn trong hình ảnh, video, file âm thanh, hoặc bất kỳ loại tệp tin nào khác mà không làm thay đổi quá nhiều nội dung gốc. Các phương pháp Steganography bao gồm:

- Ẩn tin trong hình ảnh: Đây là phương pháp phổ biến, trong đó thông tin được ẩn dưới dạng bit hoặc đổi màu sắc, không gây thay đổi rõ ràng đối với hình ảnh.
- Ẩn tin trong file âm thanh hoặc video: Tương tự như hình ảnh, dữ liệu có thể được ẩn trong các file âm thanh hoặc video bằng cách thêm thông tin không nghe thấy hoặc không nhìn thấy.

- Ẩn tin trong văn bản hoặc tập tin khác: Steganography cũng có thể được thực hiện bằng cách chèn thông tin vào các tập tin văn bản hoặc các loại tệp tin khác thông qua các kỹ thuật như việc thay đổi khoảng trắng, thêm ký tự ẩn, hoặc sử dụng kỹ thuật mã hóa dữ liệu để che giấu thông tin.

5.5 BÀI TẬP THỰC HÀNH

5.5.1 Bài thực hành 01: Base64

Viết chương trình mã hoá và giải mã thông tin sử dụng Base64.

Hướng dẫn:

- Clone Git repo về máy tính thực hành (Mở trong CMD):

```
git clone <đường dẫn git repo của sinh viên>
cd .\bmtt-nc-hutech-1511060249\
code .
```

- Tại giao diện VS Code, mở Terminal, pull mới code về:

```
git pull origin main
```

- Tách nhánh lab05 từ nhánh main.

```
git checkout -b lab05
```

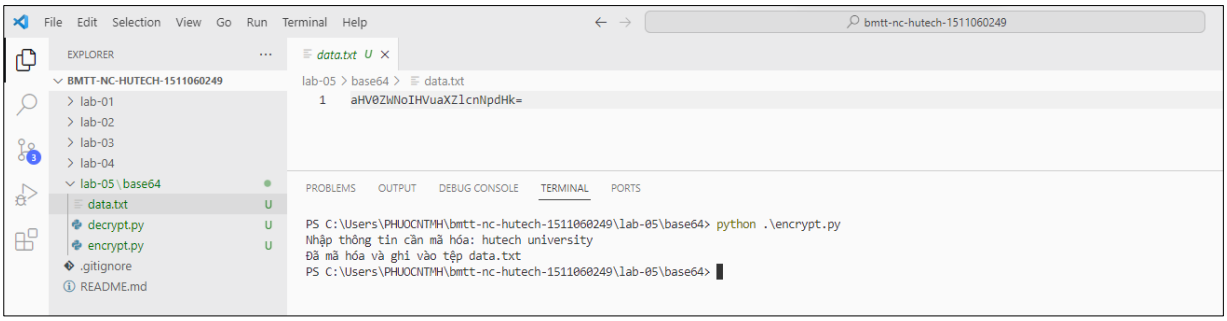
- Tạo folder "lab-05". Trong folder "lab-05" tạo folder "base64".
- Trong folder "base64" tạo file "encrypt.py".

```python
import base64

def main():
    input_string = input("Nhập thông tin cần mã hóa: ")

    encoded_bytes = base64.b64encode(input_string.encode("utf-8"))
    encoded_string = encoded_bytes.decode("utf-8")

    with open("data.txt", "w") as file:
        file.write(encoded_string)

    print("Đã mã hóa và ghi vào tệp data.txt")

if __name__ == "__main__":
    main()
```

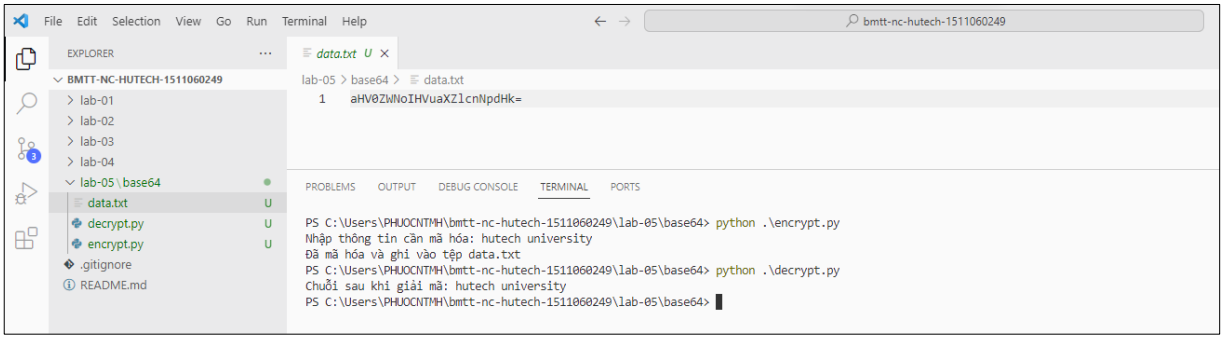
- Trong folder “base64” tạo file “decrypt.py”.

```python
import base64

def main():
    try:
        with open("data.txt", "r") as file:
            encoded_string = file.read().strip()

        decoded_bytes = base64.b64decode(encoded_string)
        decoded_string = decoded_bytes.decode("utf-8")

        print("Chuỗi sau khi giải mã:", decoded_string)
    except Exception as e:
        print("Lỗi:", e)

if __name__ == "__main__":
    main()
```

- Kiểm tra chương trình, chạy file “encrypt.py”.



- Chạy file "decrypt.py".



- Commit code:

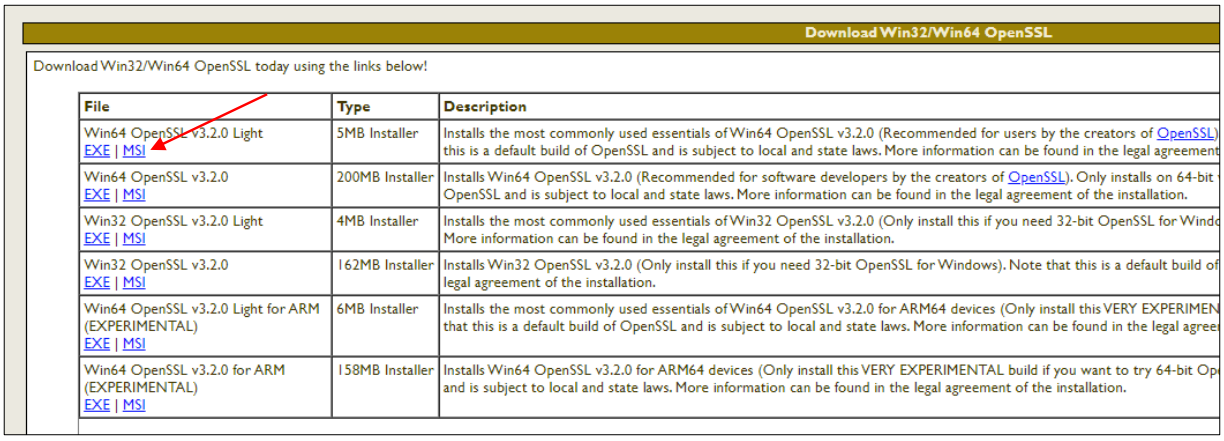
```
git add .
git commit -m "[add] lab-05 base64"
```

5.5.2 Bài thực hành 02: SSL

Viết chương trình demo cho việc kết nối và gửi dữ liệu từ client lên server thông qua SSL (được tạo từ OpenSSL).

- Tải và cài đặt OpenSSL tại liên kết sau:

"https://slproweb.com/products/Win32OpenSSL.html"

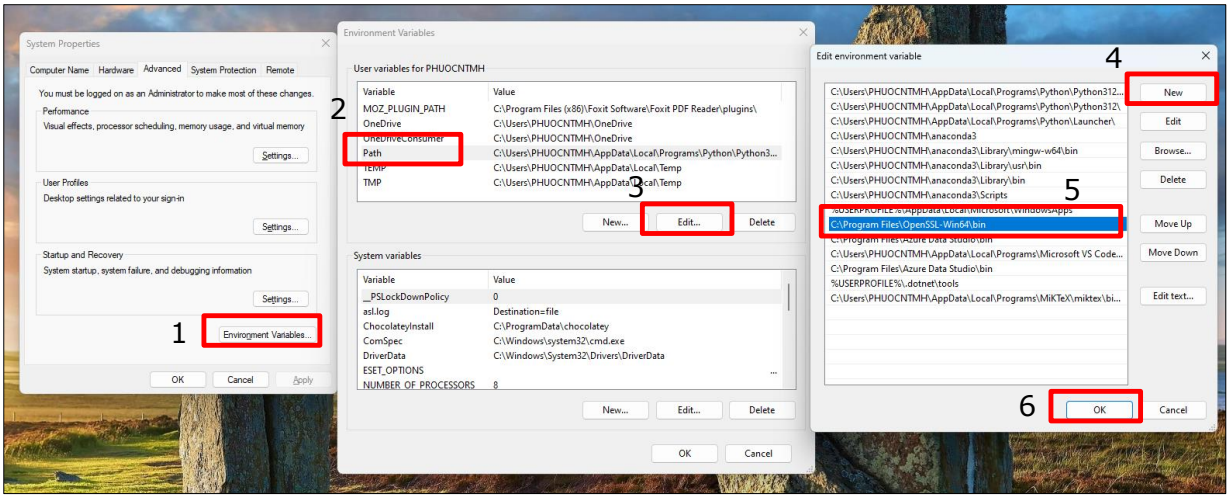


Download Win32/Win64 OpenSSL

Download Win32/Win64 OpenSSL today using the links below!

File	Type	Description
Win64 OpenSSL v3.2.0 Light EXE | MSI	5MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v3.2.0 (Recommended for users by the creators of OpenSSL) this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement
Win64 OpenSSL v3.2.0 EXE | MSI	200MB Installer	Installs Win64 OpenSSL v3.2.0 (Recommended for software developers by the creators of OpenSSL). Only installs on 64-bit OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v3.2.0 Light EXE | MSI	4MB Installer	Installs the most commonly used essentials of Win32 OpenSSL v3.2.0 (Only install this if you need 32-bit OpenSSL for Windo More information can be found in the legal agreement of the installation.
Win32 OpenSSL v3.2.0 EXE | MSI	162MB Installer	Installs Win32 OpenSSL v3.2.0 (Only install this if you need 32-bit OpenSSL for Windows). Note that this is a default build of legal agreement of the installation.
Win64 OpenSSL v3.2.0 Light for ARM (EXPERIMENTAL) EXE | MSI	6MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v3.2.0 for ARM64 devices (Only install this VERY EXPERIMEN that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agree
Win64 OpenSSL v3.2.0 for ARM (EXPERIMENTAL) EXE | MSI	158MB Installer	Installs Win64 OpenSSL v3.2.0 for ARM64 devices (Only install this VERY EXPERIMENTAL build if you want to try 64-bit Op and is subject to local and state laws. More information can be found in the legal agreement of the installation.

- Thêm OpenSSL vào Path của Windows:



- Kiểm tra OpenSSL bằng cách truy cập vào CMD và gõ lệnh:

```
Microsoft Windows [Version 10.0.22621.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PHUOCNTMH>openssl
OpenSSL>
```

- Trong folder "lab-05" tạo folder "ssl". Trong folder "ssl" tạo folder "certificates".

- Trong folder "certificates" tạo file "server-cert.cnf".

```
[req]
default_bits = 2048
prompt = no
default_md = sha256
distinguished_name = dn

[dn]
C=US
ST=State
L=Location
O=Organization
OU=Organizational Unit
CN=localhost
```

- Trong folder "certificates" tạo file "make-cert.bat".

```
openssl req -new -x509 -newkey rsa:2048 -nodes -keyout server-key.key -out server-cert.crt -days 365 -config server-cert.cnf
```

- Chạy file "make-cert.bat", ta thu được hai file "server-cert.crt" và "server-key.key"

- Trong folder "ssl" tạo file "server.py".

```python
import socket
import ssl
import threading

# Thông tin server
server_address = ('localhost', 12345)

# Danh sách các client đã kết nối
clients = []
```

```python
def handle_client(client_socket):
    # Thêm client vào danh sách
    clients.append(client_socket)

    print("Đã kết nối với:", client_socket.getpeername())

    try:
        # Nhận và gửi dữ liệu
        while True:
            data = client_socket.recv(1024)
            if not data:
                break
            print("Nhận:", data.decode('utf-8'))

            # Gửi dữ liệu đến tất cả các client khác
            for client in clients:
                if client != client_socket:
                    try:
                        client.send(data)
```

```
                except:
                    clients.remove(client)
        except:
            clients.remove(client_socket)
        finally:
            print("Đã ngắt kết nối:", client_socket.getpeername())
            clients.remove(client_socket)
            client_socket.close()

# Tạo socket server
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(server_address)
server_socket.listen(5)

print("Server đang chờ kết nối...")
```

```
# Lắng nghe các kết nối
while True:
    client_socket, client_address = server_socket.accept()

    # Tạo SSL context
    context = ssl.SSLContext(ssl.PROTOCOL_TLS)
```

```
    context.load_cert_chain(certfile="./certificates/server-cert.crt",
    keyfile="./certificates/server-key.key")

    # Thiết lập kết nối SSL
    ssl_socket = context.wrap_socket(client_socket, server_side=True)

    # Bắt đầu một luồng xử lý cho mỗi client
    client_thread = threading.Thread(target=handle_client, args=
    (ssl_socket,))
    client_thread.start()
```

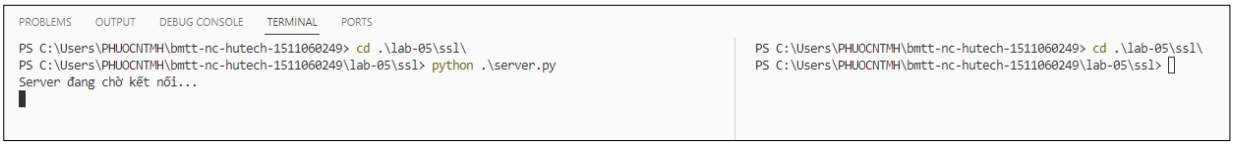
- Trong folder "ssl" tạo file "client.py".

```python
import socket
import ssl
import threading

# Thông tin server
server_address = ('localhost', 12345)

def receive_data(ssl_socket):
    try:
        while True:
            data = ssl_socket.recv(1024)
            if not data:
                break
            print("Nhận:", data.decode('utf-8'))
    except:
        pass
    finally:
        ssl_socket.close()
        print("Kết nối đã đóng.")

# Tạo socket client
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Tạo SSL context
context = ssl.SSLContext(ssl.PROTOCOL_TLS)
context.verify_mode = ssl.CERT_NONE  # Change this according to your needs
context.check_hostname = False  # Change this according to your needs

# Thiết lập kết nối SSL
ssl_socket = context.wrap_socket(client_socket, server_hostname='localhost')

ssl_socket.connect(server_address)

# Bắt đầu một luồng để nhận dữ liệu từ server
receive_thread = threading.Thread(target=receive_data, args=(ssl_socket,))
receive_thread.start()
```

```
# Gửi dữ liệu lên server
try:
    while True:
        message = input("Nhập tin nhắn: ")
        ssl_socket.send(message.encode('utf-8'))
except KeyboardInterrupt:
    pass
finally:
    ssl_socket.close()
```

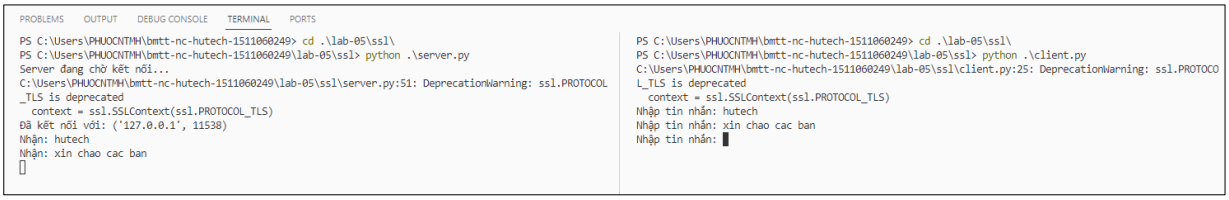
- Kiểm tra ứng dụng:
  - Bước 1: Split Terminal, chạy file "server.py".

```
python .\server.py
```



  - Bước 2: Chạy file "client.py". Nhập tin nhắn và kiểm tra.

```
python .\client.py
```



- Commit code:

```
git add .
git commit -m "[add] lab-05 ssl"
```

5.5.3 Bài thực hành 03: Blockchain

Viết chương trình mô phỏng blockchain bằng Python, giữ được các khái niệm cơ bản của blockchain, gồm các khối (blocks), giao dịch (transactions), chứng minh công việc (proof of work).

Hướng dẫn:

- Trong folder "lab-05" tạo folder "blockchain". Trong folder "blockchain" tạo file "block.py".

```python
import hashlib
import time

class Block:
    def __init__(self, index, previous_hash, timestamp, transactions, proof):
        self.index = index
        self.previous_hash = previous_hash
        self.timestamp = timestamp
        self.transactions = transactions
        self.proof = proof
        self.hash = self.calculate_hash()

    def calculate_hash(self):
        data = str(self.index) + str(self.previous_hash) + str(self.timestamp) + str(self.transactions) + str(self.proof)
        return hashlib.sha256(data.encode()).hexdigest()
```

- Trong folder "blockchain" tạo file "blockchain.py".

```python
from block import Block
import hashlib
import time

class Blockchain:
    def __init__(self):
        self.chain = []
        self.current_transactions = []
        self.create_block(proof=1, previous_hash='0')

    def create_block(self, proof, previous_hash):
        block = Block(len(self.chain) + 1, previous_hash, time.time(), self.current_transactions, proof)
        self.current_transactions = []
        self.chain.append(block)
        return block

    def get_previous_block(self):
        return self.chain[-1]

```

```
20      def proof_of_work(self, previous_proof):
21          new_proof = 1
22          check_proof = False
23          while not check_proof:
24              hash_operation = hashlib.sha256(str(new_proof**2 -
                previous_proof**2).encode()).hexdigest()
25              if hash_operation[:4] == '0000':
26                  check_proof = True
27              else:
28                  new_proof += 1
29          return new_proof
30
31      def add_transaction(self, sender, receiver, amount):
32          self.current_transactions.append({'sender': sender, 'receiver':
            receiver, 'amount': amount})
33          return self.get_previous_block().index + 1
34
```

```
35      def is_chain_valid(self, chain):
36          previous_block = chain[0]
37          block_index = 1
38          while block_index < len(chain):
39              block = chain[block_index]
40              if block.previous_hash != previous_block.hash:
41                  return False
42              previous_proof = previous_block.proof
43              proof = block.proof
44              hash_operation = hashlib.sha256(str(proof**2 -
                previous_proof**2).encode()).hexdigest()
```

```
45              if hash_operation[:4] != '0000':
46                  return False
47              previous_block = block
48              block_index += 1
49          return True
50
```

- Trong folder "blockchain" tạo file "test_blockchain.py".

```python
from blockchain import Blockchain

# Testing the blockchain
my_blockchain = Blockchain()

# Adding transactions
my_blockchain.add_transaction('Alice', 'Bob', 10)
my_blockchain.add_transaction('Bob', 'Charlie', 5)
my_blockchain.add_transaction('Charlie', 'Alice', 3)

# Mining a new block
previous_block = my_blockchain.get_previous_block()
previous_proof = previous_block.proof
new_proof = my_blockchain.proof_of_work(previous_proof)
previous_hash = previous_block.hash
my_blockchain.add_transaction('Genesis', 'Miner', 1)
new_block = my_blockchain.create_block(new_proof, previous_hash)

```

```python
# Displaying the blockchain
for block in my_blockchain.chain:
    print(f"Block #{block.index}")
    print("Timestamp:", block.timestamp)
    print("Transactions:", block.transactions)
    print("Proof:", block.proof)
    print("Previous Hash:", block.previous_hash)
    print("Hash:", block.hash)
    print("----------------------------------")

# Check if the blockchain is valid
print("Is Blockchain Valid:", my_blockchain.is_chain_valid(my_blockchain.chain))
```

- Kiểm tra chương trình: Chạy file "test_blockchain.py".

```
python .\test_blockchain.py
```

- Giải thích:
  - Kết quả xuất ra thông qua việc chạy "**test_blockchain.py**" trình bày chi tiết của hai khối trên blockchain và kết quả của hàm kiểm tra tính hợp lệ của chuỗi blockchain.

1. Block #1:
   - Index: 1 Timestamp: 1703922903.5387828
   - Transactions: Rỗng ([]), tức là không có giao dịch nào trong khối này.
   - Proof: 1 Previous Hash: Giá trị hash của khối trước đó.
   - Hash: Giá trị hash của khối này.
2. Block #2:
   - Index: 2 Timestamp: 1703922903.54079
   - Transactions: Danh sách giao dịch bao gồm 4 giao dịch, từ 'Alice' đến 'Bob' với số tiền 10, từ 'Bob' đến 'Charlie' với số tiền 5, từ 'Charlie' đến 'Alice' với số tiền 3 và một giao dịch khác từ 'Genesis' đến 'Miner' với số tiền 1.
   - Proof: 533 Previous Hash: Giá trị hash của khối trước đó.
   - Hash: Giá trị hash của khối này.

+ Cuối cùng, dòng cuối cùng trong output chứa thông điệp "Is Blockchain Valid: True" biểu thị rằng hàm kiểm tra tính hợp lệ của chuỗi blockchain đã trả về kết quả là True.

- Commit code:

```
git add .
git commit -m "[add] lab-05 blockchain"
```

5.5.4 Bài thực hành 04: Giấu tin trong ảnh

Viết chương trình Python để thực hiện việc giấu thông tin trong ảnh. Sau đó, giải mã thông điệp đã được ẩn trong ảnh sau khi hoàn thành quá trình giấu tin.

Hướng dẫn:

- Trong folder "lab-05" tạo folder "img-hidden". Trong folder "img-hidden" tạo file "requirements.txt".

```
Pillow
cryptography
```

- Vào Terminal, chạy các lệnh sau:

```
cd .\lab-05\img-hidden\
pip install -r .\requirements.txt
```

- Trong folder "img-hidden" tạo file "encrypt.py".

```
import sys
from PIL import Image

def encode_image(image_path, message):
    img = Image.open(image_path)
    width, height = img.size
    pixel_index = 0
    binary_message = ''.join(format(ord(char), '08b') for char in message)
    binary_message += '1111111111111110'  # Đánh dấu kết thúc thông điệp

```

```
    data_index = 0
    for row in range(height):
        for col in range(width):
            pixel = list(img.getpixel((col, row)))

            for color_channel in range(3):
                if data_index < len(binary_message):
                    pixel[color_channel] = int(format(pixel[color_channel],
                    '08b')[:-1] + binary_message[data_index], 2)
                    data_index += 1

            img.putpixel((col, row), tuple(pixel))

            if data_index >= len(binary_message):
                break

```

```
    encoded_image_path = 'encoded_image.png'
    img.save(encoded_image_path)
    print("Steganography complete. Encoded image saved as",
    encoded_image_path)

def main():
    if len(sys.argv) != 3:
        print("Usage: python encrypt.py <image_path> <message>")
        return

    image_path = sys.argv[1]
    message = sys.argv[2]
    encode_image(image_path, message)

if __name__ == "__main__":
    main()
```

- Trong folder “img-hidden” tạo file “decrypt.py”.

```
import sys
from PIL import Image

def decode_image(encoded_image_path):
    img = Image.open(encoded_image_path)
    width, height = img.size
    binary_message = ""

```

```
    for row in range(height):
        for col in range(width):
            pixel = img.getpixel((col, row))

            for color_channel in range(3):
                binary_message += format(pixel[color_channel], '08b')[-1]

    message = ""
    for i in range(0, len(binary_message), 8):
        char = chr(int(binary_message[i:i+8], 2))
        if char == '\0':  # Kết thúc thông điệp khi gặp dấu '\0'
            break
        message += char

    return message

def main():
    if len(sys.argv) != 2:
        print("Usage: python decrypt.py <encoded_image_path>")
        return

    encoded_image_path = sys.argv[1]
    decoded_message = decode_image(encoded_image_path)
    print("Decoded message:", decoded_message)

if __name__ == "__main__":
    main()
```

- Kiểm tra chương trình:
  - Bước 1: Chuẩn bị một ảnh, lưu vào thư mục "img-hidden" với tên "image.jpg".

- Bước 2: Thực thi file "encrypt.py". Sau khi thực thi, file "encoded_image.jpg" được tạo.

```
python .\encrypt.py .\image.jpg thongdiepcuaban
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\img-hidden> python .\encrypt.py .\image.jpg phuocnguyen
Steganography complete. Encoded image saved as encoded_image.png
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\img-hidden>
```

- Bước 3: Thực hiện giải mã thông điệp bằng cách chạy file "decrypt.py".

```
python .\decrypt.py .\encoded_image.png
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\img-hidden> python .\encrypt.py .\image.jpg phuocnguyen
Steganography complete. Encoded image saved as encoded_image.png
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\img-hidden> python .\decrypt.py .\encoded_image.png
Decoded message: phuocnguyenÿþ¶Ú©$Ûm¶
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\img-hidden>
```

- Bước 4: Thực hiện giải mã bằng hình gốc ban đầu, ta không thu được kết quả như mong muốn.

```
python .\decrypt.py .\image.jpg
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\img-hidden> python .\encrypt.py .\image.jpg phuocnguyen
Steganography complete. Encoded image saved as encoded_image.png
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\img-hidden> python .\decrypt.py .\encoded_image.png
Decoded message: phuocnguyenÿþ¶Ú©$Ûm¶
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\img-hidden> python .\decrypt.py .\image.jpg
Decoded message: J8ãØåd"d
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\img-hidden>
```

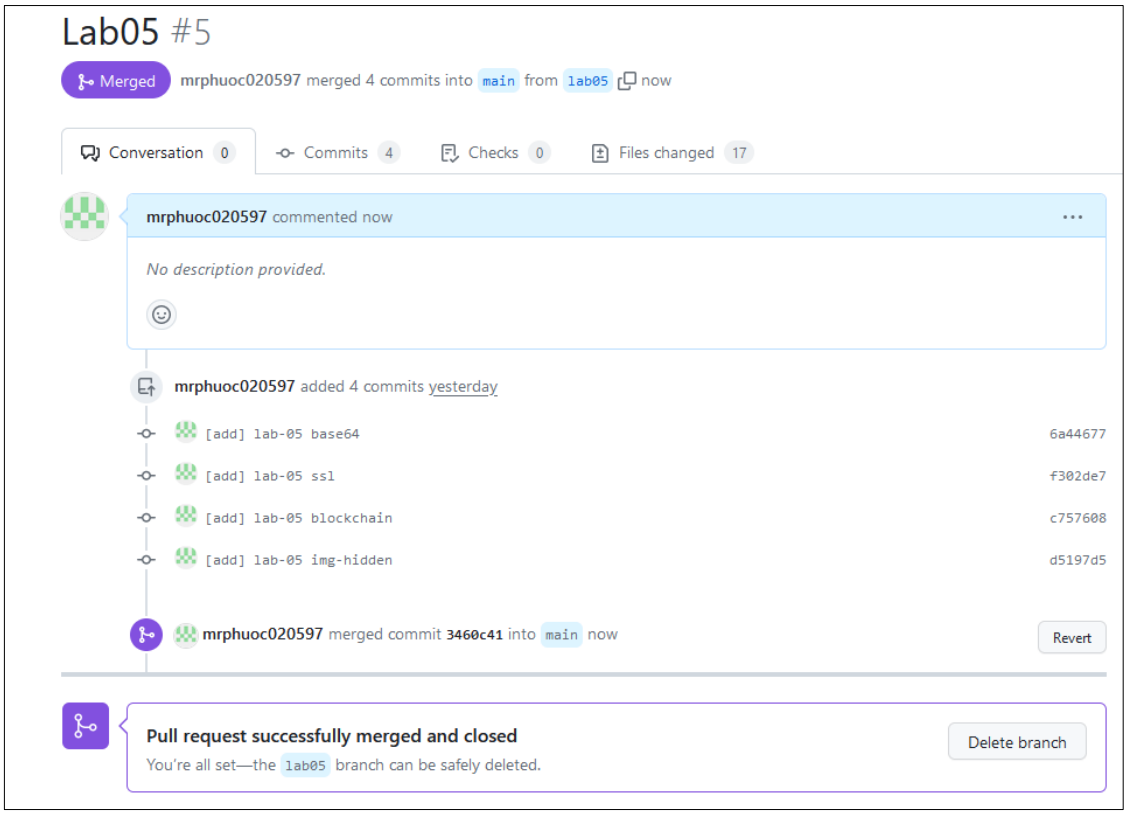
- Commit code:

```
git add .
git commit -m "[add] lab-05 img-hidden"
```

- Push các thay đổi lên remote repo:

```
git push origin lab05
```

- Tiến hành tạo PR từ nhánh lab05 về nhánh main. Review, kiểm tra đủ số lượng file, sau đó tiến hành "Merge pull request".



5.6 BÀI TẬP MỞ RỘNG

- **Câu 01:** Thực hiện tạo giao diện UI desktop cho các bài thực hành phía trên.
- **Câu 02:** Thực hiện tìm hiểu về các kỹ thuật giấu tin trong các phương tiện truyền thông khác như video, văn bản hoặc âm thanh.