

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
KHOA AN TOÀN THÔNG TIN**



**BÁO CÁO BÀI THỰC HÀNH  
HỌC PHẦN: THỰC TẬP CƠ SỞ  
MÃ HỌC PHẦN: INT13147**

**BÀI THỰC HÀNH 4.1  
LẬP TRÌNH CLIENT/SERVER ĐỂ TRAO ĐỔI THÔNG TIN AN  
TOÀN**

Sinh viên thực hiện:

B22DCAT199 Đỗ Duy Nam

Giảng viên hướng dẫn: TS.Đình Trường Duy

**HỌC KỲ 2 NĂM HỌC 2024-2025**

# MỤC LỤC

MỤC LỤC .....	2
DANH MỤC CÁC HÌNH VẼ.....	3
<b>CHƯƠNG 1. GIỚI THIỆU CHUNG VỀ BÀI THỰC HÀNH .....</b>	<b>4</b>
1.1 Mục đích.....	4
1.2 Tìm hiểu lý thuyết .....	4
1.3 Kết chương .....	6
<b>CHƯƠNG 2. NỘI DUNG THỰC HÀNH .....</b>	<b>7</b>
2.1 Chuẩn bị môi trường .....	7
2.2 Các bước thực hiện.....	7
<b>2.2.1</b> Lập trình client và Server với TCP socket .....	<b>7</b>
<b>2.2.2</b> Trao đổi thông điệp giữa client và server và đảm bảo tính toàn vẹn của thông điệp khi trao đổi.....	11
2.3 Kết chương .....	20
<b>KẾT LUẬN .....</b>	<b>21</b>
<b>TÀI LIỆU THAM KHẢO.....</b>	<b>22</b>

## DANH MỤC CÁC HÌNH VẼ

Hình 1 Lập trình client .....	7
Hình 2 Lập trình Server.....	8
Hình 3 Kết quả sau khi chạy trên Client .....	9
Hình 4 Kết quả sau khi chạy trên Server.....	9
Hình 5 Bắt thông tin được gửi từ Server đến Client .....	10
Hình 6 Bắt thông tin được gửi từ Client đến Server .....	11
Hình 7 Code của Client .....	12
Hình 8 Code của Server .....	13
Hình 9 Kết quả sau khi chạy trên Server.....	14
Hình 10 Kết quả sau khi chạy trên Client .....	14
Hình 11 Bắt được thông tin client gửi thông điệp và key được mã hóa SHA256 .....	15
Hình 12 Bắt được thông tin Server phản hồi Client sau khi nhận thông điệp và key không bị thay đổi.....	16
Hình 13 Thay đổi key ở Client.....	17
Hình 14 Bên Server, mã hash khác nhau -> Không toàn vẹn dữ liệu .....	17
Hình 15 Bên Client nhận được thông báo dữ liệu đã bị mất mát .....	18
Hình 16 Truyền thông tin từ Client tới Server, mã hash bị thay đổi.....	19
Hình 17 Server phản hồi Client: Dữ liệu đã mất mát .....	20

# CHƯƠNG 1. GIỚI THIỆU CHUNG VỀ BÀI THỰC HÀNH

## 1.1 Mục đích

Sinh viên hiểu về cơ chế client/server và có thể tự lập trình client/server dựa trên socket, sau đó thực hiện ca đặt giao thức đơn giản để trao đổi thông tin an toàn.

## 1.2 Tìm hiểu lý thuyết

Lập trình socket với TCP (Transmission Control Protocol) là một kỹ thuật quan trọng trong việc xây dựng ứng dụng mạng, cho phép hai thiết bị giao tiếp với nhau qua mạng Internet hoặc mạng nội bộ. Các khái niệm cốt lõi liên quan đến lập trình socket với TCP bao gồm:

### 1. Socket

- **Định nghĩa:** Socket là một giao diện lập trình (API) cho phép các ứng dụng gửi và nhận dữ liệu qua mạng. Nó hoạt động như một điểm cuối (endpoint) trong giao tiếp mạng.
- **Loại socket TCP:** SOCK\_STREAM – cung cấp kết nối đáng tin cậy, theo luồng, đảm bảo dữ liệu đến đúng thứ tự và không bị mất.

### 2. TCP (Transmission Control Protocol)

- **Đặc điểm:**
  - **Đáng tin cậy:** Đảm bảo dữ liệu được gửi và nhận chính xác, không bị mất hoặc trùng lặp.
  - **Kết nối hướng:** Yêu cầu thiết lập kết nối trước khi truyền dữ liệu (qua cơ chế bắt tay 3 bước - three-way handshake).
  - **Theo luồng:** Dữ liệu được gửi dưới dạng luồng byte, không có ranh giới thông điệp cố định.
- **So sánh với UDP:** TCP chậm hơn UDP nhưng đảm bảo độ tin cậy, phù hợp cho các ứng dụng như web (HTTP), email (SMTP), hoặc truyền file (FTP).

### 3. Client-Server Model

- **Server:**
  - Lắng nghe (listen) các kết nối đến từ client trên một cổng (port) cụ thể.
  - Sử dụng các hàm như bind(), listen(), accept() để thiết lập và chấp nhận kết nối.
- **Client:**
  - Kết nối đến server bằng cách chỉ định địa chỉ IP và cổng của server.
  - Sử dụng hàm connect() để thiết lập kết nối.

## 4. Các bước lập trình socket TCP

### Phía Server:

1. **Tạo socket:** Dùng hàm `socket()`.
2. **Gán địa chỉ (bind):** Gắn socket với địa chỉ IP và cổng bằng `bind()`.
3. **Lắng nghe (listen):** Chuyển socket sang chế độ chờ kết nối với `listen()`.
4. **Chấp nhận kết nối (accept):** Chấp nhận kết nối từ client bằng `accept()`, tạo socket mới để giao tiếp với client.
5. **Gửi/nhận dữ liệu:** Sử dụng `send()` hoặc `recv()` để trao đổi dữ liệu.
6. **Đóng kết nối:** Dùng `close()` để đóng socket.

### Phía Client:

1. **Tạo socket:** Dùng hàm `socket()`.
2. **Kết nối (connect):** Kết nối đến server bằng `connect()`, chỉ định IP và cổng của server.
3. **Gửi/nhận dữ liệu:** Sử dụng `send()` hoặc `recv()`.
4. **Đóng kết nối:** Dùng `close()`.

## 5. Các khái niệm quan trọng khác

- **IP Address:** Địa chỉ định danh thiết bị trên mạng (IPv4: 192.168.1.1, IPv6: 2001:0db8::1).
- **Port:** Số định danh (0-65535) để xác định ứng dụng cụ thể trên thiết bị. Ví dụ: 80 cho HTTP, 443 cho HTTPS.
- **Three-way Handshake:** Quá trình thiết lập kết nối TCP:
  1. Client gửi SYN (synchronize).
  2. Server trả lời SYN-ACK (synchronize-acknowledgment).
  3. Client gửi ACK (acknowledgment).
- **Buffer:** Bộ đệm để lưu trữ dữ liệu trước khi gửi hoặc sau khi nhận. Cần quản lý cẩn thận để tránh tràn bộ đệm.
- **Blocking vs Non-blocking:**
  - **Blocking:** Các hàm như `accept()`, `recv()` sẽ chờ đến khi có dữ liệu hoặc kết nối.
  - **Non-blocking:** Cho phép xử lý đồng thời nhiều kết nối, thường dùng với `select()`, `poll()`, hoặc các framework bất đồng bộ như `asyncio` (Python).
- **Multithreading/Asynchronous:** Để xử lý nhiều client đồng thời, server có thể dùng đa luồng (thread) hoặc lập trình bất đồng bộ.

### **1.3 Kết chương**

Ở chương này đã tìm hiểu cơ bản về các khái niệm liên quan đến lập trình socket với TCP. Thông qua đó để có thể thực hiện các yêu cầu của bài thực hành.

## CHƯƠNG 2. NỘI DUNG THỰC HÀNH

### 2.1 Chuẩn bị môi trường

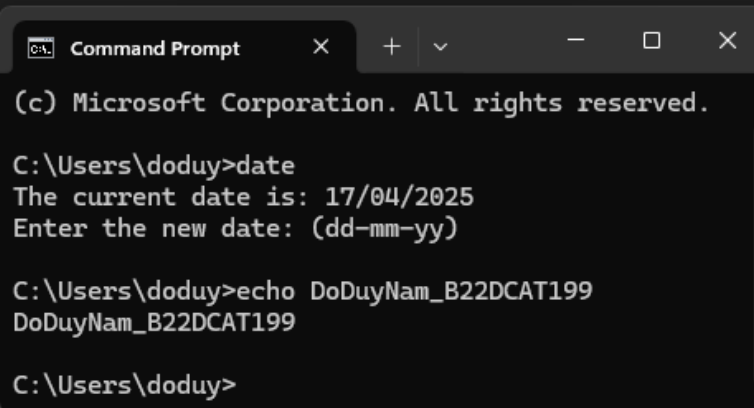
- Môi trường Python hoặc Java để chạy được ứng dụng client/server đã lập trình.
- Phần mềm Wireshark.

### 2.2 Các bước thực hiện

#### 2.2.1 Lập trình client và Server với TCP socket

- Lập trình client

```
D: > tics > Phần 4 > client.py > ...
1  import socket
2  SERVER_HOST = '127.0.0.1'
3  SERVER_PORT = 12345
4  client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5  try:
6      client_socket.connect((SERVER_HOST, SERVER_PORT))
7      print("Đã kết nối đến server!")
8      student_id = "B22DCAT199"
9      message = f"Hello, I am {student_id} client."
10     client_socket.sendall(message.encode())
11     response = client_socket.recv(1024).decode()
12     print(f"Server phản hồi: {response}")
13 except Exception as e:
14     print(f"Lỗi: {e}")
15 finally:
16     client_socket.close()
17
```



```
(c) Microsoft Corporation. All rights reserved.

C:\Users\doduy>date
The current date is: 17/04/2025
Enter the new date: (dd-mm-yy)

C:\Users\doduy>echo DoDuyNam_B22DCAT199
DoDuyNam_B22DCAT199

C:\Users\doduy>
```

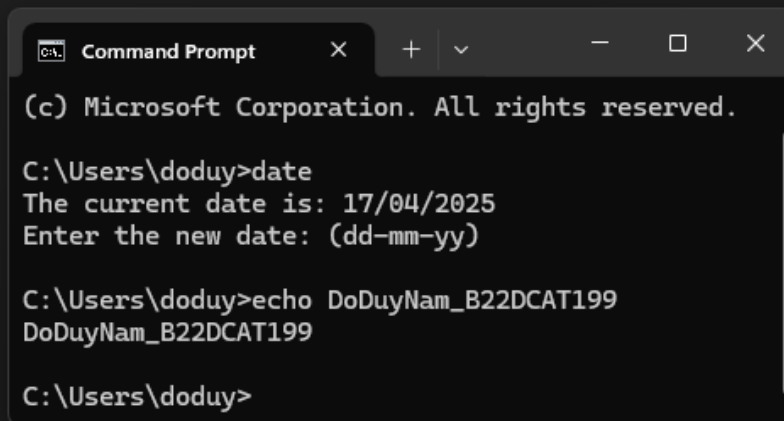
Hình 1 Lập trình client

- Lập trình server

```

1  import socket
2  HOST = '0.0.0.0'
3  PORT = 12345
4  server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5  server_socket.bind((HOST, PORT))
6  server_socket.listen()
7  print(f"Server đang lắng nghe tại {HOST}:{PORT}...")
8  while True:
9      conn, addr = server_socket.accept()
10     print(f"Kết nối từ {addr}")
11     data = conn.recv(1024).decode()
12     print(f"Nhận từ client: {data}")
13     student_id = "B22DCAT199"
14     response = f"Hello, I am {student_id} server."
15     conn.sendall(response.encode())
16     conn.close()
17

```



```

(c) Microsoft Corporation. All rights reserved.

C:\Users\doduy>date
The current date is: 17/04/2025
Enter the new date: (dd-mm-yy)

C:\Users\doduy>echo DoDuyNam_B22DCAT199
DoDuyNam_B22DCAT199

C:\Users\doduy>

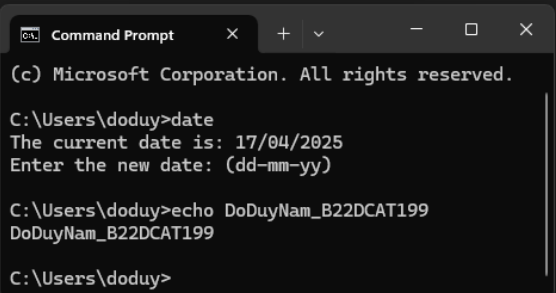
```

*Hình 2 Lập trình Server*

- Chạy server sau đó chạy client
- Client gửi thông điệp cá nhân hóa cho server: “Hello, I am B22DCAT199 client.”
- Server nhận được hiển thị thông điệp nhận được và gửi lại client thông điệp: server gửi lại “Hello, I am B22DCAT199 server”



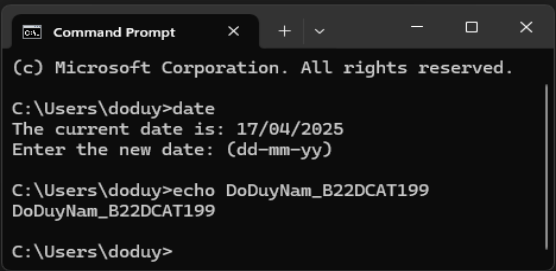
```
1 import socket
2 SERVER_HOST = '127.0.0.1'
3 SERVER_PORT = 12345
4 client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5 try:
6     client_socket.connect((SERVER_HOST, SERVER_PORT))
7     print("Đã kết nối đến server!")
8     student_id = "B22DCAT199"
9     message = f"Hello, I am {student_id} client."
10    client_socket.sendall(message.encode())
11    response = client_socket.recv(1024).decode()
12    print(f"Server phản hồi: {response}")
13 except Exception as e:
14    print(f"Lỗi: {e}")
15 finally:
16    client_socket.close()
17
```



PS D:\ttcs\Phần 4> python client.py  
Đã kết nối đến server!  
Server phản hồi: Hello, I am B22DCAT199 server.  
PS D:\ttcs\Phần 4>

Hình 3 Kết quả sau khi chạy trên Client

```
1 import socket
2 HOST = '0.0.0.0'
3 PORT = 12345
4 server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5 server_socket.bind((HOST, PORT))
6 server_socket.listen()
7 print(f"Server đang lắng nghe tại {HOST}:{PORT}...")
8 while True:
9     conn, addr = server_socket.accept()
10    print(f"Kết nối từ {addr}")
11    data = conn.recv(1024).decode()
12    print(f"Nhận từ client: {data}")
13    student_id = "B22DCAT199"
14    response = f"Hello, I am {student_id} server."
15    conn.sendall(response.encode())
16    conn.close()
17
```



D:\> ttcs > Phần 4 > server.py > ...  
PS D:\ttcs\Phần 4> python server.py  
Server đang lắng nghe tại 0.0.0.0:12345...  
Kết nối từ ('127.0.0.1', 56265)  
Nhận từ client: Hello, I am B22DCAT199 client.

Hình 4 Kết quả sau khi chạy trên Server

- Sử dụng Wireshark để bắt các thông tin đã gửi từ client đến server và ngược lại

The image displays a Wireshark packet capture window with a filter set to 'tcp.port==12345'. The packet list shows a series of TCP packets between 127.0.0.1 and 127.0.0.1. Packet 308 is selected, showing details for the Transmission Control Protocol (Seq=1, Ack=31, Win=65280, Len=30) and the Data field (30 bytes). The data field contains the text: 'o, I am B22DCAT199 serve r.'

Below the Wireshark window, a Command Prompt window is open, showing the following commands and output:

```
(c) Microsoft Corporation. All rights reserved.

C:\Users\doduy>date
The current date is: 17/04/2025
Enter the new date: (dd-mm-yy)

C:\Users\doduy>echo DoDuyNam_B22DCAT199
DoDuyNam_B22DCAT199

C:\Users\doduy>
```

Hình 5 Bắt thông tin được gửi từ Server đến Client

The image displays a Wireshark packet capture window with a filter set to 'tcp.port==12345'. The packet list shows a sequence of TCP packets between 127.0.0.1 and 127.0.0.1, including SYN, ACK, PSH, and FIN flags. The packet details pane shows the selected packet (No. 306) with its structure: Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol. The packet bytes pane shows the raw data of the packet, which includes a null terminator and the string 'Hello, I am B22DCAT199 client.'.

No.	Time	Source	Destination	Protocol	Length	Info
303	83.478297	127.0.0.1	127.0.0.1	TCP	56	55977 → 12345 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=
304	83.478355	127.0.0.1	127.0.0.1	TCP	56	12345 → 55977 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MS
305	83.478380	127.0.0.1	127.0.0.1	TCP	44	55977 → 12345 [ACK] Seq=1 Ack=1 Win=65280 Len=0
306	83.478575	127.0.0.1	127.0.0.1	TCP	74	55977 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=30
307	83.478603	127.0.0.1	127.0.0.1	TCP	44	12345 → 55977 [ACK] Seq=1 Ack=31 Win=65280 Len=0
308	83.478825	127.0.0.1	127.0.0.1	TCP	74	12345 → 55977 [PSH, ACK] Seq=1 Ack=31 Win=65280 Len=30
309	83.478843	127.0.0.1	127.0.0.1	TCP	44	55977 → 12345 [ACK] Seq=31 Ack=31 Win=65280 Len=0
310	83.478862	127.0.0.1	127.0.0.1	TCP	44	12345 → 55977 [FIN, ACK] Seq=31 Ack=31 Win=65280 Len=0
311	83.478870	127.0.0.1	127.0.0.1	TCP	44	55977 → 12345 [ACK] Seq=31 Ack=32 Win=65280 Len=0
312	83.478942	127.0.0.1	127.0.0.1	TCP	44	55977 → 12345 [FIN, ACK] Seq=31 Ack=32 Win=65280 Len=0
313	83.478964	127.0.0.1	127.0.0.1	TCP	44	12345 → 55977 [ACK] Seq=32 Ack=32 Win=65280 Len=0

Frame 306: 74 bytes on wire (592 bits), 74 bytes captured (592 b)  
 > Null/Loopback  
 > Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1  
 > Transmission Control Protocol, Src Port: 55977, Dst Port: 12345,  
 > Data (30 bytes)

```

00 00 00 45 00 00 46 91 1a 40 00 80 06 00 00  ....E..F..@.....
00 00 01 7f 00 00 01 da a9 30 39 3d ad f2 98  .........09=...
44 d8 4b 50 18 00 ff 21 9f 00 00 48 65 6c 6c  .D.KP...!...Hell
2c 20 49 20 61 6d 20 42 32 32 44 43 41 54 31  o, I am B22DCAT1
39 20 63 6c 69 65 6e 74 2e                      99 client.
  
```

Command Prompt

```

(c) Microsoft Corporation. All rights reserved.

C:\Users\doduy>date
The current date is: 17/04/2025
Enter the new date: (dd-mm-yy)

C:\Users\doduy>echo DoDuyNam_B22DCAT199
DoDuyNam_B22DCAT199

C:\Users\doduy>
  
```

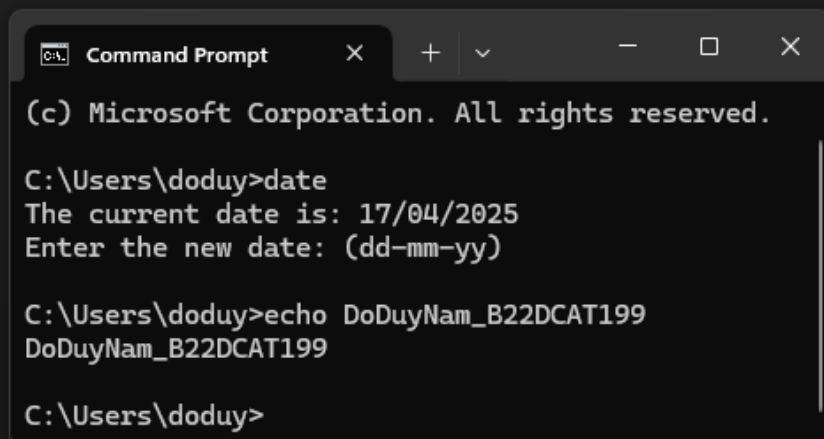
Hình 6 Bắt thông tin được gửi từ Client đến Server

## 2.2.2 Trao đổi thông điệp giữa client và server và đảm bảo tính toàn vẹn của thông điệp khi trao đổi

- Từ client và server, sửa đổi để sao cho: khi gửi thông điệp sẽ gửi kèm theo giá trị băm của (thông điệp+key) để phía bên kia kiểm tra xác minh tính toàn vẹn. Hai bên có thể thống nhất một giá trị key trước đó.

D: > tcs > Phần 4 > client.py > ...

```
1  import socket
2  import hashlib
3
4  def generate_hash(message, key):
5      return hashlib.sha256((message + key).encode()).hexdigest()
6  client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7  client.connect(('127.0.0.1', 12345))
8
9  message = "Hello, I am B22DCAT199 client."
10 key = "DoDuyNam_B22DCAT199"
11 msg_hash = generate_hash(message, key)
12 payload = f"{message}||{msg_hash}"
13 client.sendall(payload.encode())
14 response = client.recv(1024)
15 print("Server:", response.decode())
16 print("Ma_hash_Client: ",msg_hash)
17 client.close()
18
```



```
(c) Microsoft Corporation. All rights reserved.

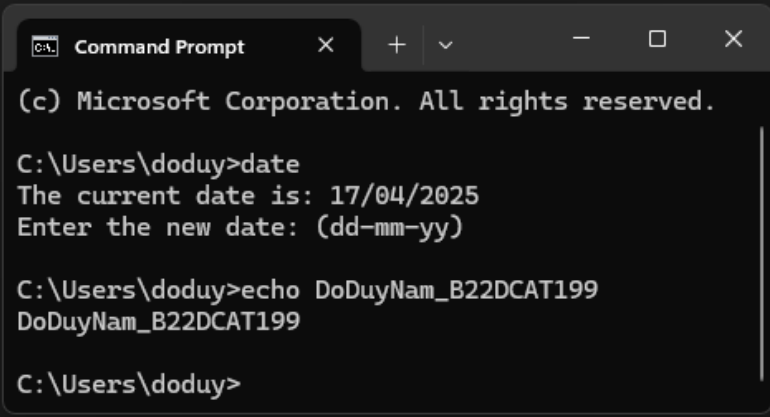
C:\Users\doduy>date
The current date is: 17/04/2025
Enter the new date: (dd-mm-yy)

C:\Users\doduy>echo DoDuyNam_B22DCAT199
DoDuyNam_B22DCAT199

C:\Users\doduy>
```

Hình 7 Code của Client

```
server.py
import socket
import hashlib
def generate_hash(message, key):
    return hashlib.sha256((message + key).encode()).hexdigest()
shared_key = "DoDuyNam_B22DCAT199"
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(('0.0.0.0', 12345))
server.listen(1)
print("Server is waiting for connection...")
conn, addr = server.accept()
print("Connected by", addr)
data = conn.recv(1024).decode()
try:
    message, received_hash = data.split("||")
except ValueError:
    conn.sendall(b"Invalid format.")
    conn.close()
    server.close()
    exit()
calculated_hash = generate_hash(message, shared_key)
print("Message OK:", message)
print("Ma_hash_Server_nhan_duoc:", received_hash)
print("Ma_hash_Server: ", calculated_hash)
response = "Hello, I am B22DCAT199 server."
if calculated_hash!=received_hash:
    response = "The received message has lost its integrity."
conn.sendall(response.encode())
conn.close()
server.close()
```



```
Command Prompt
(c) Microsoft Corporation. All rights reserved.

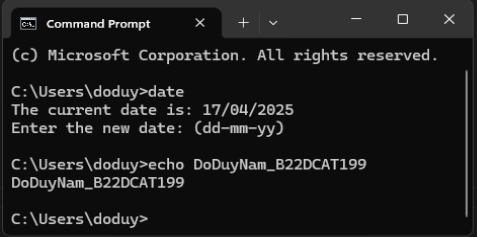
C:\Users\doduy>date
The current date is: 17/04/2025
Enter the new date: (dd-mm-yy)

C:\Users\doduy>echo DoDuyNam_B22DCAT199
DoDuyNam_B22DCAT199

C:\Users\doduy>
```

Hình 8 Code của Server

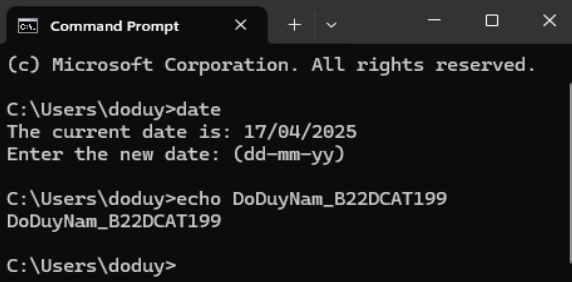
```
D: > ttcs > Phần 4 > server.py > ...
1 import socket
2 import hashlib
3 def generate_hash(message, key):
4     return hashlib.sha256((message + key).encode()).hexdigest()
5 shared_key = "DoDuyNam_B22DCAT199"
6 server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7 server.bind(('0.0.0.0', 12345))
8 server.listen(1)
9 print("Server is waiting for connection...")
10 conn, addr = server.accept()
11 print("Connected by", addr)
12 data = conn.recv(1024).decode()
13 try:
14     message, received_hash = data.split("||")
15 except ValueError:
16     conn.sendall(b"Invalid format.")
17     conn.close()
18     server.close()
19     exit()
20 calculated_hash = generate_hash(message, shared_key)
21 print("Message OK:", message)
22 print("Ma_hash_Server_nhan_duoc:", received_hash)
23 print("Ma_hash_Server: ", calculated_hash)
24 response = "Hello, I am B22DCAT199 server."
25 if calculated_hash!=received_hash:
26     response = "The received message has lost its integrity."
```



```
PS D:\ttcs\Phần 4> python server.py
Server is waiting for connection...
Connected by ('127.0.0.1', 56171)
Message OK: Hello, I am B22DCAT199 client.
Ma_hash_Server_nhan_duoc: 5c2563303468f1fdc839163e3f42e08ff2bcda8c84bf87b83257465b635868dd
Ma_hash_Server: 5c2563303468f1fdc839163e3f42e08ff2bcda8c84bf87b83257465b635868dd
PS D:\ttcs\Phần 4>
```

Hình 9 Kết quả sau khi chạy trên Server

```
D: > ttcs > Phần 4 > client.py > ...
1 import socket
2 import hashlib
3
4 def generate_hash(message, key):
5     return hashlib.sha256((message + key).encode()).hexdigest()
6 client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7 client.connect(('127.0.0.1', 12345))
8
9 message = "Hello, I am B22DCAT199 client."
10 key = "DoDuyNam_B22DCAT199"
11 msg_hash = generate_hash(message, key)
12 payload = f"{message}||{msg_hash}"
13 client.sendall(payload.encode())
14 response = client.recv(1024)
15 print("Server:", response.decode())
16 print("Ma_hash_Client: ",msg_hash)
17 client.close()
18
```



```
PS D:\ttcs\Phần 4> python client.py
Server: Hello, I am B22DCAT199 server.
Ma_hash_Client: 5c2563303468f1fdc839163e3f42e08ff2bcda8c84bf87b83257465b635868dd
PS D:\ttcs\Phần 4>
```

Hình 10 Kết quả sau khi chạy trên Client

- The figure displays two windows from Wireshark. The top window shows a packet capture filter set to "tcp.port==12345". It lists several packets, with packet 440 selected. Packet details show it's a SYN-ACK from 127.0.0.1 port 56171 to 127.0.0.1 port 12345.

No.	Time	Source	Destination	Protocol	Length	Info
437	28.554111	127.0.0.1	127.0.0.1	TCP	56	56171 → 12345 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=...
438	28.554168	127.0.0.1	127.0.0.1	TCP	56	12345 → 56171 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MS=...
439	28.554192	127.0.0.1	127.0.0.1	TCP	44	56171 → 12345 [ACK] Seq=1 Ack=1 Win=65280 Len=0
440	28.554261	127.0.0.1	127.0.0.1	TCP	140	56171 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=96
441	28.554271	127.0.0.1	127.0.0.1	TCP	44	12345 → 56171 [ACK] Seq=1 Ack=97 Win=65280 Len=0
442	28.554610	127.0.0.1	127.0.0.1	TCP	74	12345 → 56171 [PSH, ACK] Seq=1 Ack=97 Win=65280 Len=30
443	28.554625	127.0.0.1	127.0.0.1	TCP	44	56171 → 12345 [ACK] Seq=97 Ack=31 Win=65280 Len=0
444	28.554642	127.0.0.1	127.0.0.1	TCP	44	12345 → 56171 [FIN, ACK] Seq=31 Ack=97 Win=65280 Len=0
445	28.554649	127.0.0.1	127.0.0.1	TCP	44	56171 → 12345 [ACK] Seq=97 Ack=32 Win=65280 Len=0
446	28.554737	127.0.0.1	127.0.0.1	TCP	44	56171 → 12345 [FIN, ACK] Seq=97 Ack=32 Win=65280 Len=0
447	28.554771	127.0.0.1	127.0.0.1	TCP	44	12345 → 56171 [ACK] Seq=32 Ack=98 Win=65280 Len=0

The bottom window shows the details of frame 440:

  - > Frame 440: 140 bytes on wire (1120 bits), 140 bytes captured (1120 bits) on interface
  - > Null/Loopback
  - > Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
  - > Transmission Control Protocol, Src Port: 56171, Dst Port: 12345,
  - > Data (96 bytes)
  - Data: 48656c6cf2c204920616d204232324443415431393920636c69656e64  
[Length: 96]

To the right of the data field is a hex dump of the payload, which corresponds to the ASCII string "Hello, I am B22DCAT199 client. Hell...".

15

The image displays a Wireshark packet capture window with the filter 'tcp.port==12345'. The packet list shows a series of TCP packets between 127.0.0.1 and 127.0.0.1. Packet 442 is selected, showing details for the Transmission Control Protocol (Seq=1, Ack=97, Win=65280, Len=30) and the Data field (30 bytes). The data field contains a hexadecimal string: 48656c66f2c204920616d2042323244434154313939207365727665.

Overlaid on the Wireshark window is a Windows Command Prompt window. The prompt shows the following commands and output:

```
(c) Microsoft Corporation. All rights reserved.

C:\Users\doduy>date
The current date is: 17/04/2025
Enter the new date: (dd-mm-yy)

C:\Users\doduy>echo DoDuyNam_B22DCAT199
DoDuyNam_B22DCAT199

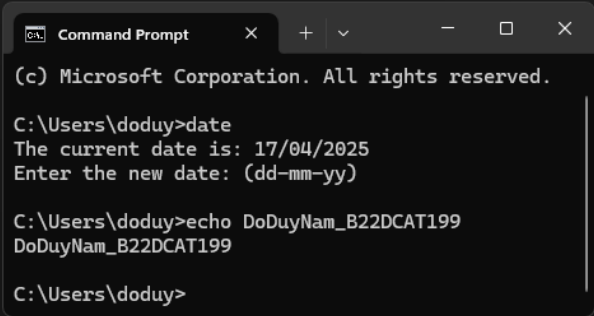
C:\Users\doduy>
```

Hình 12 Bắt được thông tin Server phản hồi Client sau khi nhận thông điệp và key không bị thay đổi

- Thay đổi giá trị key tại client và thực hiện gửi lại, nếu không đáp ứng tính toàn vẹn cần thông báo: “The received message has lost its integrity.”



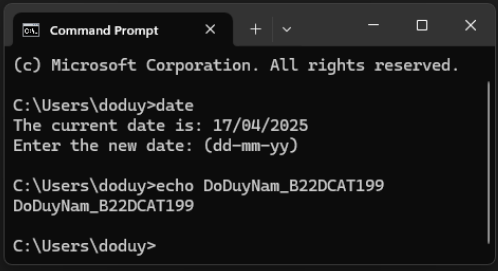
```
D: > ttcs > Phần 4 > client.py > ...
1  import socket
2  import hashlib
3
4  def generate_hash(message, key):
5      return hashlib.sha256((message + key).encode()).hexdigest()
6  client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7  client.connect(('127.0.0.1', 12345))
8
9  message = "Hello, I am B22DCAT199 client."
10 key = "DoDuyNam"
11 msg_hash = generate_hash(message, key)
12 payload = f"{message}||{msg_hash}"
13 client.sendall(payload.encode())
14 response = client.recv(1024)
15 print("Server:", response.decode())
16 print("Ma_hash_Client: ", msg_hash)
17 client.close()
18
```



Hình 13 Thay đổi key ở Client

- Sau khi đổi key chạy lại Server sau đó là chạy Client.

```
D: > ttcs > Phần 4 > server.py > ...
1  import socket
2  import hashlib
3  def generate_hash(message, key):
4      return hashlib.sha256((message + key).encode()).hexdigest()
5  shared_key = "DoDuyNam_B22DCAT199"
6  server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7  server.bind(('0.0.0.0', 12345))
8  server.listen(1)
9  print("Server is waiting for connection...")
10 conn, addr = server.accept()
11 print("Connected by", addr)
12 data = conn.recv(1024).decode()
13 try:
14     message, received_hash = data.split("||")
15 except ValueError:
16     conn.sendall(b"Invalid format.")
17     conn.close()
18     server.close()
19     exit()
20 calculated_hash = generate_hash(message, shared_key)
21 print("Message OK:", message)
22 print("Ma_hash_Server_nhan_duoc:", received_hash)
23 print("Ma_hash_Server: ", calculated_hash)
24 response = "Hello, I am B22DCAT199 server."
25 if calculated_hash != received_hash:
26     response = "The received message has lost its integrity."
```



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\ttcs\Phần 4> python server.py
Server is waiting for connection...
Connected by ('127.0.0.1', 56208)
Message OK: Hello, I am B22DCAT199 client.
Ma_hash_Server_nhan_duoc: 72e95e466e807c50015b5f3124055fae6f9d8e56f17530b8f4ae3db82001ec85
Ma_hash_Server: 5c2563303468f1fdc839163e3f42e08ff2bcd8c84bf87b83257465b635868dd
PS D:\ttcs\Phần 4>
```

Hình 14 Bên Server, mã hash khác nhau -> Không toàn vẹn dữ liệu

```
D: > ttcs > Phần 4 > client.py > ...
1  import socket
2  import hashlib
3
4  def generate_hash(message, key):
5      return hashlib.sha256((message + key).encode()).hexdigest()
6  client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7  client.connect(('127.0.0.1', 12345))
8
9  message = "Hello, I am B22DCAT199 client."
10 key = "DoDuyNam"
11 msg_hash = generate_hash(message, key)
12 payload = f"{message}||{msg_hash}"
13 client.sendall(payload.encode())
14 response = client.recv(1024)
15 print("Server:", response.decode())
16 print("Ma_hash_Client: ",msg_hash)
17 client.close()
18
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS D:\ttcs\Phần 4> python client.py
Server: The received message has lost its integrity.
Ma_hash_Client: 72e95e466e807c50015b5f3124055fae6f9d8e56f17530b8f4ae3db82001ec85
PS D:\ttcs\Phần 4> 
```

*Hình 15 Bên Client nhận được thông báo dữ liệu đã bị mất mát*

- Bắt được các bản tin trao đổi giữa client và server trong Wireshark khi key khác nhau.



The image displays a Wireshark packet capture of a TCP connection between 127.0.0.1 and 127.0.0.1. The filter is set to 'tcp.port==12345'. The packet list shows a sequence of SYN, ACK, PSH, and FIN packets. Packet 4298 is highlighted, showing 88 bytes of data. The packet details pane shows the data as a sequence of bytes, with a hex dump and ASCII representation. The ASCII part shows a corrupted message: 'received message has lost its integrity.' Below the Wireshark window, a Windows Command Prompt is open, showing the user's current date and the output of an echo command.

No.	Time	Source	Destination	Protocol	Length	Info
437	28.554111	127.0.0.1	127.0.0.1	TCP	56	56171 → 12345 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=...
438	28.554168	127.0.0.1	127.0.0.1	TCP	56	12345 → 56171 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MS...
439	28.554192	127.0.0.1	127.0.0.1	TCP	44	56171 → 12345 [ACK] Seq=1 Ack=1 Win=65280 Len=0
440	28.554261	127.0.0.1	127.0.0.1	TCP	140	56171 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=96
441	28.554271	127.0.0.1	127.0.0.1	TCP	44	12345 → 56171 [ACK] Seq=1 Ack=97 Win=65280 Len=0
442	28.554610	127.0.0.1	127.0.0.1	TCP	74	12345 → 56171 [PSH, ACK] Seq=1 Ack=97 Win=65280 Len=30
443	28.554625	127.0.0.1	127.0.0.1	TCP	44	56171 → 12345 [ACK] Seq=97 Ack=31 Win=65280 Len=0
444	28.554642	127.0.0.1	127.0.0.1	TCP	44	12345 → 56171 [FIN, ACK] Seq=31 Ack=97 Win=65280 Len=0
445	28.554649	127.0.0.1	127.0.0.1	TCP	44	56171 → 12345 [ACK] Seq=97 Ack=32 Win=65280 Len=0
446	28.554737	127.0.0.1	127.0.0.1	TCP	44	56171 → 12345 [FIN, ACK] Seq=97 Ack=32 Win=65280 Len=0
447	28.554771	127.0.0.1	127.0.0.1	TCP	44	12345 → 56171 [ACK] Seq=32 Ack=98 Win=65280 Len=0
4293	559.184804	127.0.0.1	127.0.0.1	TCP	56	56208 → 12345 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=...
4294	559.184866	127.0.0.1	127.0.0.1	TCP	56	12345 → 56208 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MS...
4295	559.184893	127.0.0.1	127.0.0.1	TCP	44	56208 → 12345 [ACK] Seq=1 Ack=1 Win=65280 Len=0
4296	559.185107	127.0.0.1	127.0.0.1	TCP	140	56208 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=96
4297	559.185127	127.0.0.1	127.0.0.1	TCP	44	12345 → 56208 [ACK] Seq=1 Ack=97 Win=65280 Len=0
4298	559.185377	127.0.0.1	127.0.0.1	TCP	88	12345 → 56208 [PSH, ACK] Seq=1 Ack=97 Win=65280 Len=44
4299	559.185394	127.0.0.1	127.0.0.1	TCP	44	56208 → 12345 [ACK] Seq=97 Ack=45 Win=65280 Len=0
4300	559.185412	127.0.0.1	127.0.0.1	TCP	44	12345 → 56208 [FIN, ACK] Seq=45 Ack=97 Win=65280 Len=0
4301	559.185420	127.0.0.1	127.0.0.1	TCP	44	56208 → 12345 [ACK] Seq=97 Ack=46 Win=65280 Len=0
4302	559.185507	127.0.0.1	127.0.0.1	TCP	44	56208 → 12345 [FIN, ACK] Seq=97 Ack=46 Win=65280 Len=0
4303	559.185540	127.0.0.1	127.0.0.1	TCP	44	12345 → 56208 [ACK] Seq=46 Ack=98 Win=65280 Len=0

```

> Frame 4298: 88 bytes on wire (704 bits), 88 bytes captured (704) on interface 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 12345, Dst Port: 56208,
  Data (44 bytes)
    Data: 546865207265636569766564206d65737361676520686173206c6f73
    [Length: 44]
    00 00 00 45 00 00 54 ca 45 40 00 80 06 00 00 .....E..T..E@.....
    00 00 01 7f 00 00 01 30 39 db 90 3d 25 1f 99 .....09...=%...
    02 6f d9 50 18 00 ff 61 0c 00 00 54 68 65 20 ..oP....a...The
    65 63 65 69 76 65 64 20 6d 65 73 73 61 67 65 received message
    68 61 73 20 6c 6f 73 74 20 69 74 73 20 69 6e has lost its in
    65 67 72 69 74 79 2e tegrity.
  
```

```

(c) Microsoft Corporation. All rights reserved.

C:\Users\doduy>date
The current date is: 17/04/2025
Enter the new date: (dd-mm-yy)

C:\Users\doduy>echo DoDuyNam_B22DCAT199
DoDuyNam_B22DCAT199

C:\Users\doduy>
  
```

Hình 17 Server phản hồi Client: Dữ liệu đã mất mát

## 2.3 Kết chương

Ở chương này đã thực hiện các yêu cầu của bài thực hành. Thực hiện thành công lập trình client, server với TCP socket và kiểm tra được tính toàn vẹn của thông điệp khi trao đổi. Và trên Wireshark bắt thành công các bản tin trao đổi giữa client và Server

## **KẾT LUẬN**

- Tìm hiểu về các khái niệm liên quan tới lập trình socket với TCP.
- Chạy thành công client và Server.
- Bắt được thành công các bản tin trao đổi giữa client và server trong Wireshark.
- Xác minh được tính toàn vẹn của thông điệp khi bị thay đổi key.

## TÀI LIỆU THAM KHẢO

- [1] Chapter 2: Application Layer V8.1 (9/2020) tại địa chỉ  
[http://gaia.cs.umass.edu/kurose\\_ross/ppt.php](http://gaia.cs.umass.edu/kurose_ross/ppt.php)