TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN 02:

Báo cáo lần 4:

End-to-End Encryption

GV hướng dẫn:

Nguyễn Văn Quang Huy

Sinh viên thực hiện:

20127192 - Trần Anh Huy

20127066 – Nguyễn Nhật Quân

Thành phố Hồ Chí Minh, ngày 28 tháng 3 năm 2024

1. TỔNG QUAN

1.1. THÔNG TIN NHÓM: NHÓM 10

MSSV	Họ tên	Email
20127192	Trần Anh Huy	20127192@student.hcmus.edu.vn
20127066	Nguyễn Nhật Quân	20127066@student.hcmus.edu.vn

1.2. CÁC CÔNG VIỆC ĐÃ HOÀN THÀNH

STT	Công việc	Tiến độ
1	Code hoàn thành sản phẩm cuối	100%
2	Triển khai các kịch bản kiểm thử	100%
3	Viết báo cáo	100%
4	Đánh giá nhận xét	100%
5	Chuẩn bị slide thuyết trình cho vấn đáp cuối kỳ	100%

2. LÝ THUYẾT

2.1. MỤC TIÊU:

2.1.1 Yêu cầu:

Xây dựng một ứng dụng nhắn tin cơ bản với các chức năng đăng nhập, xác thực trao đổi khóa an toàn và nhắn tin tích hợp mã hóa đầu cuối.

Ứng dụng nhán tin được xây dựng dựa trên mô hình Client-Server, nơi mà Server quản lý các kết nối từ nhiều Client và nhận nhiệm vụ chuyển tiếp thông điệp giữa chúng, ngoài ra Server cũng sẽ có nhiệm vụ hỗ trợ các Client trong quá trình xác thực,… Còn Client sẽ có thể nhắn tin cho các Client khác mà đảm bảo tính bí mật, toàn vẹn của tin nhắn.

2.1.2 Một số yêu cầu chính:

Mã hóa và giải mã thông điệp: Để đảm bảo tính bảo mật, các thông điệp phải được mã hóa trước khi gửi và giải mã khi nhận. Thuật toán mã hóa được sử dụng là **AES** với chế độ **CBC** để đảm bảo tính toàn vẹn và bảo mật của dữ liệu.

Xử lý kết nối đồng thời: Các kết nối này được xử lý trong các "luồng" riêng biệt, đảm bảo rằng Server có thể tiếp tục nhận và phản hồi các yêu cầu khác nhau mà không bị chậm trễ.

Giao thức truyền thông: TCP/IP được sử dụng làm giao thức truyền thông chính giữa Client và Server, đảm bảo rằng dữ liệu được truyền một cách đáng tin cậy và theo thứ tự. TCP cũng hỗ trợ tái truyền và quản lý lưu lượng, làm cho nó thích hợp cho ứng dụng yêu cầu truyền thông tin cậy như chat.

Giao thức xử lý tin nhắn: Tin nhắn được cấu trúc theo các định dạng được xác định trước, không chỉ bao gồm nội dung tin nhắn đã được mã hoá mà còn cả metadata như tên người gửi, chữ ký,…

2.2. MÔ TẢ:

2.2.1 Các chức năng chính:

- Giao tiếp trực tiếp: Người dùng có thể gửi và nhận tin nhắn văn bản với nhau qua giao diện đơn giản và thân thiện.
- Quản lý người dùng: Chương trình hỗ trợ quản lý người dùng, cho phép họ đăng nhập và tham gia trò chuyện.
- Bảo mật: Dữ liệu được mã hóa và giải mã trước khi gửi và sau khi nhận, đảm bảo tính bảo mật trong quá trình truyền thông.
- Tính linh hoạt: Chương trình hỗ trợ nhiều người dùng cùng lúc.

2.2.2 Các thành phần chính:

- Client: Đại diện cho người dùng cuối, gửi và nhận tin nhắn từ server thông qua giao diện người dùng.
- Server: Quản lý kết nối từ các clients, xử lý tin nhắn và phân phối chúng đến các clients khác nếu cần.
- Giao thức truyền thông: Sử dụng giao thức TCP/IP để đảm bảo tính ổn định và toàn vẹn trong quá trình truyền thông.

2.2.3 Công nghệ và thuật toán:

- AES (Advanced Encryption Standard): Được sử dụng để mã hóa và giải mã dữ liệu, đảm bảo tính bảo mật trong quá trình truyền thông.
- TCP (Transmission Control Protocol): Sử dụng làm giao thức truyền thông chính giữa Client và Server, đảm bảo dữ liệu được truyền một cách đáng tin cậy và theo thứ tự.
- Diffie-Hellman: Sử dụng để tạo kênh truyền khóa an toàn đảm bảo bảo mật khi chia sẻ các khóa chung.
- Thủ tục xử lý tin nhắn: Các thông điệp được cấu trúc trong một định dạng cụ thể để dễ dàng phân tích và xử lý.

2.2.4 Ưu nhược điểm:

❖ Ưu điểm:

• Đem lại nhiều quyền riêng tư hơn. Mang lại cảm giác an toàn khi cần phải truyền đạt và lưu trữ thông tin nhạy cảm (chi tiết tài chính, y tế, tài liệu kinh doanh, thủ tục pháp lý…) trên không gian mạng. Một số ứng dụng thường được các doanh nghiệp lớn dùng như : Zalo, Lark, Skype, Telegram,...

• Đối với những tổ chức kinh tế, chính trị sẽ tránh được những thiệt hại khi những thông tin mật nếu vô tình bị lộ ra ngoài, cũng khó lòng giải mã ngay lập tức.

• Có thể chủ động sử dụng nó để giảm thiểu rủi ro khi giao tiếp trực tuyến trên các mạng xã hội như Facebook, Instagram, Twitter, … Ngày nay thì các ứng dụng mạng xã hội phổ biến đều có áp dụng chế độ E2EE (Mã Hóa Đầu Cuối) hoặc cần được người dùng bật lên khi muốn sử dụng

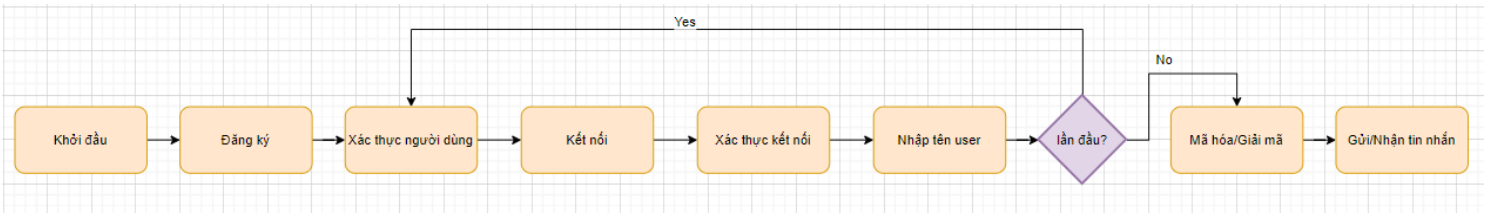
❖ Nhược điểm:

• Nếu mất khóa mã hóa, có thể sẽ mất quyền truy cập vào dữ liệu được nhận.

• Các mã khóa đơn giản không thể tránh khỏi việc bị tấn công và thử các khóa ngẫu nhiên cho đến khi khóa đúng được tìm thấy. Để giảm thiểu điều này có thể tăng chiều dài và độ phức tạp của khóa.

3. THỰC HÀNH

3.1. FLOW CHƯƠNG TRÌNH:



1. Server được khởi động trước và chờ đợi các kết nối từ client.
2. Client khởi động và kết nối đến Server. Khi Client kết nối thành công đến Server, một phiên làm việc mới được thiết lập giữa họ. Server sẽ lưu ID **duy nhất** của Client đó tạm thời trong 1 phiên làm việc của Server để phục vụ các các trường hợp Client muốn tương tác với các Client khác và cần Server làm cầu nối trung gian.
3. Nếu là lần đầu tiên kết nối, Server sẽ yêu cầu Client cung cấp tên người dùng, rồi sau đó sẽ gửi cho Client 1 yêu cầu đặc biệt để chương trình phía Client tạo cặp khoá công khai và bí mật rồi gửi lại cho Server khoá công khai để Server lưu thông tin người dùng bao gồm tên và khoá công khai vào Database của Server.
4. Ngược lại, nếu Client đã có tên trong DB thì Server sẽ gửi Challenge để cho Client giải mã nhằm xác thực người dùng. Sau khi Client đã được xác thực, họ có thể gửi và nhận tin nhắn với các Client khác thông qua Server.
5. Quá trình gửi tin nhắn đã được mã hoá bắt đầu từ **Client gửi** rồi đến Server, sau đó Server sẽ phân phối tin nhắn đó đến **Client nhận**. Cụ thể, trước khi gửi tin nhắn, **Client gửi** sẽ mã hóa tin nhắn bằng khoá chung đã được trao đổi thông qua giao thức Diffie-Hellman và dùng khoá chung đó kết hợp với thuật toán AES để mã hoá nhắn, sau đó tạo thêm chữ ký HMAC để đảm bảo tính toàn vẹn và xác thực. Server nhận tin nhắn đã mã hóa và chữ ký rồi gửi cho **Client nhận** và phía **Client nhận** sẽ thực hiện giải mã tin nhắn.
6. Server duy trì danh sách các người dùng đang kết nối và quản lý các kết nối của họ. Khi một Client muốn

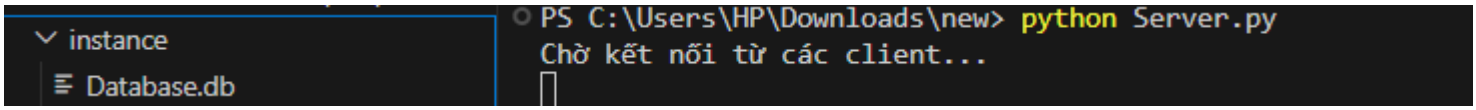
thoát khỏi trò chuyện, họ gửi một tin nhắn đặc biệt đến Server để thông báo. Server nhận tin nhắn này và đóng kết nối với Client.

3.2. HƯỚNG DẪN CHẠY CHƯƠNG TRÌNH:

Chương trình gồm 2 file, đại diện cho 2 thành phần tham gia là Server và Client.

3.2.1 Cách chạy chương trình:

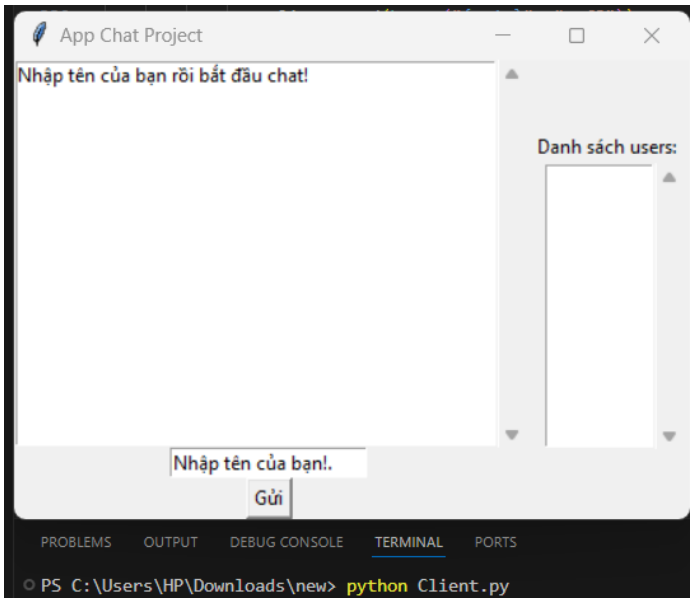
Bước 1: Chạy file **Server.py** trước thông qua lệnh **"python Server.py"**, sau đó Server sẽ kiểm tra xem Database có đang tồn tại không, nếu không thì nó sẽ khởi tạo Database mới trong thư mục **instance**. Sau đó, chương trình sẽ khởi chạy và chờ các Client kết nối vào.



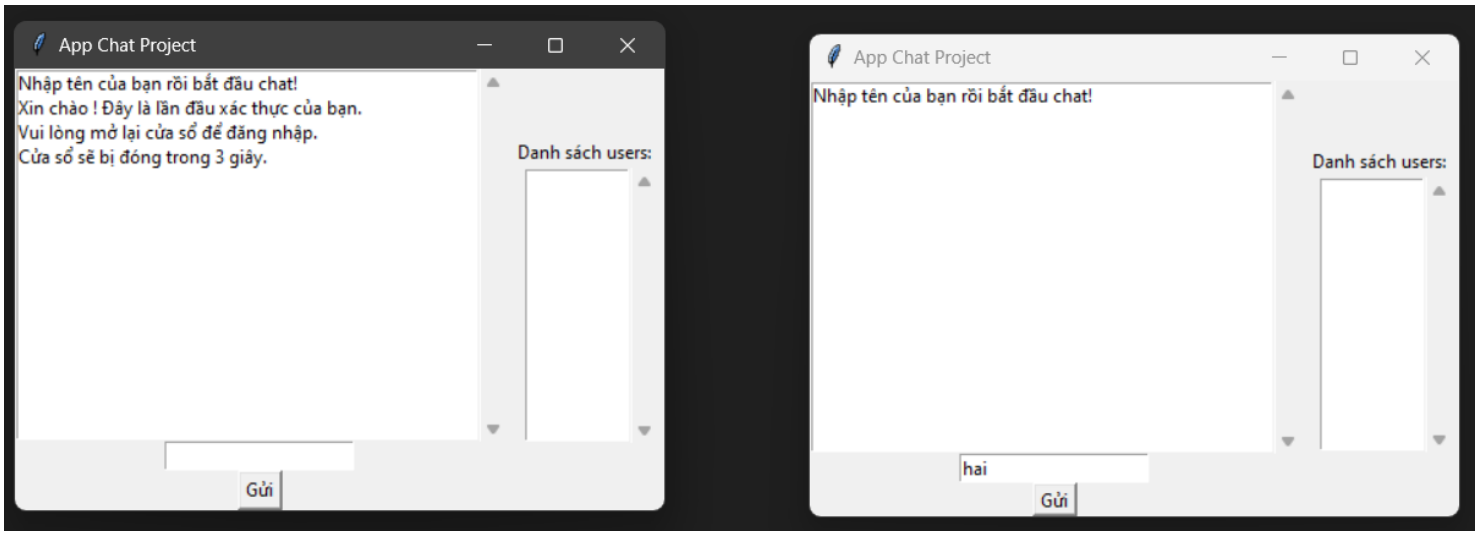
Bước 2: Chạy file **Client.py** để vào vai trò của một người dùng bằng lệnh **"python Client.py":**

+ Với mỗi file **Client.py** được chạy tương ứng với một người dùng.

+ Về giao diện sẽ gồm có thanh chat dưới cùng, gửi tin nhắn bằng cách nhập nội dung vào khung chat, sau đó nhấn enter hoặc nhấn vào nút **Gửi**. Bên trên là chat box hiển thị tin nhắn. Bên phải là danh sách các user có tên trên Database hệ thống.

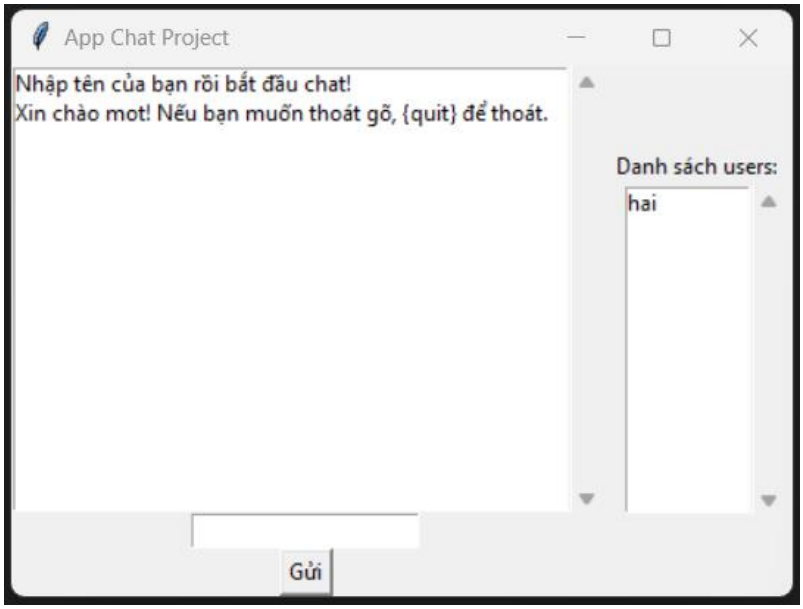


Bước 3: Khi người dùng nhập tên vào thanh chat



Tên sẽ được gửi cho Server và Server sẽ kiểm tra xem liệu có phải lần đầu người dùng sử dụng bằng cách truy vấn vào Database. Nếu đấy là lần đầu, phía Client sẽ tạo cặp khoá công khai và bí mật, sau đó khoá bí mật sẽ đc tự động lưu vào máy của người dùng, còn khoá công khai sẽ được gửi lên Server để Server thực hiện lưu trữ tên người dùng kèm khoá công khai vào Database, sau đó Server sẽ gửi 1 tín hiệu để đóng cửa sổ chat phía Client trong **3 giây**.

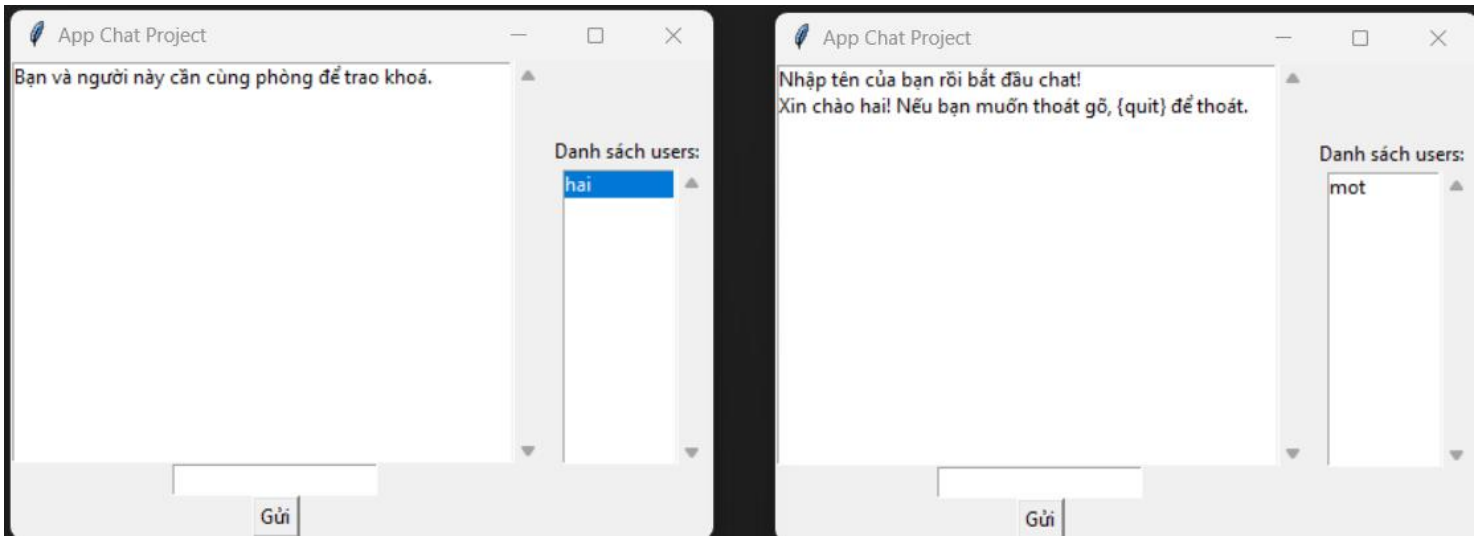
Sau khi đăng ký thành công, người dùng chạy lại lệnh **“python Client.py”** để mở cửa sổ chat và nhập tên mình vừa tạo.



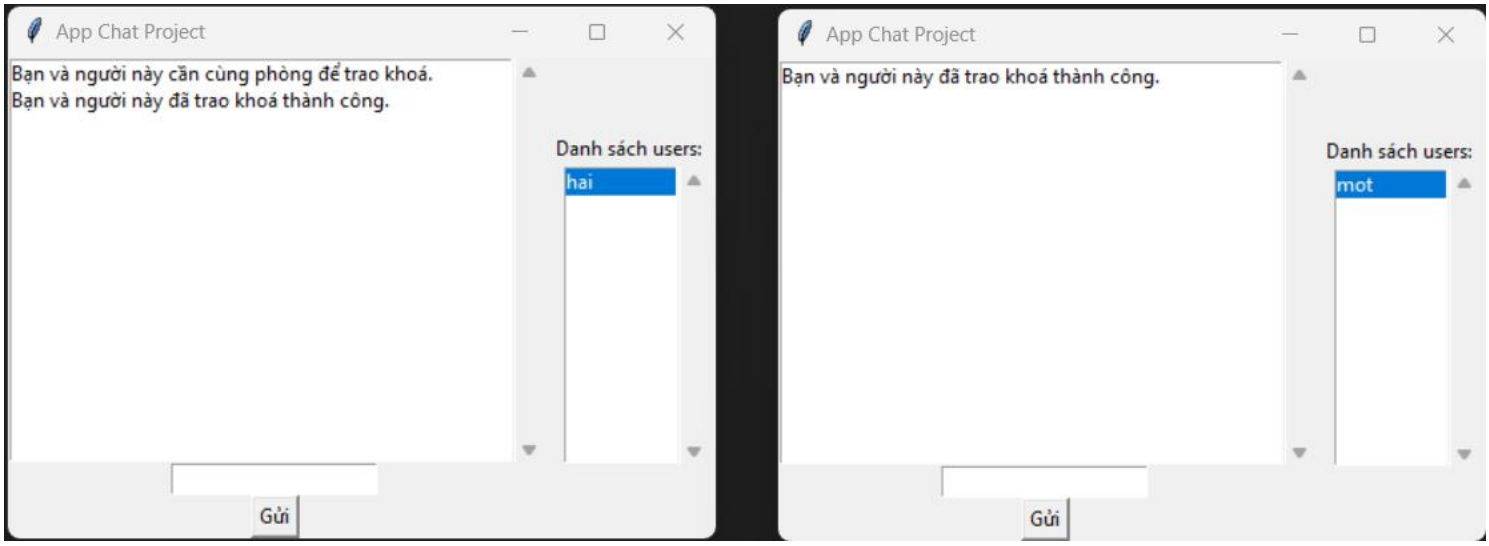
Lúc này chat box sẽ hiển thị lời chào và cột bên phải cũng sẽ hiển thị những người dùng đang có trên Database hệ thống.

Bước 5: Khi 2 người dùng đã đăng ký muốn nhắn tin cho nhau.

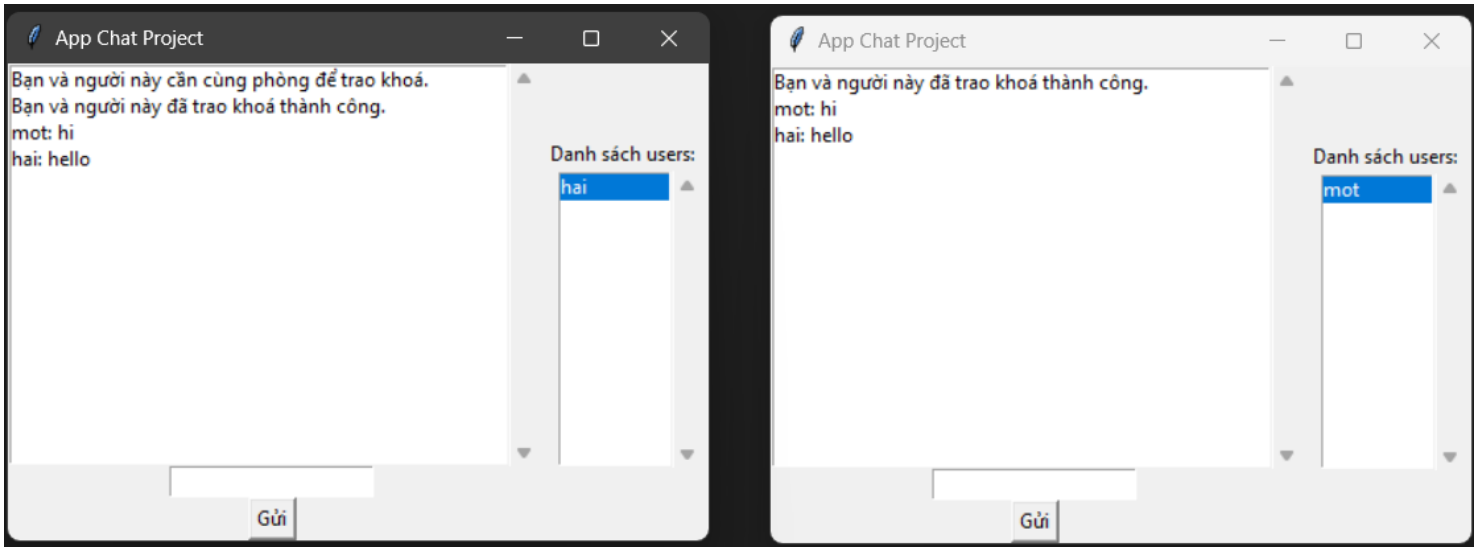
Client sẽ nhấn vào tên người dùng mà Client muốn nhắn ở cột bên phải.



Chương trình sẽ thông báo chờ người dùng thứ 2 tham gia vào phòng chat thì mới trao đổi khoá được.



Sau khi người 2 người dùng cùng tham gia vào 1 phòng thì quá trình trao đổi khóa sẽ diễn ra và khi thành công, chat box sẽ thông báo cho thấy hai người dùng đã kết nối thành công. Lúc này cả 2 đã có thể nhắn tin cho nhau. Tin nhắn sẽ được hiển thị trên chat box của cả 2 người dùng.



3.3. MỘT SỐ QUY TRÌNH XỬ LÝ:

3.3.1 Quy trình Kết nối:

Chương trình này thực hiện quy trình kết nối giữa Client và Server thông qua giao thức TCP/IP. Đầu tiên, khi client được khởi động, nó tạo một kết nối TCP/IP đến server bằng cách sử dụng giao thức **socket**, như được mô tả trong hàm **receive()** của client. Tiếp theo, Client gửi một yêu cầu kết nối đến server, đợi phản hồi từ Server và xử lý yêu cầu này trong một vòng lặp vô hạn.

Qua đó, Client và Server thiết lập một phiên làm việc, cho phép họ trao đổi dữ liệu và thông tin. Client có thể gửi và nhận dữ liệu từ Server thông qua giao thức TCP/IP, như được thể hiện trong hàm **send()** và **receive().** Mỗi khi

Client gửi dữ liệu, như tin nhắn hoặc yêu cầu, nó sẽ đợi phản hồi từ Server trước khi tiếp tục.

Khi phiên làm việc kết thúc, client gửi một yêu cầu đặc biệt đến server để thông báo về việc đó và đóng kết nối. Quá trình này được thực hiện thông qua hàm **on_closing()**. Cuối cùng, client đóng kết nối với server, hoàn thành quy trình kết nối giữa hai đầu mút.

Trong quá trình này, Client sử dụng các phương thức như mã hóa **AES** và tạo chữ ký **HMAC** để bảo mật dữ liệu trước khi gửi qua mạng, như được thể hiện trong hàm **send().** Đồng thời, Client cũng thực hiện việc xác thực thông tin và tạo mã băm **SHA-256** để xác nhận tính toàn vẹn của dữ liệu nhận được từ Server.

3.3.2 Quy trình xác thực người bằng challenge:

Sau khi kết nối thành công, server gửi cho mỗi Client một thách thức để xác thực

```
client.send(b"xThUc"+encrypted_challenge)
```

Ở mỗi Client muốn xác thực, phải nhận thách thức và xử lý nó, sau đó tạo ra một phản hồi dựa trên thách thức đó, như sau:

```
if (msg[:5] == b"xThUc"):
                msg = msg[5:]
                print("msg sau khi loại bỏ: ", msg)
                print("indexname ở xác thực: ", indexName)
                # Mở file khoá bí mật ở phía Client để giải mã challenge
                with open(f"privateKey_{indexName}.key", 'rb') as file:
                    privatekey = file.read()

                # Giải mã Challenge
                decrypted_challenge_RSA = decrypt_challenge_RSA(msg, privatekey)
                print ("Challenge giãi mã là: ", decrypted_challenge_RSA)
                #  Chuyển đổi tin nhắn thành dạng byte trước khi băm
                message_bytes = decrypted_challenge_RSA.encode('utf-8')

                # # Sử dụng SHA-256 để băm tin nhắn
                hash_object = hashlib.sha256(message_bytes)

                # # Lấy giá trị băm dưới dạng chuỗi hex
                hashed_message = hash_object.hexdigest()

                # # Gửi giá trị vừa băm cho server
                print(hashed_message)
                print(type(hashed_message))
                client_socket.send(hashed_message.encode("utf8"))
```

Cụ thể, Client sẽ nhận Challenge được mã hoá bằng PublicKey của Client được lưu trên Database hệ thống, sau đó Client sẽ lấy khoá bí mật được lưu trên máy mình để giải mã và băm challenge đã được giải mã thành chuỗi hash, kế đó gửi chuỗi hash sang cho Server.
Sau đó, Server xác minh tính hợp lệ của phản hồi từ phía Client bằng cách băm Challenge mà Server đã tạo, rồi đem so sánh chúng với chuỗi hash được gửi từ Client. Kết quả là, Client sẽ xác thực thành công nếu Client vượt

qua thử thách. Điều này đảm bảo tính an toàn và bảo mật trong quá trình giao tiếp giữa các Client và Server.

3.3.3 Quy trình xử lý tin nhắn:

Sau khi 2 Client đã trao đổi khoá với nhau, thì sẽ bắt đầu gửi tin nhắn. Cụ thể quá trình đó như sau:

1. Khi gửi tin nhắn, phía Client sẽ mã hoá tin nhắn bằng khoá chung đã được trao đổi trước đó rồi gửi tin nhắn mã hoá cho Server và Server chỉ có nhiệm vụ là chuyển tiếp đến người nhận.
2. Ở phía Client nhận, thì sẽ nhận tin nhắn mã hoá do Server chuyển tiếp, rồi dùng khoá chung đã được trao đổi trước đó giải mã tin nhắn.

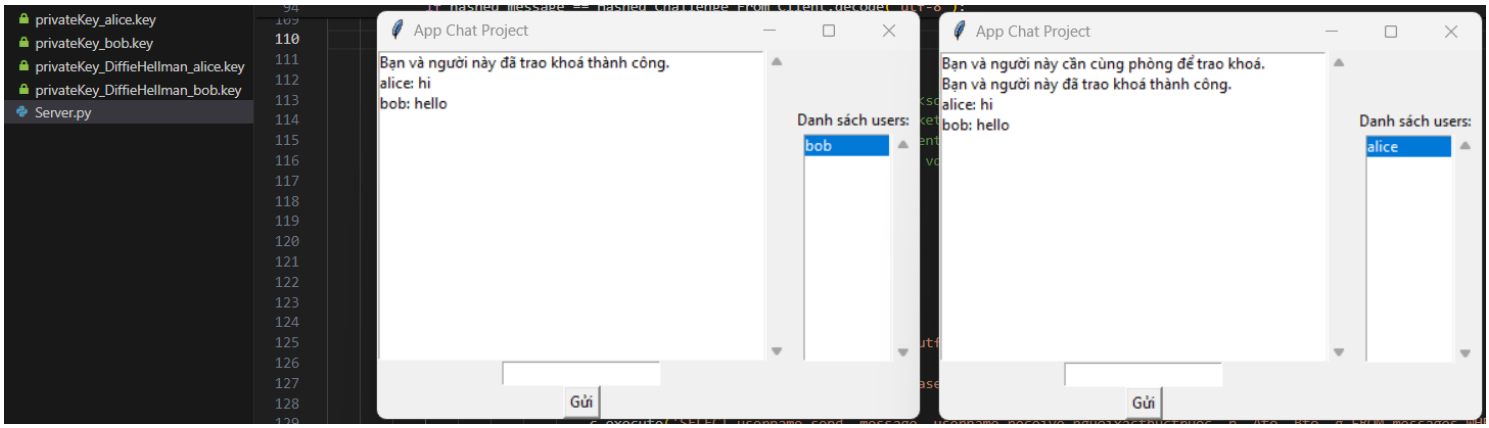
Cả quá trình xử lý tin nhắn thì Server không có can thiệp bất cứ hành động nào vào tin nhắn của 2 Client ngoài chuyển tiếp nó.

3.3. KỊCH BẢN KIỂM THỬ:

Để kiểm thử, ta giả định 3 người dùng có tên lần lượt là: **Alice**, **Bob** và **Eve**. Người dùng sẽ có 2 trạng thái là online và offline.

3.3.1 Trường hợp 1: Alice và Bob chưa trao đổi khóa và nhắn cho nhau (cả 2 đều online)

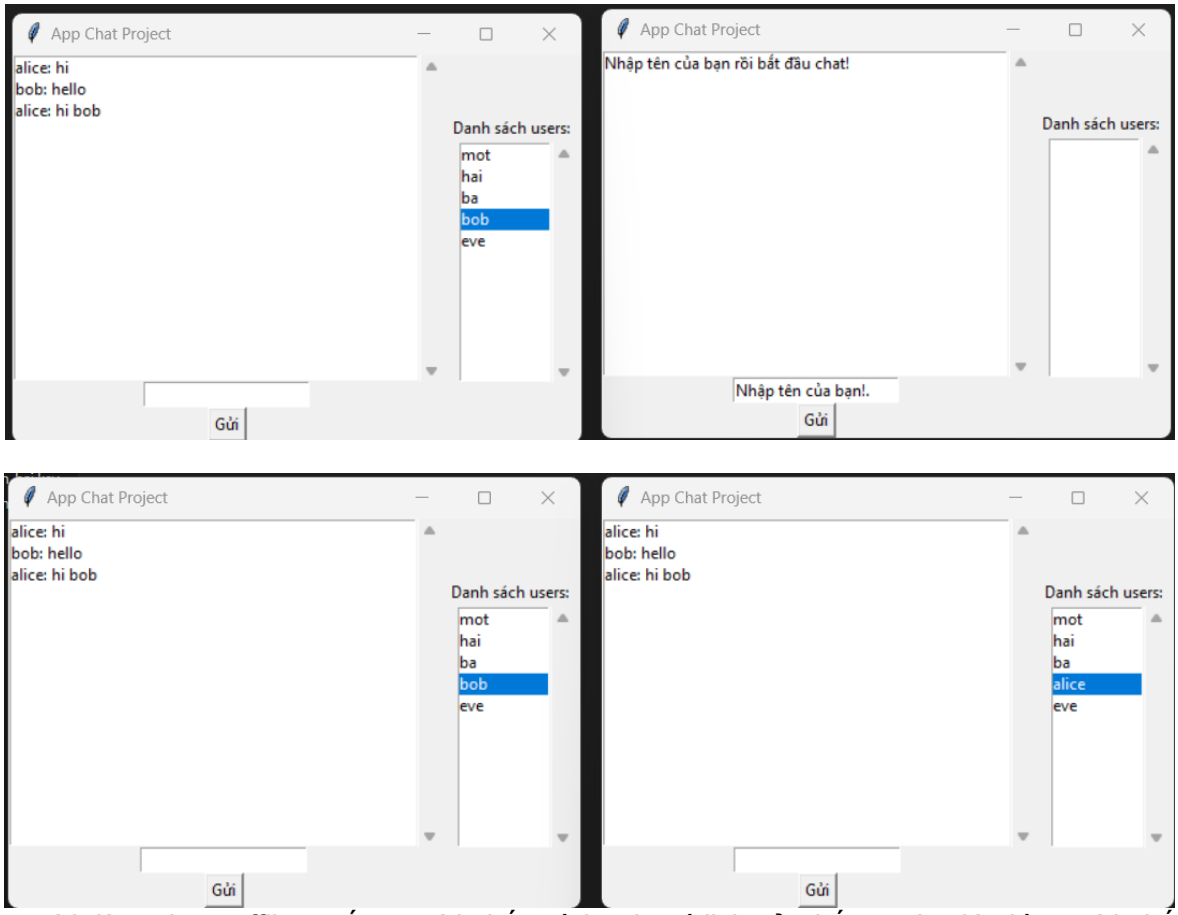
Đầu tiên, **Bob** chọn tên **Alice** ở cột bên phải, sẽ hiển thị thông báo **"Bạn và người này cần cùng phòng để trao đổi khóa"**.
Sau khi **Alice** chọn tên **Bob**, chatbox sẽ hiển thị **"Bạn và người này đã trao đổi khóa thành công"** và nhắn tin cho nhau



=> Nếu 2 người dùng đã trao đổi khóa cho nhau và online có thể nhắn tin trực tiếp cho nhau

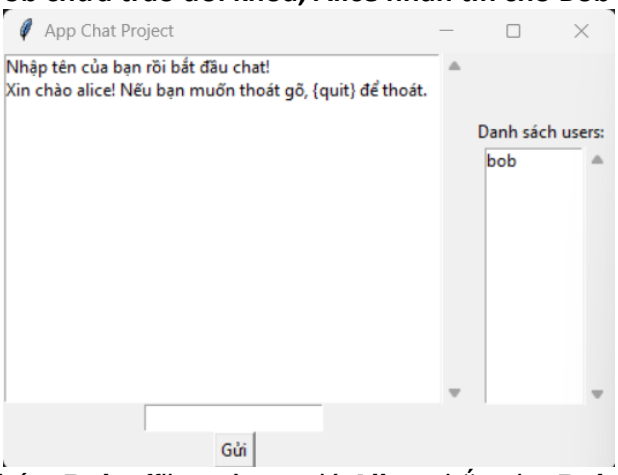
3.3.2 Trường hợp 2: Alice và Bob đã có lịch sử nhắn tin trước đó và Alice nhắn tin cho Bob khi Bob đang offline

Alice và **Bob** đã trao đổi khóa và đã nhắn tin cho nhau trước đó. Sau đó chỉ **Alice** online, **Bob** offline, lúc này **Alice** thử nhắn cho **Bob** 1 câu **"hi Bob",** ta sẽ thu được kết quả như sau:

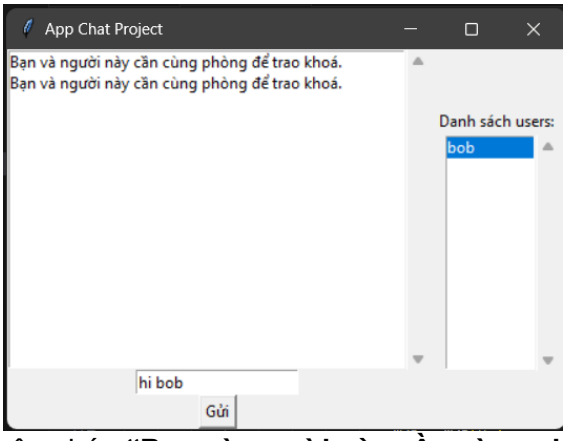


=> Khi một người dùng đang offline, nếu người nhắn và họ đã có lịch sử nhắn trước đó, thì người nhắn vẫn có thể nhắn được, và sau khi người nhận online trở lại thì các tin nhắn chưa đọc vẫn sẽ hiển thị lên đầy đủ.

3.3.3 Trường hợp 3: Alice và Bob chưa trao đổi khóa, Alice nhắn tin cho Bob khi Bob đang offline

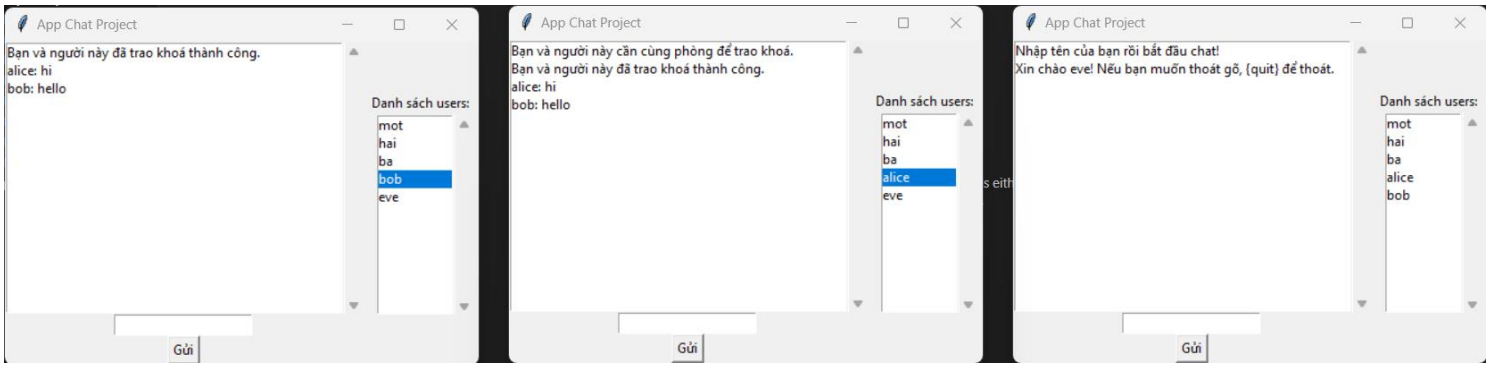


Khi **Alice** và **Bob** chưa trao đổi khóa, **Bob** offline và sau đó **Alice** nhắn cho **Bob**, ta thu được kết quả như sau:



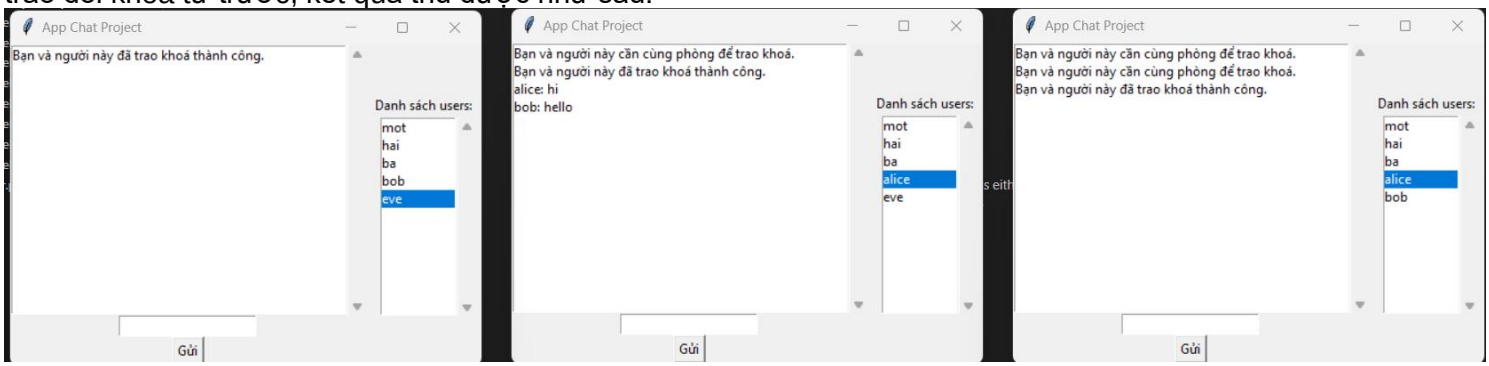
Cửa sổ chat của **Alice** sẽ hiển thị thông báo **"Bạn và người này cần cùng phòng để trao đổi khóa"**
=> Khi chưa trao đổi khóa, người nhắn không thể nhắn cho người nhận dù bên kia đang online hay offline.

3.3.4 Trường hợp 4: Alice và Bob đã trao đổi khóa, Eve và Alice chưa trao đổi khóa và nhắn cho nhau



ALICE **BOB** **EVE**

Alice và **Bob** đã trao đổi khóa và đã nhắn tin cho nhau, lúc này **Eve** thực hiện chọn tên **Alice** để nhắn dù chưa trao đổi khóa từ trước, kết quả thu được như sau:



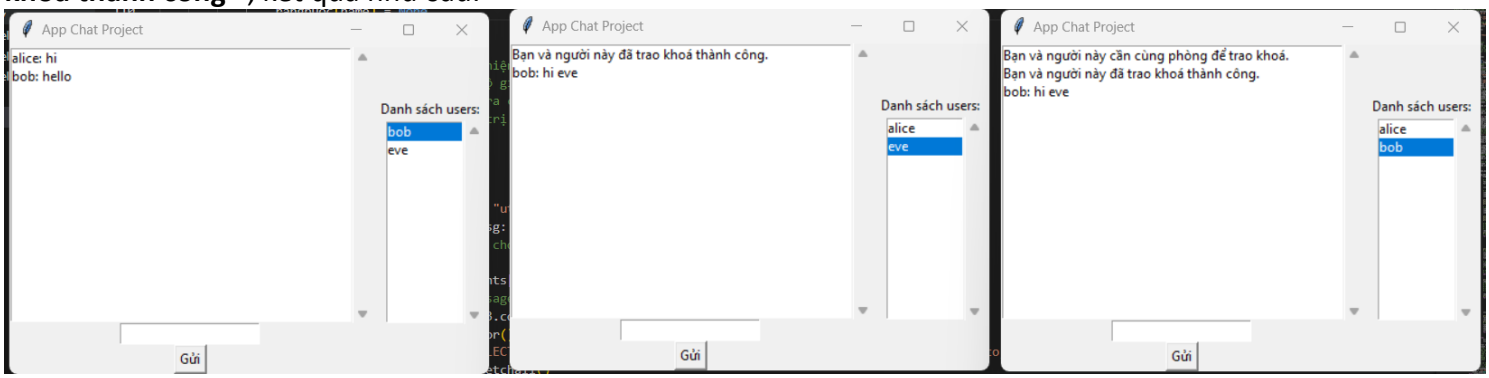
ALICE **BOB** **EVE**

Chương trình sẽ hiển thị thông báo bên **Eve** là **"Bạn và người này cần cùng phòng để trao đổi khóa"** và khi **Alice** chọn vào tên **Eve**, lúc này sẽ hiển thị thông báo "**Bạn và người này đã trao đổi khóa thành công**" và lúc này **Alice** và **Eve** có thể nhắn cho nhau.
=> Nếu người thứ 3 chưa trao đổi khóa sẽ không thể nhắn tin cho đối phương được và sau khi trao đổi khóa, các cặp khóa hoàn toàn không bị xáo trộn hay nhầm lẫn lẫn nhau.

3.3.5 Trường hợp 5: Alice và Bob đã trao đổi khóa, Alice và Eve đã trao đổi khóa, nhưng Bob và Eve chưa

trao đổi khóa và nhắn cho nhau

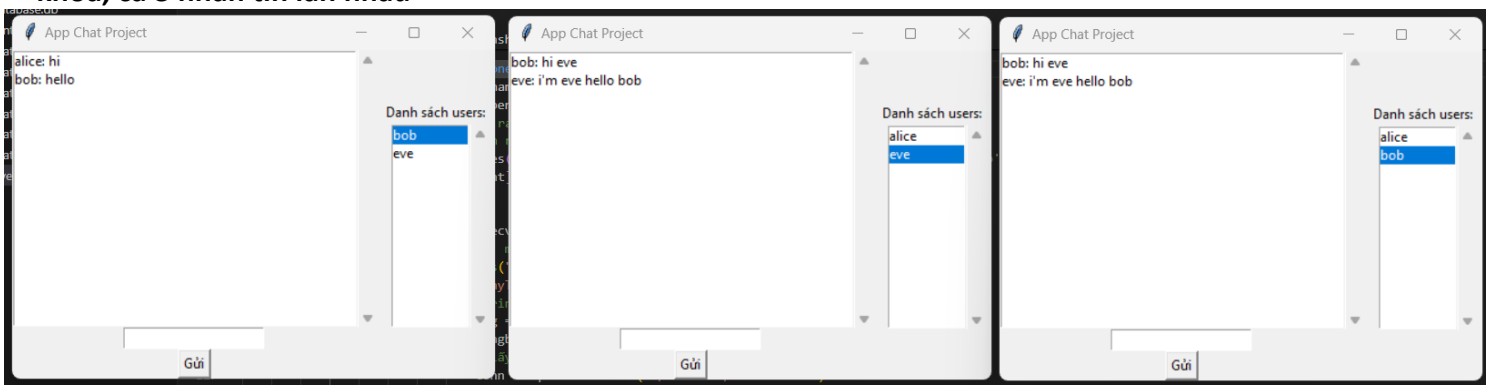
Cặp **Alice** và **Bob** và cặp **Alice** và **Eve** đã trao đổi khóa, khi **Bob** nhắn cho **Eve**, sẽ hiển thị **"Bạn và người này cần cùng phòng để trao đổi khóa"** và sau khi **Eve** nhấn vào tên **Bob**, sẽ hiển thị **"bạn và người này đã trao đổi khóa thành công"** , kết quả như sau:



ALICE **BOB** **EVE**

=> Khi 2 người chưa trao đổi khóa cho nhau, dù 2 người có tham gia trao đổi khóa với cùng 1 người khác, thì cũng cần phải trao đổi khóa với nhau thì mới có thể nhắn tin cho nhau.

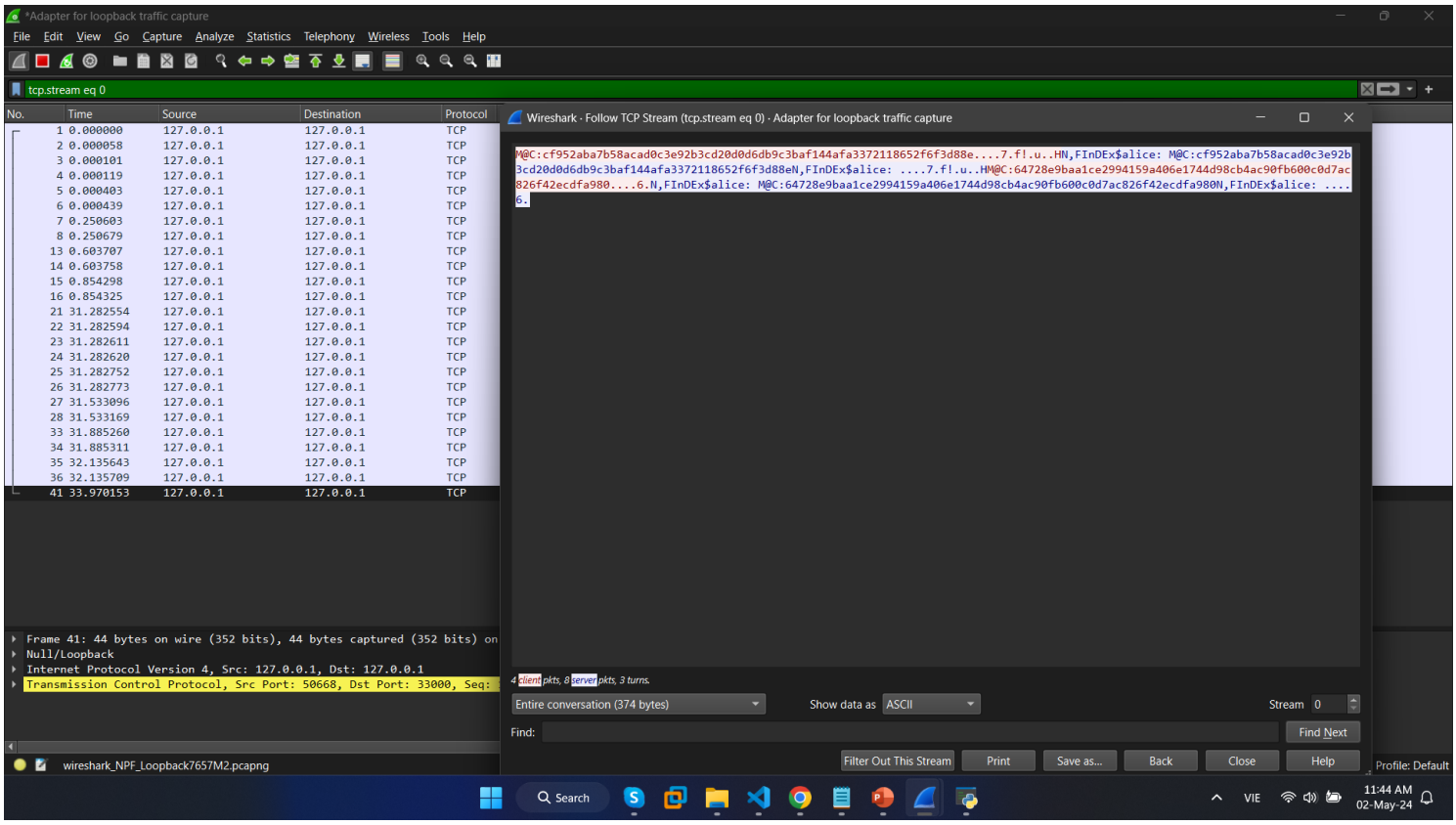
3.3.6 Trường hợp 6: Alice với Bob đã trao đổi khóa, Alice với Eve đã trao đổi khóa và Bob với Eve đã trao đổi khóa, cả 3 nhắn tin lẫn nhau



ALICE **BOB** **EVE**

=> Các cặp khóa được trao đổi lẫn nhau, mà không bị nhầm lẫn trong quá trình nhắn giữa các người dùng trên.

3.4. BẢO MẬT CHỐNG MAN-IN-THE-MIDDLE ATTACK:



Sử dụng WireShark để bắt gói tin trong quá trình nhắn tin giữa Alice và Bob trên port 33000, phân tích gói tin, ta thấy:
=> Tin nhắn văn bản đã được mã hóa đầu cuối, bên thứ 3 không thể đọc được nội dung tin nhắn

3.5. ĐÁNH GIÁ QUÁ TRÌNH LÀM VIỆC:

Vì đây là một chủ đề thực tế, do đó trong quá trình tìm hiểu, nhóm cũng gặp một số khó khăn nhất định. Tuy nhiên nhóm vẫn tuân thủ và hoàn thành theo phương hướng và kế hoạch đã đề ra ban đầu. Trong lúc tìm hiểu, nhóm cũng cảm thấy một số kiến thức và mô hình mang tính khả thi không cao. Có thể kể đến như mô hình ZKP, một mô hình đòi hỏi khá nhiều về kiến thức mã hóa và bảo mật nâng cao tương đối khó triển khai trong sản phẩm demo nhỏ thực tế lúc này. Do do nhóm đã phải điều chỉnh một số mô hình để giúp cho phù hợp hơn với nội dung. Bên cạnh đó, việc sắp xếp thời gian làm việc nhóm cũng gặp đôi chút khó khăn. Tuy nhiên, sau cùng nhóm cũng đã hoàn thành được nội dung tìm hiểu và có được sản phẩm demo thực tế tương ứng.

Nhìn chung, nhóm cảm thấy đây là một chủ đề rất hay và mang tính thực tế cao.Trong tương lai, có thể sẽ ứng dụng rộng rãi hơn và hoàn toàn có thể cải tiến và tích hợp thêm được các công nghệ mới.

3.6. KẾT LUẬN:

E2EE (Mã Hóa Đầu Cuối) là điểm cốt lõi cho trải nghiệm an toàn và riêng tư trên không gian mạng, mang lại nhiều lợi ích cho người dùng cũng như doanh nghiệp. Trong thời buổi hiện đại ngày nay, quyền riêng tư và ao toàn của người dùng trên không gian mạng đang rất nghiêm trọng. E2EE là giải pháp có thể giải quyết vấn đề bảo mật, đảm bảo dữ liệu không bị tiết lộ khi máy chủ bị xâm nhập. Có thể nói, nếu thực hiện được đầy đủ các thuật toán đáng tin cậy, E2EE sẽ có thể cung cấp mức độ an toàn cao nhất của việc bảo vệ dữ liệu.

4. NGUỒN THAM KHẢO

STT	Link tham khảo
1	https://res-zalo.zadn.vn/help/0620_E2EE_Whitepaper.pdf
2	https://inseclab.uit.edu.vn/giao-thuc-ma-hoa-dau-cuoi-message-layer-security-mls-trong-cac-ung-dung-nhan-tin-nhom/
3	https://rp.os3.nl/2018-2019/p25/report.pdf
4	https://pure.royalholloway.ac.uk/ws/files/23295486/Paper.pdf
5	https://www.aciworldwide.com/end-to-end-encryption
6	https://www.studocu.com/in/document/dr-d-y-patil-vidyapeeth-pune/computer-engineering/nis-mini-project/54851041
7	https://www.docsity.com/en/final-project-report-implementation-of-the-diffie-hellman-key-exchange-cs-5950/6419847/
8	https://www.mdpi.com/1424-8220/24/2/438
9	https://www.youtube.com/watch?v=_WMMWQ9l6a8&t=187s
10	https://www.youtube.com/watch?v=mkXdvs8H7TA

HẾT