

**HỌC VIỆN KỸ THUẬT QUÂN SỰ**

**LÝ VĂN CHẨN, NGUYỄN NGỌC KHÁNH**

**KHÓA 15**

**HỆ ĐÀO TẠO KỸ SƯ DÂN SỰ**

# **ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC**

**CHUYÊN NGÀNH: CÔNG NGHỆ DỮ LIỆU**

**XÂY DỰNG DỊCH VỤ CHAT AN TOÀN**

**NĂM 2021**

**HỌC VIỆN KỸ THUẬT QUÂN SỰ**

**LÝ VĂN CHẨN, NGUYỄN NGỌC KHÁNH**

**KHÓA 15**

**HỆ ĐÀO TẠO KỸ SƯ DÂN SỰ**

# **ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC**

**NGÀNH: CÔNG NGHỆ THÔNG TIN.**

**MÃ SỐ: 52480201**

**XÂY DỰNG DỊCH VỤ CHAT AN TOÀN**

*Cán bộ hướng dẫn: Trung tá, GV, TS. Cao Văn Lợi*

**NĂM 2021**

## NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Họ và tên: Lý Văn Chấn, Nguyễn Ngọc Khánh, Lớp: CNDL15, Khóa: 15

Ngành: Công nghệ thông tin, Chuyên ngành: Công nghệ dữ liệu.

1. Tên đề tài: Xây dựng dịch vụ chat an toàn.

2. Các số liệu ban đầu:

- Quyết định Giao đồ án tốt nghiệp đại học – Học viện KTQS
- Tài liệu tham khảo

3. Nội dung bản thuyết minh:

- Mở đầu
- Chương 1: Tổng quan về mã hóa bảo mật thông tin
- Chương 2: Tổng quan về mã hóa RSA
- Chương 3: Tổng quan về hệ thống chat an toàn
- Chương 4: Xây dựng chương trình
- Kết luận
- Tài liệu tham khảo

4. Số lượng, nội dung các bản vẽ (ghi rõ loại, kích thước và cách thực hiện các bản vẽ) và các sản phẩm cụ thể (nếu có):

Được sử dụng máy tính và máy chiếu để trình chiếu.

5. Cán bộ hướng dẫn:

- Họ và tên: Cao Văn Lợi
- Cấp bậc: Trung tá
- Học hàm, học vị: GV, TS
- Đơn vị: Bộ môn An toàn thông tin - Khoa Công nghệ thông tin
- Hướng dẫn toàn bộ

Ngày giao: 14/01/2021

Ngày hoàn thành: 04/06/2021

*Hà Nội, ngày 04, tháng 06, năm 2021*

**Chủ nhiệm bộ môn**

**Cán bộ hướng dẫn**

(Ký, ghi rõ họ tên, học hàm, học vị)

**Học viên thực hiện**

Đã hoàn thành và nộp đồ án, ngày 04 tháng 06 năm 2021

(Ký và ghi rõ họ tên)

## KÝ HIỆU VIẾT TẮT

Nội dung	Ký hiệu viết tắt
CSDL	Cơ sở dữ liệu
HTTP	HyperText Transfer Protocol
IoT	Internet of Things
MIT	Massachusetts Institute of Technology
PKI	Public Key Infrastructure
PC	Personal computer
TCP	Transmission Control Protocol

## DANH MỤC HÌNH ẢNH

Hình 1.1 Tổng quan về mã hóa .....	4
Hình 1.2 Mã hóa một chiều.....	7
Hình 1.3 Mã hóa đối xứng .....	9
Hình 1.4 Mã hóa bất đối xứng .....	11
Hình 2.1 Sơ đồ tạo khóa trên mã hóa RSA .....	16
Hình 2.1 Quy trình tạo chữ kí số cho văn bản .....	18
Hình 2.3 Biểu đồ thời gian mã hóa của thuật toán RSA dựa vào độ lớn của key .....	21
Hình 3.1 Mô hình Websocket .....	23
Hình 3.2 Hệ thống chat an toàn.....	24
Hình 3.2.1 Sơ đồ quy trình tạo khoá .....	25
Hình 3.2.2 Sơ đồ quy trình gửi tin nhắn .....	26
Hình 3.2.3 Sơ đồ quy trình nhận tin nhắn .....	27
Hình 3.2.4 Sơ đồ quy trình gửi tin nhắn lưu ở client .....	28
Hình 3.2.5 Sơ đồ quy trình nhận tin nhắn lưu ở client.....	29
Hình 4.1 Mô hình quan hệ database.....	30
Hình 4.3.1.1 Kiến trúc project backend .....	32
Hình 4.3.2.1 Kiến trúc project frontend .....	34
Hình 4.3.2.2 Hàm sinh khoá công khai và khoá bí mật .....	35
Hình 4.4.1.1 Màn hiển thị private key và public key.....	36
Hình 4.4.1.2 Hiển thị chuỗi sau khi đã mã hoá.....	37
Hình 4.4.1.3 Màn hiển thị kết quả giải mã.....	38
Hình 4.4.2.1 Màn đăng nhập.....	39
Hình 4.4.2.2 Màn tạo khoá.....	40
Hình 4.4.2.3 Màn thiết lập mức độ an toàn.....	41
Hình 4.4.2.4 Màn thông báo cập nhật khoá .....	42
Hình 4.4.2.5 Màn menu .....	43
Hình 4.4.2.6 Màn trang cá nhân.....	44
Hình 4.4.2.7 Màn hiển thị mức độ an toàn hiện tại của tài khoản .....	45
Hình 4.4.2.8 Màn thiết lập mức độ an toàn.....	46

Hình 4.4.2.9 Màn đổi mật khẩu .....	47
Hình 4.4.2.10 Màn danh sách các cuộc trò chuyện .....	48
Hình 4.4.2.11 Màn chat 2 người .....	49
Hình 4.4.2.12 Màn chat nhóm.....	50

## MỤC LỤC

MỞ ĐẦU.....	1
1. Lý do chọn đề tài.....	1
2. Cơ sở khoa học thực tiễn của đề tài .....	2
3. Mục tiêu.....	2
4. Cấu trúc đồ án .....	3
Chương 1: TỔNG QUAN VỀ MÃ HÓA BẢO MẬT THÔNG TIN .....	4
1.1 Giới thiệu về mã hóa .....	4
1.2 Thuật toán mã hóa.....	5
1.3 Phân loại các phương pháp mã hóa.....	6
1.3.1 Mã hóa cổ điển .....	7
1.3.2 Mã hóa một chiều.....	7
1.3.3 Mã hóa đối xứng .....	9
1.3.4 Mã hóa bất đối xứng .....	11
Chương 2: TỔNG QUAN VỀ MÃ HÓA RSA.....	13
2.1 Lịch sử, cách hoạt động RSA:.....	13
2.1.1 Mô tả sơ lược .....	13
2.1.2 Cơ sở toán học của thuật toán .....	14
2.1.3 Tạo khóa.....	16
2.2 Mã hóa và giải mã: .....	17
2.3 Tạo chữ kí số cho văn bản: .....	18
2.4 Các vấn đề cần đặt ra về thực tế.....	20
2.4.1 Quá trình tạo khóa.....	20
2.4.2 Tốc độ.....	21
2.4.3 Phân phối khóa.....	22
2.4.4 Nhược điểm và cách khắc phục .....	22
Chương 3: TỔNG QUAN VỀ HỆ THỐNG CHAT AN TOÀN .....	23
3.1 Tổng quan về websocket.....	23
3.2 Hệ thống chat an toàn.....	24
3.2.1 Quy trình tạo khoá.....	25



3.2.2 Quy trình gửi tin nhắn lưu ở server.....	26
3.2.3 Quy trình nhận tin nhắn lưu ở server .....	27
3.2.4 Quy trình gửi tin nhắn lưu ở client.....	28
3.2.5 Quy trình nhận tin nhắn lưu ở client .....	29
Chương 4: XÂY DỰNG TRƯỜNG TRÌNH .....	30
4.1 Mô hình quan hệ database.....	30
4.2 Tổng quan về phần mềm .....	31
4.2.1 Backend.....	31
4.2.2 Frontend .....	31
4.3 Giới thiệu về các module của chương trình .....	32
4.3.1 Backend.....	32
4.3.2 Frontend .....	34
4.4 Thử nghiệm thực tế .....	36
4.4.1 Demo mã hoá và giải mã RSA.....	36
4.4.2 Các màn hình chính của app .....	39
KẾT LUẬN .....	51
1. Kết luận .....	51
2. Hướng phát triển .....	51

## BẢNG PHÂN CÔNG NGHIỆM VỤ

Phân công	Công việc
Làm chung	<ul style="list-style-type: none"><li>- Thảo luận tìm hiểu lý thuyết tổng quan về các loại mã hóa.</li><li>- Tìm hiểu tổng quan về mã hóa RSA</li><li>- Thảo luận xây dựng hệ thống chat an toàn.</li></ul>
Lý Văn Chấn	<ul style="list-style-type: none"><li>- Phụ trách backend</li></ul>
Nguyễn Ngọc Khánh	<ul style="list-style-type: none"><li>- Phụ trách frontend</li></ul>

## MỞ ĐẦU

### 1. Lý do chọn đề tài

Trong bối cảnh ngày nay khi mạng Internet càng mở rộng với hàng tỉ thiết bị PC, Laptop, smart phone, IoT được kết nối chặt chẽ với nhau, yêu cầu mở rộng đầu tư cho an ninh thông tin, an toàn mạng trở nên thiết thực hơn bao giờ hết. Bên cạnh những nguy cơ phổ biến như tấn công từ chối dịch vụ DDoS, chèn SQL Injection, khai thác Exploit CVE thì việc mã hóa thông tin đóng vai trò vô cùng then chốt trong ngành An toàn thông tin.

Yêu cầu bảo vệ quyền riêng tư đối với dữ liệu số hoá và bảo vệ bí mật của các thuật toán xử lý dữ liệu đang tăng lên rất nhanh trong những năm gần đây, đặc biệt là với xu thế phát triển của điện toán đám mây và sự xuất hiện của các kiểu tấn công phá huỷ dữ liệu, đánh cắp thông tin nhạy cảm. Để lưu trữ và truy cập dữ liệu an toàn, người dùng thường sử dụng các công nghệ như mã hoá và các phần cứng chống can thiệp. Sau đó, tiến hành giải mã dữ liệu để sử dụng. Điều đó dẫn đến nhu cầu tính toán với dữ liệu bí mật ngay ở dạng mã hoá. Từ đó, mã hoá đồng cấu (homomorphic encryption) xuất hiện và được các nhà nghiên cứu kỳ vọng.

Trong mọi lĩnh vực kinh tế, chính trị, xã hội, quân sự... luôn có nhu cầu trao đổi thông tin giữa các cá nhân, các công ty, tổ chức, hoặc giữa các quốc gia với nhau. Ngày nay, với sự phát triển của công nghệ thông tin đặt biệt là mạng internet thì việc truyền tải thông tin đã dễ dàng và nhanh chóng hơn.

Và vấn đề đặt ra là tính bảo mật trong quá trình truyền tải thông tin, đặt biệt quan trọng đối với những thông tin liên quan đến chính trị, quân sự, hợp đồng kinh tế... Vì vậy ngành khoa học nghiên cứu về mã hóa thông tin được phát triển. Việc mã hóa là làm cho thông tin biến sang một dạng khác

khi đó chỉ có bên gửi và bên nhận mới đọc được, còn người ngoài dù nhận được thông tin nhưng cũng không thể hiểu được nội dung.

Vì thế đề tài “Xây dựng dịch vụ chat an toàn” được nhóm em lựa chọn nhằm nghiên cứu các giải pháp mã hóa bảo mật thông tin để xây dựng một ứng dụng chat trực tuyến riêng tư và bảo mật cho người dùng. Nội dung cuộc trò chuyện chỉ có thể đọc bởi người gửi và người nhận. Thuật toán mã hóa RSA là thuật toán mang tính kinh điển của ngành mật mã học sẽ được nhóm em lựa chọn để mã hóa thông tin.

## **2. Cơ sở khoa học thực tiễn của đề tài**

Cơ sở khoa học:

- Cơ sở lý thuyết về các phương pháp mã hóa bảo mật thông tin.

Ý nghĩa thực tiễn đề tài:

Với sự phát triển công nghệ như ngày nay thì việc rò rỉ thông tin là việc rất khó tránh khỏi, khi đó việc mã hóa các thông tin quan trọng càng trở nên cần thiết hơn bao giờ hết.

Việc mã hóa là để đảm bảo tính an toàn cho thông tin, đặc biệt trong thời đại công nghệ số như hiện nay. Đặc biệt là trong giao dịch điện tử. Có thể nói mã hóa chính là việc đảm bảo bí mật, toàn vẹn thông tin, khi thông tin được truyền trên mạng internet. Mã hóa cũng là nền tảng của kỹ thuật chữ ký điện tử, hệ thống PKI

## **3. Mục tiêu**

Tìm hiểu một số phương pháp mã hóa bảo mật thông tin.

Xây dựng ứng dụng chat an toàn với mã hóa RSA.

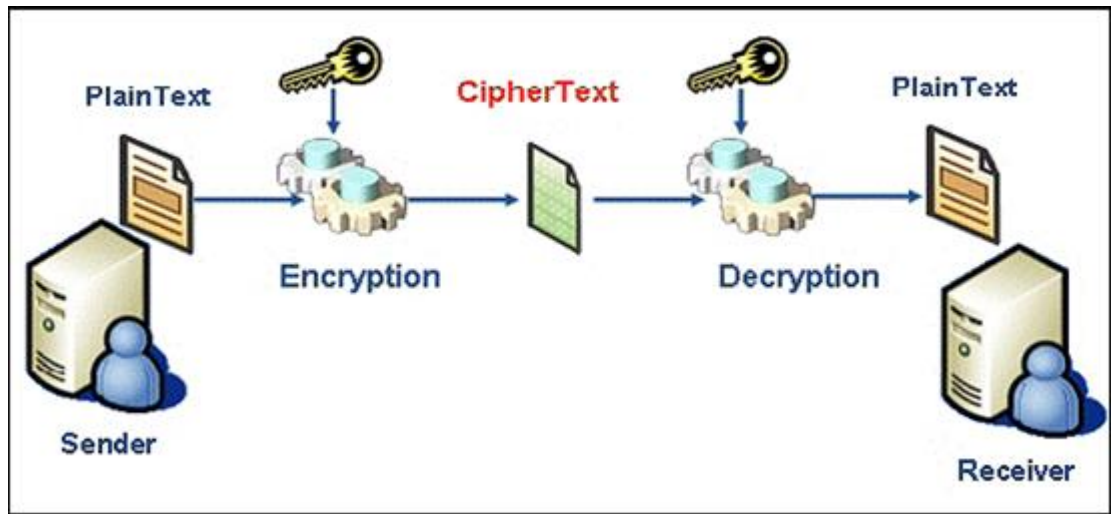
#### **4. Cấu trúc đề án**

- Mở đầu
- Chương 1: Tổng quan về mã hóa bảo mật thông tin
- Chương 2: Tổng quan về mã hóa RSA
- Chương 3: Tổng quan về hệ thống chat an toàn
- Chương 4: Xây dựng chương trình
- Kết luận

## Chương 1: TỔNG QUAN VỀ MÃ HÓA BẢO MẬT THÔNG TIN

### 1.1 Giới thiệu về mã hóa

Mạng máy tính là một môi trường mở, những thông tin ta gửi lên internet hoặc nhận về internet đều có thể bị nghe trộm. Do đó việc bảo mật những thông tin này là cần thiết, và một trong những cách để bảo mật thông tin hữu hiệu nhất hiện nay là mã hóa.



*Hình 1.1 Tổng quan về mã hóa*

Cụm từ Mã hóa có thể cảm thấy hơi xa lạ. Nhưng nó là một thứ cực kì quan trọng, và hiện hữu ở rất nhiều nơi trong đời sống hàng ngày của chúng ta. Để biết được nó quan trọng như thế nào, và được sử dụng rộng rãi ra sao, hãy tưởng tượng: Nếu không có mã hóa, hệ thống ATM sẽ không tồn tại, sẽ không tồn tại chuỗi hệ thống ngân hàng, sẽ không có giao dịch mua bán online, internet sẽ không phát triển... Và nếu không có mã hóa, ta sẽ không thể ngồi đây, ngay giờ này và đọc bài viết này, bởi không có nó thì internet sẽ không thể phát triển được như ngày nay.

Giả sử con người chưa biết đến mã hóa, một hacker chỉ cần làm một thiết bị lắng nghe và chuyển đổi các gói packet được truyền đi trong mạng và gán thiết bị này vào cáp mạng của một máy ATM nào đó. Khi bạn hoặc ai đó sử dụng máy ATM này để chuyển tiền, thiết bị này chỉ việc phân tích các packet chứa thông tin giao dịch được truyền đi, và chuyển đổi số tài khoản mà bạn muốn gửi thành số tài khoản của anh ta. Và thế là, tất cả số tiền giao dịch của máy ATM đó sẽ chảy vào túi của anh ta, trong khi anh ta chỉ việc ngồi máy lạnh soi cà phê! Nếu vậy thì làm sao máy ATM có thể tồn tại? Vì nó quá thiếu an toàn nên sẽ không ai dùng đến nó, và sẽ không ai tạo ra nó.

Ta có thể dễ dàng khái quát, mã hóa là một phương pháp bảo vệ thông tin, bằng cách chuyển đổi thông tin từ dạng rõ (Thông tin có thể dễ dàng đọc hiểu được) sang dạng mờ (Thông tin đã bị che đi, nên không thể đọc hiểu được. Để đọc được ta cần phải giải mã nó). Nó giúp ta có thể bảo vệ thông tin, để những kẻ đánh cắp thông tin, dù có được thông tin của chúng ta, cũng không thể hiểu được nội dung của nó.

Lấy ví dụ, khi bạn muốn gửi thư cho bạn mình, và trong đó chứa những thông tin quan trọng mà bạn không muốn ai biết (Giả sử nội dung ban đầu là "Ngày mai xăng tăng giá đó"). Do đó bạn muốn bảo mật thông tin này, để dù có người cố tình đọc trộm nội dung thì cũng không thể hiểu, thì bạn sẽ mã hóa nó (Giả sử bạn mã hóa thành "fd%\$23fDd432FDs4#@Vdserf3%\$3"). Xong khi đưa đến bạn mình, bạn sẽ bày cho họ cách giải mã để họ có thể hiểu được nội dung thư.

## **1.2 Thuật toán mã hóa**

Thuật toán mã hóa là một thuật toán nhằm mã hóa thông tin của chúng ta, biến đổi thông tin từ dạng rõ sang dạng mờ, để ngăn cản việc đọc trộm nội

dung của thông tin (Dù hacker có được thông tin đó cũng không hiểu nội dung chứa trong nó là gì).

Thông thường các thuật toán sử dụng một hoặc nhiều key (Một chuỗi chìa khóa để mã hóa và giải mã thông tin) để mã hóa và giải mã (Ngoại trừ những thuật toán cổ điển). Ta có thể coi key này như một cái password để có thể đọc được nội dung mã hóa. Người gửi sẽ dùng key mã hóa để mã hóa thông tin sang dạng mờ, và người nhận sẽ sử dụng key giải mã để giải mã thông tin sang dạng rõ. Chỉ những người nào có key giải mã mới có thể đọc được nội dung.

Nhưng đôi khi "kẻ thứ ba" (hacker) không có key giải mã vẫn có thể đọc được thông tin, bằng cách phá vỡ thuật toán. Và có một nguyên tắc là bất kì thuật toán mã hóa nào cũng đều có thể bị phá vỡ. Do đó không có bất kì thuật toán mã hóa nào được coi là an toàn mãi mãi. Độ an toàn của thuật toán được dựa vào nguyên tắc:

Nếu chi phí để giải mã một khối lượng thông tin lớn hơn giá trị của khối lượng thông tin đó thì thuật toán đó được tạm coi là an toàn. (Không ai lại đi bỏ ra 50 năm để giải mã một thông tin mà chỉ mang lại cho anh ta 1000 đô).

Nếu thời gian để phá vỡ một thuật toán là quá lớn (giả sử lớn hơn 100 năm, 1000 năm) thì thuật toán được tạm coi là an toàn.

### **1.3 Phân loại các phương pháp mã hóa**

Có rất nhiều loại phương pháp mã hóa khác nhau đã ra đời. Mỗi loại có những ưu và nhược điểm riêng. Ta có thể phân chia các phương pháp mã hóa thành 4 loại chính:

- Mã hóa cổ điển



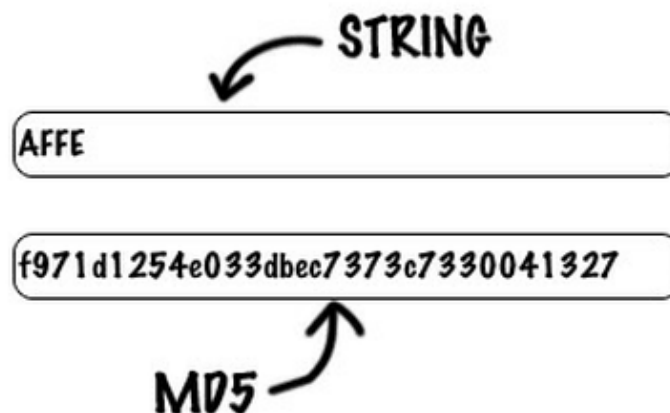
- Mã hóa một chiều
- Mã hóa đối xứng
- Mã hóa bất đối xứng

### 1.3.1 Mã hóa cổ điển

Đây là phương pháp mã hóa đầu tiên, và cổ xưa nhất, và hiện nay rất ít được dùng đến so với các phương pháp khác. Ý tưởng của phương pháp này rất đơn giản, bên A mã hóa thông tin bằng thuật toán mã hóa cổ điển, và bên B giải mã thông tin, dựa vào thuật toán của bên A, mà không dùng đến bất kì key nào. Do đó, độ an toàn của thuật toán sẽ chỉ dựa vào độ bí mật của thuật toán, vì chỉ cần ta biết được thuật toán mã hóa, ta sẽ có thể giải mã được thông tin.

Một ví dụ về phương pháp mã hóa cổ điển: Giả sử ta mã hóa bằng cách thay đổi một kí tự trong chuỗi cần mã hóa thành kí tự liền kề (“Di hoc ve” thành “Ek ipd xg”). Thì bất cứ người nào, chỉ cần biết cách ta mã hóa, đều có thể giải mã được.

### 1.3.2 Mã hóa một chiều



Hình 1.2 Mã hóa một chiều

Đôi khi ta chỉ cần mã hóa thông tin chứ không cần giải mã thông tin, khi đó ta sẽ dùng đến phương pháp mã hóa một chiều (Chỉ có thể mã hóa chứ không thể giải mã). Thông thường phương pháp mã hóa một chiều sử dụng một hàm băm (hash function) để biến một chuỗi thông tin thành một chuỗi hash có độ dài nhất định. Ta không có bất kì cách nào để khôi phục (hay giải mã) chuỗi hash về lại chuỗi thông tin ban đầu.

Hàm băm (Hash function) là một hàm mà nó nhận vào một chuỗi có độ dài bất kì, và sinh ra một chuỗi kết quả có độ dài cố định (Gọi là chuỗi hash), dù hai chuỗi dữ liệu đầu vào, được cho qua hàm băm thì cũng sinh ra hai chuỗi hash kết quả khác nhau rất nhiều. Ví dụ như đối với kiểu dữ liệu Hash-table, ta có thể coi đây là một dạng kiểu dữ liệu mảng đặc biệt mà index nó nhận vào là một chuỗi, nó được định nghĩa bằng cách bên trong nó chứa một mảng thông thường, mỗi khi truyền vào index là một chuỗi, thì chuỗi này sẽ đi qua hàm băm và ra một giá trị hash, giá trị này sẽ tương ứng với index thật của phần tử đó trong mảng bên dưới.

Đặc điểm của hash function là khi thực hiện băm hai chuỗi dữ liệu như nhau, dù trong hoàn cảnh nào thì nó cũng cùng cho ra một chuỗi hash duy nhất có độ dài nhất định và thường nhỏ hơn rất nhiều so với chuỗi gốc, và hai chuỗi thông tin bất kì dù khác nhau rất ít cũng sẽ cho ra chuỗi hash khác nhau rất nhiều. Do đó hash function thường được sử dụng để kiểm tra tính toàn vẹn của dữ liệu.

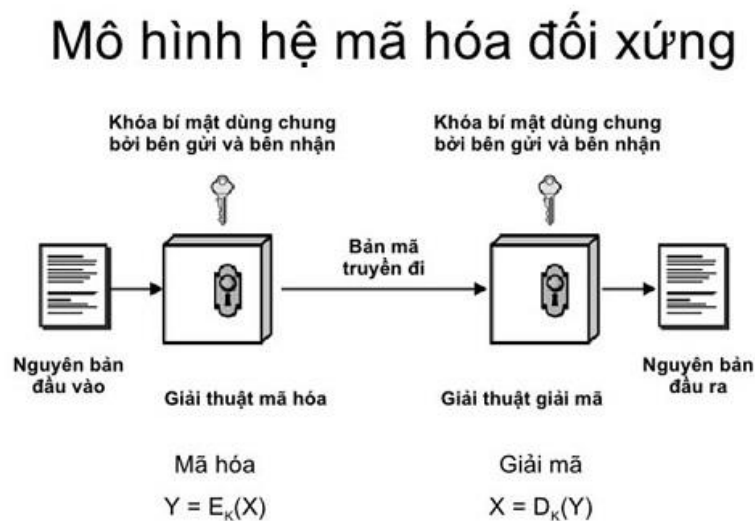
Giả sử bạn có một file dữ liệu định up lên mạng, và bạn muốn người dùng có thể kiểm tra xem dữ liệu họ down về có chính xác dữ liệu mình up lên hay không. Thì bạn sẽ dùng một hash function để băm dữ liệu của file đó ra một chuỗi hash, và gửi kèm cho người dùng chuỗi hash này. Khi đó, người dùng chỉ việc dùng đúng hash function đó để tìm chuỗi hash hiện tại

của file down về, rồi so sánh với chuỗi hash ban đầu, nếu hai chuỗi này giống nhau thì dữ liệu down về vẫn toàn vẹn.

Ngoài ra có một ứng dụng mà có thể ta thường thấy, đó là để lưu giữ mật khẩu. Vì mật khẩu là một thứ cực kì quan trọng, do đó ta không nên lưu mật khẩu của người dùng dưới dạng rõ, vì như vậy nếu bị hacker tấn công, lấy được CSDL thì hacker có thể biết được mật khẩu của người dùng. Do đó, mật khẩu của người dùng nên được lưu dưới dạng chuỗi hash, và đối với server thì chuỗi hash đó chính là “mật khẩu” đăng nhập (lúc đăng nhập thì mật khẩu mà người dùng nhập cũng được mã hóa thành chuỗi hash và so sánh với chuỗi hash trong CSDL của server). Dù hacker có lấy được CSDL thì cũng không tài nào có thể giải mã được chuỗi hash để tìm ra mật khẩu của người dùng.

Thuật toán mã hóa một chiều (hàm băm) mà ta thường gặp nhất là MD5 và SHA.

### 1.3.3 Mã hóa đối xứng



Hình 1.3 Mã hóa đối xứng

Mã hóa đối xứng (Hay còn gọi là mã hóa khóa bí mật) là phương pháp mã hóa mà key mã hóa và key giải mã là như nhau (Sử dụng cùng một secret key để mã hóa và giải mã). Đây là phương pháp thông dụng nhất hiện nay dùng để mã hóa dữ liệu truyền nhận giữa hai bên. Vì chỉ cần có secret key là có thể giải mã được, nên bên gửi và bên nhận cần làm một cách nào đó để cùng thống nhất về secret key.

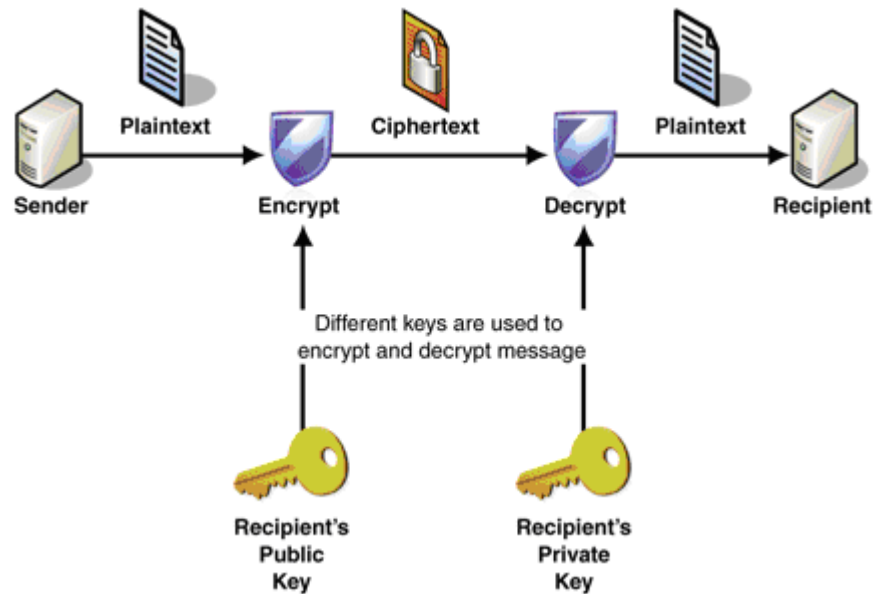
Để thực hiện mã hóa thông tin giữa hai bên thì:

- Đầu tiên bên gửi và bên nhận bằng cách nào đó sẽ phải thỏa thuận secret key (khóa bí mật) được dùng để mã hóa và giải mã. Vì chỉ cần biết được secret key này thì bên thứ ba có thể giải mã được thông tin, nên thông tin này cần được bí mật truyền đi (bảo vệ theo một cách nào đó).
- Sau đó bên gửi sẽ dùng một thuật toán mã hóa với secret key tương ứng để mã hóa dữ liệu sắp được truyền đi. Khi bên nhận nhận được sẽ dùng chính secret key đó để giải mã dữ liệu.

Vấn đề lớn nhất của phương pháp mã hóa đối xứng là làm sao để “thỏa thuận” secret key giữa bên gửi và bên nhận, vì nếu truyền secret key từ bên gửi sang bên nhận mà không dùng một phương pháp bảo vệ nào thì bên thứ ba cũng có thể dễ dàng lấy được secret key này.

Các thuật toán mã hóa đối xứng thường gặp: DES, AES...

### 1.3.4 Mã hóa bất đối xứng



*Hình 1.4 Mã hóa bất đối xứng*

Mã hóa bất đối xứng (Hay còn gọi là mã hóa khóa công khai) là phương pháp mã hóa mà key mã hóa (lúc này gọi là public key – khóa công khai) và key giải mã (lúc này gọi là private key – khóa bí mật) khác nhau. Nghĩa là key ta sử dụng để mã hóa dữ liệu sẽ khác với key ta dùng để giải mã dữ liệu. Tất cả mọi người đều có thể biết được public key (kể cả hacker), và có thể dùng public key này để mã hóa thông tin. Nhưng chỉ có người nhận mới nắm giữ private key, nên chỉ có người nhận mới có thể giải mã được thông tin.

Để thực hiện mã hóa bất đối xứng thì:

- Bên nhận sẽ tạo ra một cặp khóa (public key và private key). Bên nhận sẽ giữ lại private key và truyền cho bên gửi public key. Vì public key này là công khai nên có thể truyền tự do mà không cần bảo mật.
- Bên gửi trước khi gửi dữ liệu sẽ mã hóa dữ liệu bằng thuật toán mã hóa bất đối xứng với key là public key từ bên nhận.

- Bên nhận sẽ giải mã dữ liệu nhận được bằng thuật toán được sử dụng ở bên gửi, với key giải mã là private key.

Điểm yếu lớn nhất của mã hóa bất đối xứng là tốc độ mã hóa và giải mã rất chậm so với mã hóa đối xứng, nếu dùng mã hóa bất đối xứng để mã hóa dữ liệu truyền – nhận giữa hai bên thì sẽ tốn rất nhiều chi phí.

Do đó, ứng dụng chính của mã hóa bất đối xứng là dùng để bảo mật secret key cho mã hóa đối xứng: Ta sẽ dùng phương pháp mã hóa bất đối xứng để truyền secret key của bên gửi cho bên nhận. Và hai bên sẽ dùng secret key này để trao đổi thông tin bằng phương pháp mã hóa đối xứng.

Thuật toán mã hóa bất đối xứng thường thấy: RSA.

## **Chương 2:**

### **TỔNG QUAN VỀ MÃ HÓA RSA**

#### **2.1 Lịch sử, cách hoạt động RSA:**

Trong phần đầu tiên này, chúng ta sẽ tìm hiểu về lý thuyết nền tảng về đặc điểm thông dụng mã hóa RSA. Bằng những thông tin này, người đọc có thể có cái nhìn tổng quát về mã hóa RSA.

Thuật toán được Ron Rivest, Adi Shamir và Len Adleman mô tả lần đầu tiên vào năm 1977 tại Học viện Công nghệ Massachusetts (MIT). Tên của thuật toán lấy từ 3 chữ cái đầu của tên 3 tác giả.

Trước đó, vào năm 1973, Clifford Cocks, một nhà toán học người Anh làm việc tại GCHQ, đã mô tả một thuật toán tương tự. Với khả năng tính toán tại thời điểm đó thì thuật toán này không khả thi và chưa bao giờ được thực nghiệm. Tuy nhiên, phát minh này chỉ được công bố vào năm 1997 vì được xếp vào loại tuyệt mật.

Thuật toán RSA được MIT đăng ký bằng sáng chế tại Hoa Kỳ vào năm 1983 (Số đăng ký 4.405.829). Bằng sáng chế này hết hạn vào ngày 21 tháng 9 năm 2000. Tuy nhiên, do thuật toán đã được công bố trước khi có đăng ký bảo hộ nên sự bảo hộ hầu như không có giá trị bên ngoài Hoa Kỳ. Ngoài ra, nếu như công trình của Clifford Cocks đã được công bố trước đó thì bằng sáng chế RSA đã không thể được đăng ký.

##### **2.1.1 Mô tả sơ lược**

Thuật toán RSA có hai khóa: khóa công khai (hay khóa công cộng) và khóa bí mật (hay khóa cá nhân). Mỗi khóa là những số cố định sử dụng trong

quá trình mã hóa và giải mã. Khóa công khai được công bố rộng rãi cho mọi người và được dùng để mã hóa.

Những thông tin được mã hóa bằng khóa công khai chỉ có thể được giải mã bằng khóa bí mật tương ứng. Nói cách khác, mọi người đều có thể mã hóa nhưng chỉ có người biết khóa cá nhân (bí mật) mới có thể giải mã được. Ta có thể mô phỏng trực quan một hệ mật mã khóa công khai như sau:

- Bob muốn gửi cho Alice một thông tin mật mà Bob muốn duy nhất Alice có thể đọc được.
- Alice gửi cho Bob một chiếc hộp có khóa đã mở sẵn và giữ lại chìa khóa. Bob nhận chiếc hộp, cho vào đó một tờ giấy viết thư bình thường và khóa lại (như loại khoá thông thường chỉ cần sập chốt lại, sau khi sập chốt khóa ngay cả Bob cũng không thể mở lại được- không đọc lại hay sửa thông tin trong thư được nữa).
- Bob gửi chiếc hộp lại cho Alice. Alice mở hộp với chìa khóa của mình và đọc thông tin trong thư. Trong ví dụ này, chiếc hộp với khóa mở đóng vai trò khóa công khai, chiếc chìa khóa chính là khóa bí mật.

### 2.1.2 Cơ sở toán học của thuật toán

Khi mã hóa tin nhắn, RSA nhìn thấy thông điệp gồm các số lớn, và mã hóa bao gồm chủ yếu là phép nhân số lớn. Để hiểu cách thức hoạt động của RSA, chúng ta cần phải biết những con số lớn mà nó vận dụng và làm thế nào các phép nhân hoạt động trên những con số đó. RSA thấy bản rõ mà nó đang mã hóa như một số nguyên dương giữa 1 và  $n - 1$ , trong đó  $n$  là một số lớn được gọi là mô đun.



Để tìm số các phần tử trong một nhóm  $Z_n^*$  khi  $n$  không phải là số nguyên tố, chúng ta sử dụng hàm số của Euler, được viết là  $\varphi(n)$ , với  $\varphi$  đại diện cho chữ Hy Lạp phi. Hàm này cho biết số lượng các nguyên tố co-prime với  $n$ , đó là số lượng các phần tử trong  $Z_n^*$ . Theo nguyên tắc, nếu  $n$  là một sản phẩm của số nguyên tố  $n = p_1 \times p_2 \times \dots \times p_m$ , số phần tử trong nhóm  $Z_n^*$  là như sau:

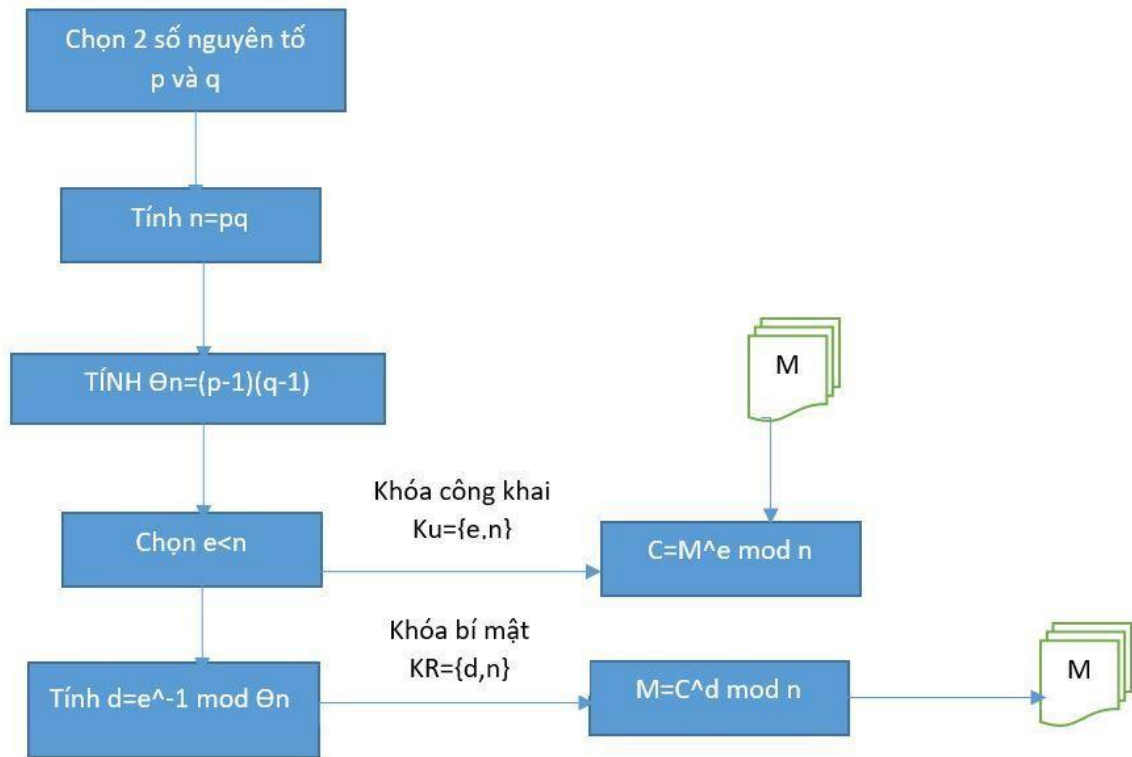
$$\varphi(n) = (p_1 - 1) \times (p_2 - 1) \times \dots \times (p_m - 1)$$

RSA chỉ đề cập đến các số  $n$  là sản phẩm của hai số nguyên tố lớn, thường được ghi nhận là  $n = pq$ . Nhóm liên kết  $Z_N^*$  sau đó sẽ chứa các phần tử  $\varphi(n) = (p - 1)(q - 1)$ .

Bằng cách mở rộng biểu thức này, chúng ta có được định nghĩa tương đương  $\varphi(n) = n - p - q + 1$ , hoặc  $\varphi(n) = (n + 1) - (p + q)$ , biểu hiện trực quan hơn giá trị của  $\varphi(n)$  tương ứng với  $n$

Nói cách khác, tất cả các số  $(p + q)$  giữa 1 và  $n - 1$  thuộc về  $Z_N^*$  và là "số hợp lệ" trong các cách tính toán của RSA

### 2.1.3 Tạo khóa



Hình 2.1 Sơ đồ tạo khóa trên mã hóa RSA

Các khóa cho thuật toán RSA được tạo ra như sau:

Chọn 2 số nguyên tố lớn  $p$  và  $q$  với  $p$  khác  $q$ , lựa chọn ngẫu nhiên và độc lập.

- Tính  $n=pq$
- Tính một số giả nguyên tố bằng phi hàm Camichael như sau:  $\lambda(n) = \text{BCNN}(\lambda(p), \lambda(q)) = \text{BCNN}(p-1, q-1)$ . Giá trị này sẽ được giữ bí mật.
- Chọn một số tự nhiên  $e$  trong khoảng  $(1, \lambda(n))$  sao cho  $\text{UCLN}(e, \lambda(n)) = 1$ , tức là  $e$  và  $\lambda(n)$  nguyên tố cùng nhau.

- Tính toán số  $d$  sao cho  $d \equiv 1/e \pmod{\lambda(n)}$  hay viết dễ hiểu hơn thì  $de \equiv 1 \pmod{\lambda(n)}$ . Số  $d$  được gọi là số nghịch đảo modulo của  $e$  (theo modulo  $\lambda(n)$ ).
- Public key sẽ là bộ số  $(n, e)$ , và private key sẽ là bộ số  $(n, d)$ . Chúng ta cần giữ private key thật cẩn thận cũng như các số nguyên tố  $p$  và  $q$  vì từ đó có thể tính toán các khóa rất dễ dàng.

## 2.2 Mã hóa và giải mã:

Nếu chúng ta có bản rõ  $M$ , chúng ta cần chuyển nó thành một số tự nhiên  $m$  trong khoảng  $(0, n)$  sao cho  $m, n$  nguyên tố cùng nhau. Việc này rất dễ dàng thực hiện bằng cách thêm một các kỹ thuật padding.

Tiếp theo, chúng ta sẽ mã hóa  $m$ , thành  $c$  như sau:

$$C \equiv m^e \pmod{n}$$

Sau đó giá trị  $c$  sẽ được chuyển cho người nhận. Ở phía người nhận, họ sẽ giải mã từ  $c$  để lấy được  $m$  như sau:

$$C^d \equiv m^{de} \equiv m \pmod{n}$$

Từ  $m$  có thể lấy lại được bản tin bằng cách đảo ngược padding. Chúng ta lấy một ví dụ đơn giản như sau:

- $p = 5$
- $q = 7$
- $n = p \cdot q = 35$   
 $\Rightarrow \lambda(n) = 24$
- Chọn  $e = 5$  vì  $\text{UCLN}(5, 24) = 1$ , cuối cùng chọn  $d = 29$  vì  $ed - 1 = 29 \times 5 - 1$  chia hết cho 24.

- Giả sử  $m = 32$  (dấu cách), chúng ta sẽ mã hóa  $m$  và thu được:  

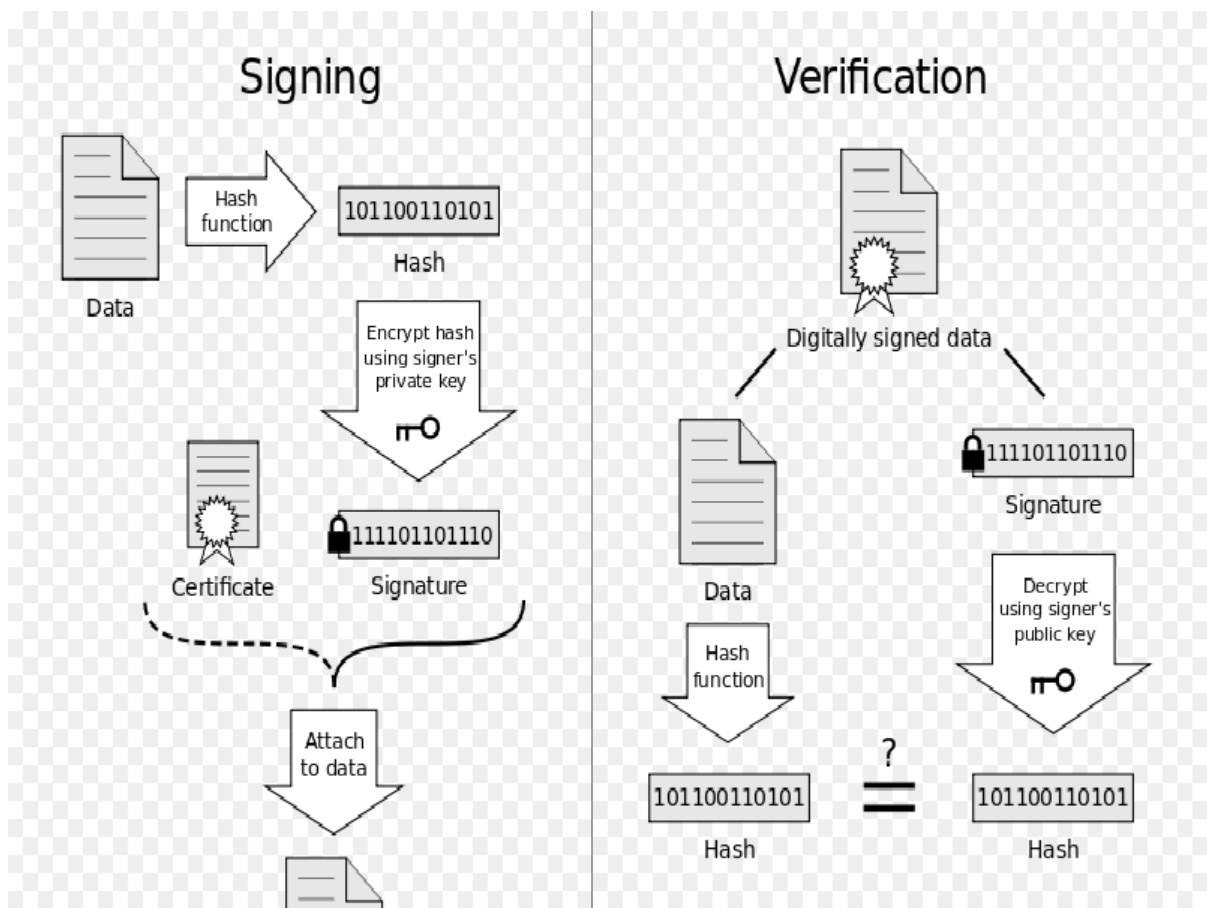
$$c = 32^5 \% 35 = 2$$

Giải mã  $c$  để thu được  $m$ :

- $m = 2^{29} \% 35 = 32$

Đây chính là  $m$  ban đầu. Ta có thể thử  $m$  với các giá trị khác nhau để thấy thuật toán hoàn toàn chính xác.

### 2.3 Tạo chữ kí số cho văn bản:



Hình 2.1 Quy trình tạo chữ kí số cho văn bản

Việc ký tên và xác thực chữ ký số sử dụng hệ mã hóa RSA tương tự như quá trình mã hóa mà giải mã ở trên. Tuy nhiên vai trò của public key và private thì có thay đổi đôi chút.

Để tạo chữ ký, người gửi sẽ dùng private key và người nhận sẽ dùng public key để xác thực chữ ký đó.

Tuy nhiên, vì bản tin rất dài nên việc mã hóa toàn bộ bản tin sẽ rất mất thời gian. Vì vậy, trong thực hành, chữ ký số thường sử dụng phương pháp mã hóa giá trị hash của bản tin. Việc này mang lại rất nhiều lợi ích như:

- Các hàm hash là hàm 1 chiều, vì vậy dù có được hash cũng không thể biết được bản tin gốc như thế nào.
- Độ dài hash là cố định và thường rất nhỏ, vì vậy chữ số sẽ không chiếm quá nhiều dung lượng.
- Giá trị hash còn có thể dùng để kiểm tra lại bản tin nhận được có nguyên vẹn hay không?

Chữ ký số đem lại nhiều giá trị hơn chữ ký tay rất nhiều. Có lẽ cũng vì vậy, việc xử lý chữ ký số phức tạp hơn hẳn chữ ký tay truyền thống.

- **Xác định nguồn gốc:** Hệ mã hóa bất đối xứng cho phép tạo chữ ký với private key mà chỉ người chủ mới biết. Khi nhận gói tin, người nhận xác thực chữ ký bằng cách dùng public key giải mã, sau đó tính giá trị hash của bản tin gốc và so sánh với hash trong gói tin nhận được. Hai chuỗi này phải trùng khớp với nhau. Tất nhiên hệ mã hóa RSA vẫn có những thách thức về an ninh nhất định nhưng dù sao thì nó vẫn khá an toàn.
- **Dữ liệu được giữ một cách toàn vẹn:** Tin nhắn gửi từ chủ private key rất khó có thể bị giả mạo. Bởi vì nếu thay đổi tin nhắn thì giá

trị hash cũng phải thay đổi theo. Những kẻ nghe lén trong mạng đương nhiên là có thể tìm cách đọc tin nhắn gốc và cả hash của nó. Nhưng hẳn ta không thể thay đổi tin nhắn được vì hẳn không có private key để sửa đổi chữ ký số cho phù hợp.

- **Chữ ký số không thể phủ nhận:** Trong giao dịch, một gói tin kèm chữ ký số rất dễ dàng tìm ra được nguồn gốc của chữ ký đó. Bởi vì private key là bí mật và chỉ người chủ của nó mới có thể biết, họ không thể chối cãi rằng chữ ký này không phải do họ phát hành. Cũng tương tự trường hợp trên, hệ mã hóa RSA hay bất kỳ hệ mã hóa nào khác cũng đều có những vấn đề về an ninh nên việc này không thể đảm bảo tuyệt đối chính xác được.

## 2.4 Các vấn đề cần đặt ra về thực tế

### 2.4.1 Quá trình tạo khóa

Việc tìm ra 2 số nguyên tố đủ lớn  $p$  và  $q$  thường được thực hiện bằng cách thử xác suất các số ngẫu nhiên có độ lớn phù hợp (dùng phép kiểm tra nguyên tố cho phép loại bỏ hầu hết các hợp số). Hai giá trị  $p$  và  $q$  còn cần được chọn không quá gần nhau để phòng trường hợp phân tích  $n$  bằng phương pháp phân tích Fermat.

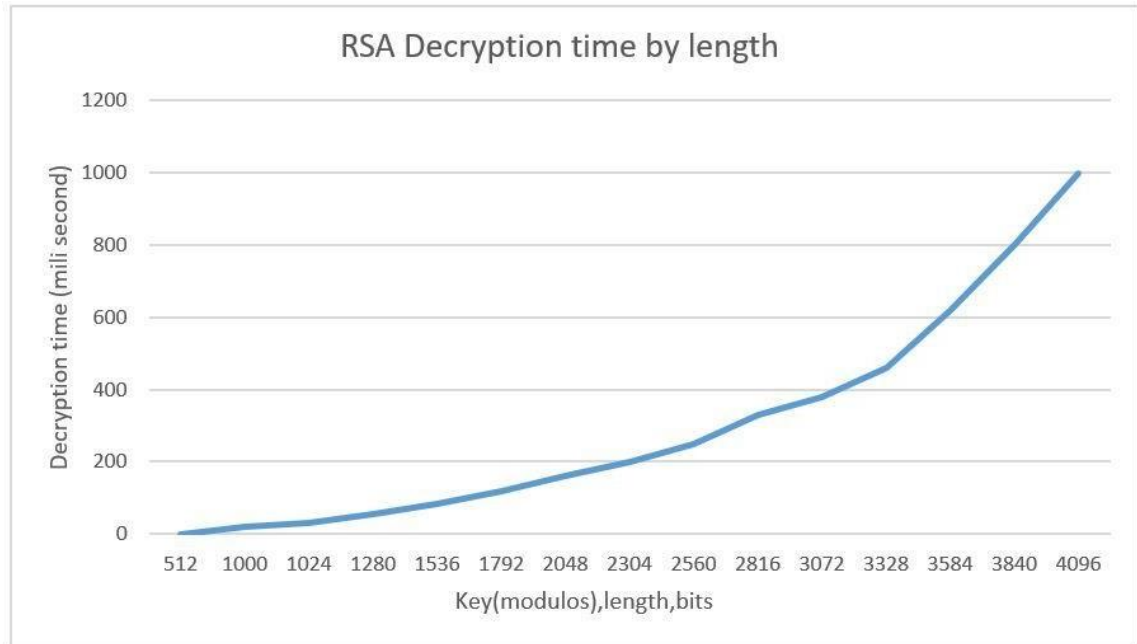
Nếu  $p-1$  hoặc  $q-1$  có thừa số nguyên tố nhỏ thì cũng có thể dễ dàng bị phân tích và vì thế  $p$  và  $q$  cũng cần được thử để tránh khả năng này.

Một điểm nữa cần nhấn mạnh là khóa bí mật  $d$  phải đủ lớn. Năm 1990, Wiener chỉ ra rằng nếu giá trị của  $p$  nằm trong khoảng  $q$  và  $2q$  (khá phổ biến) và  $d < n^{1/4/3}$  thì có thể tìm ra được  $d$  từ  $n$  và  $e$ .

Mặc dù  $e$  đã từng có giá trị là 3 nhưng hiện nay các số mũ nhỏ không còn được sử dụng do có thể tạo nên những lỗ hổng (đã đề cập ở phần chuyển đổi

văn bản rõ). Giá trị thường dùng hiện nay là 65537 vì được xem là đủ lớn và cũng không quá lớn ảnh hưởng tới việc thực hiện hàm mũ.

### 2.4.2 Tốc độ



*Hình 2.3 Biểu đồ thời gian mã hóa của thuật toán RSA dựa vào độ lớn của key*

RSA có tốc độ thực hiện chậm hơn đáng kể so với DES và các thuật toán mã hóa đối xứng khác. Trên thực tế, Bob sử dụng một thuật toán mã hóa đối xứng nào đó để mã hóa văn bản cần gửi và chỉ sử dụng RSA để mã hóa khóa để giải mã (thông thường khóa ngắn hơn nhiều so với văn bản).

Phương thức này cũng tạo ra những vấn đề an ninh mới. Một ví dụ là cần phải tạo ra khóa đối xứng thật sự ngẫu nhiên. Nếu không, kẻ tấn công (thường ký hiệu là Eve) sẽ bỏ qua RSA và tập trung vào việc đoán khóa đối xứng.

### 2.4.3 Phân phối khóa

Cũng giống như các thuật toán mã hóa khác, cách thức phân phối khóa công khai là một trong những yếu tố quyết định đối với độ an toàn của RSA.

Quá trình phân phối khóa cần chống lại được tấn công ở giữa (man-in-the-middle attack). Các phương pháp chống lại dạng tấn công này thường dựa trên các chứng thực khóa công khai (digital certificate).

Các thành phần của hạ tầng khóa công khai (public key infrastructure - PKI).

### 2.4.4 Nhược điểm và cách khắc phục

Năm 2010, các nhà khoa học thuộc Đại học Michigan đã công bố phát hiện một kẽ hở trong hệ thống mật mã hoá RSA. Cách phá vỡ hệ thống, lấy khoá bí mật RSA 1024 bit chỉ trong vài ngày thay vì vài năm nếu tấn công theo cách thông thường - tấn công bằng brute force (dò tìm lần lượt). Các nhà khoa học tạo một điện thế lớn để gây lỗi hệ thống, từ đó giúp tìm ra khoá bí mật. Việc tấn công được thực hiện trên một FPGA. Báo cáo được trình bày tại hội nghị DATE 2010 diễn ra tại Dresden, Đức tháng 3 năm 2010.

Các cách khắc phục nhược điểm trên:

- Gợi ý người dùng reset cặp khoá định kỳ, đương nhiên các tin nhắn cũ sẽ không thể giải mã được nữa, tuy nhiên người dùng có thể chọn phương án backup dữ liệu về local hoặc người dùng có thể mã hoá toàn bộ tin nhắn cũ với cặp key mới và chuyển về cho server, quá trình này sẽ khá mất thời gian.
- Sử dụng Session key



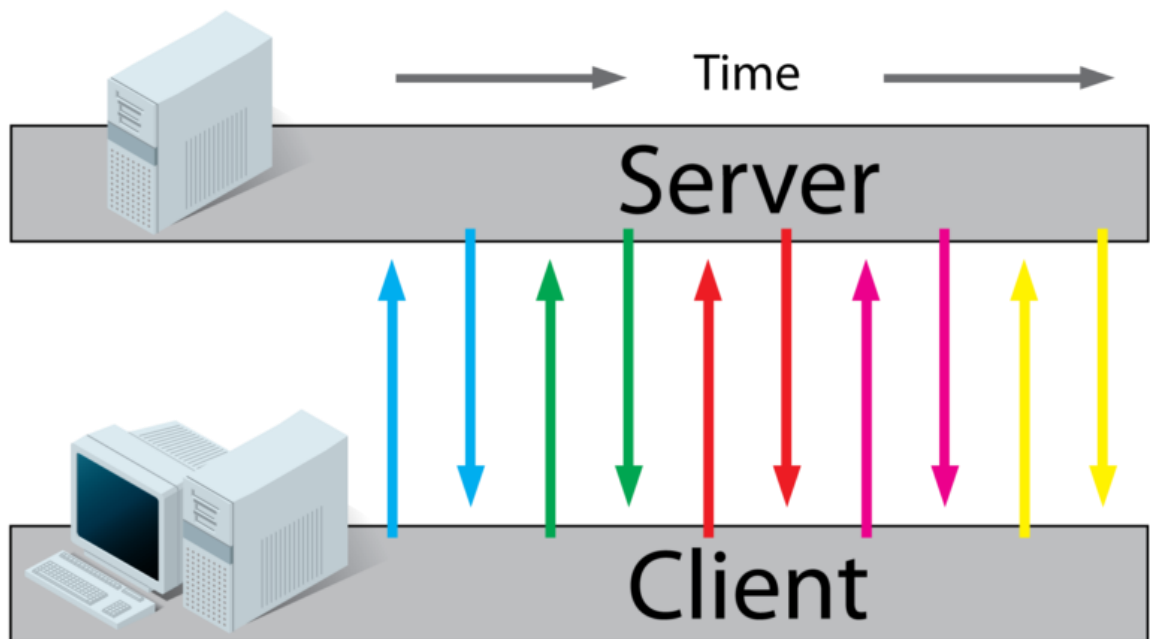
- Sử dụng kỹ thuật trao đổi khoá Diffie–Hellman

### Chương 3:

## TỔNG QUAN VỀ HỆ THỐNG CHAT AN TOÀN

### 3.1 Tổng quan về websocket

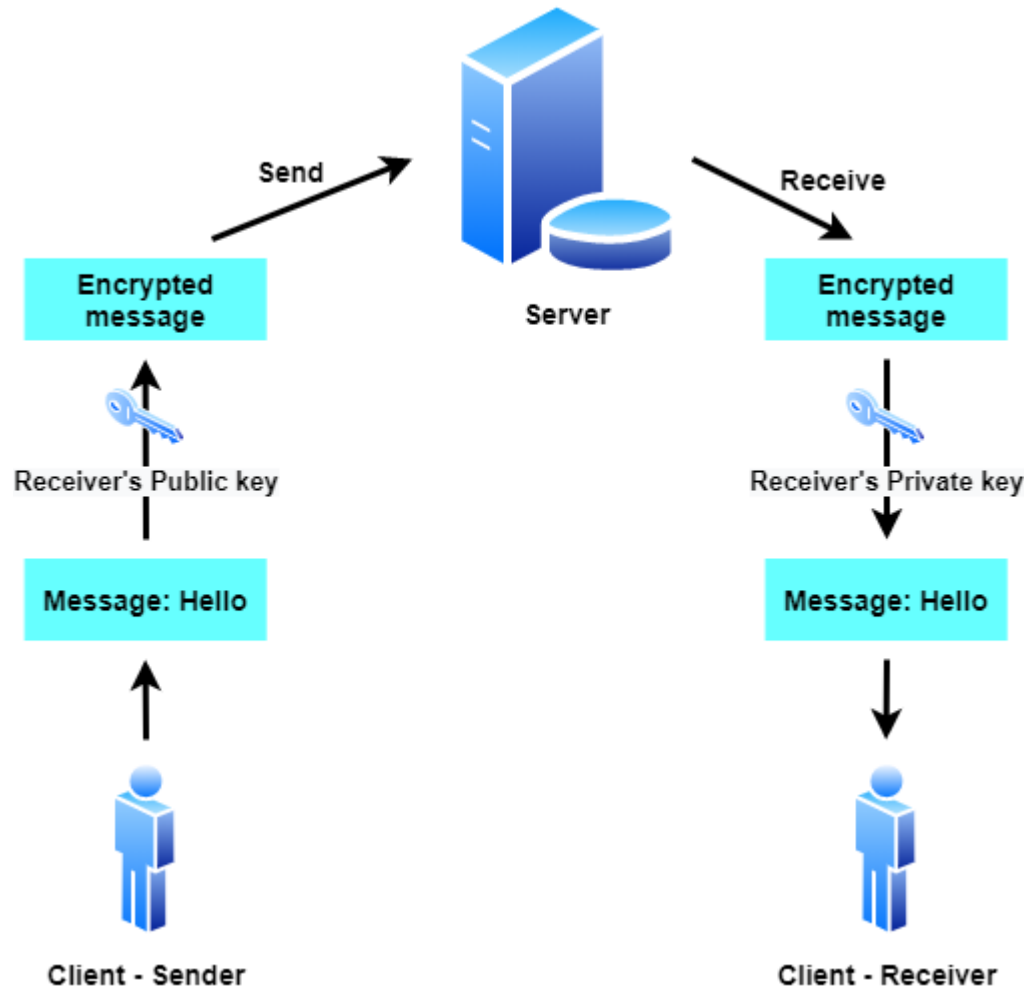
Websocket là giao thức chuẩn cho trao đổi dữ liệu hai chiều giữa client và server. Giao thức WebSocket không chạy trên HTTP, thay vào đó nó thực hiện trên giao thức TCP.



*Hình 3.1 Mô hình WebSocket*

Người ta thường dùng WebSocket thay vì HTTP cho những trường hợp yêu cầu real time (thời gian thực). Ví dụ ta muốn hiển thị biểu đồ, chỉ số chứng khoán, web chat... thì không thể gửi lệnh AJAX liên tiếp tới server để lấy dữ liệu mới rồi cập nhật lên màn hình, như thế sẽ tốn nhiều tài nguyên, traffic.

### 3.2 Hệ thống chat an toàn

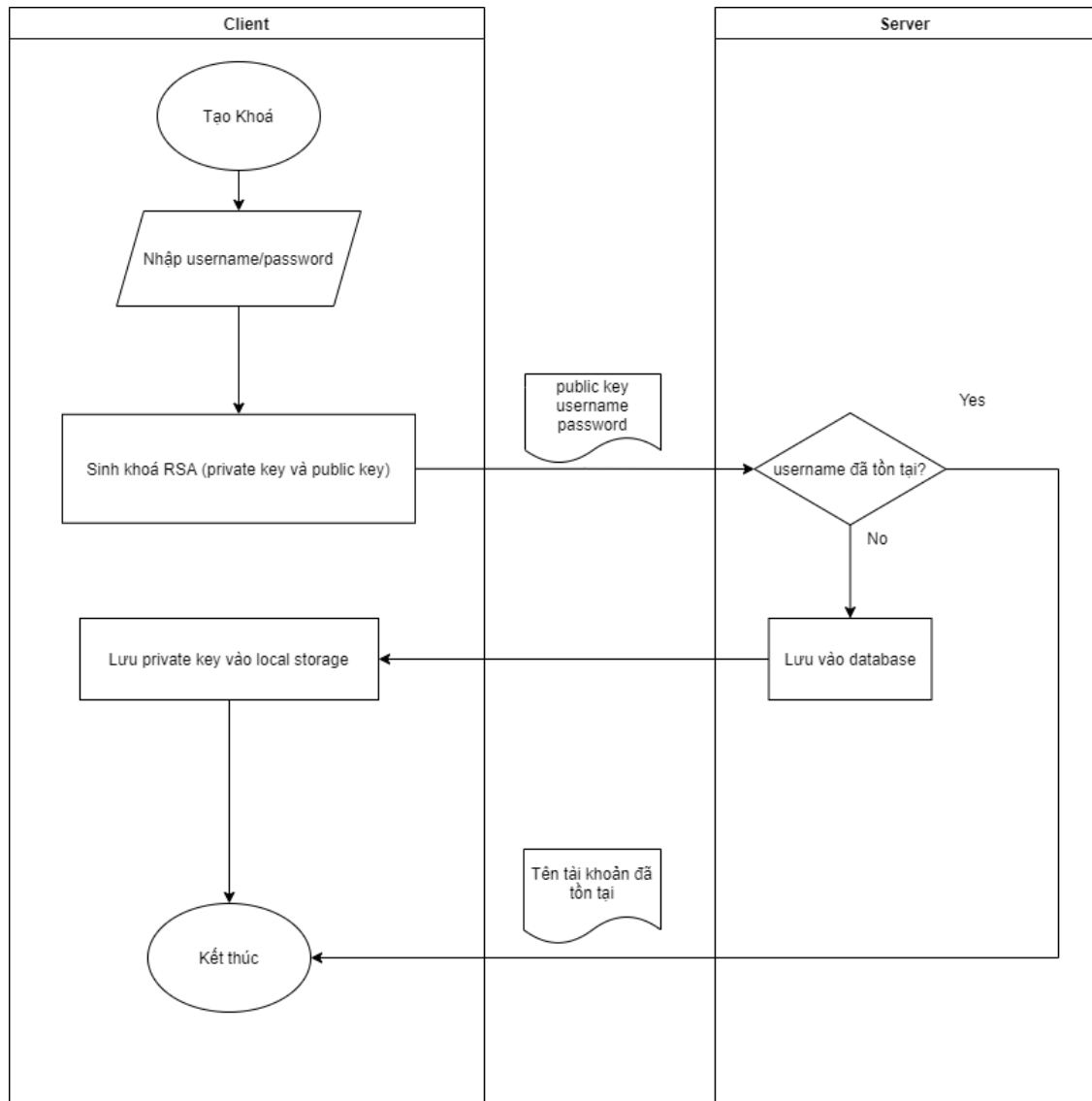


*Hình 3.2 Hệ thống chat an toàn*

Trong hệ thống, mỗi user sẽ là một client, giao tiếp với nhau bằng websocket thông qua server. Mỗi khi client tạo tài khoản, private key và public key sẽ được sinh ở chính máy local client đó và chỉ có public key được đẩy lên server để lưu lại. Khi sender muốn nhắn tin cho receiver thì sẽ sử dụng public key của receiver lấy từ server để mã hóa tin nhắn. Tin nhắn được mã hóa sẽ gửi lên server, đồng thời server gửi trả receiver. Chỉ có

receiver mới có thể hiểu được nội dung tin nhắn này vì chỉ có máy local của receiver mới có chìa khóa giải mã. Như vậy tin nhắn sẽ được an toàn.

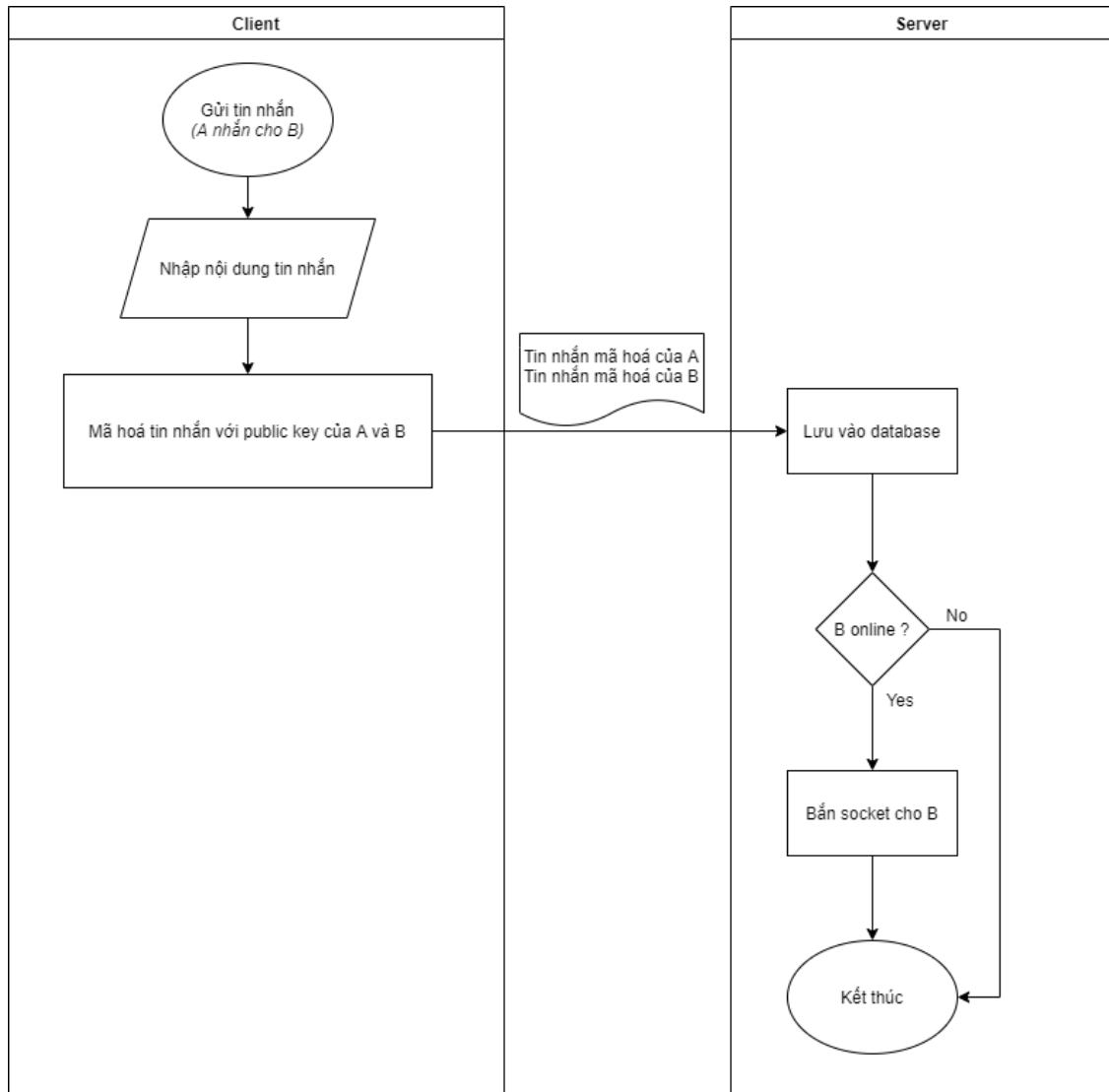
### 3.2.1 Quy trình tạo khoá



Hình 3.2.1 Sơ đồ quy trình tạo khoá

Các khoá RSA được sinh ra ở phía client, public key được gửi cho server lưu lại tương ứng với từng người dùng, để mã hoá các tin nhắn của người dùng này còn private key được phía client lưu lại dùng để giải mã các tin nhắn.

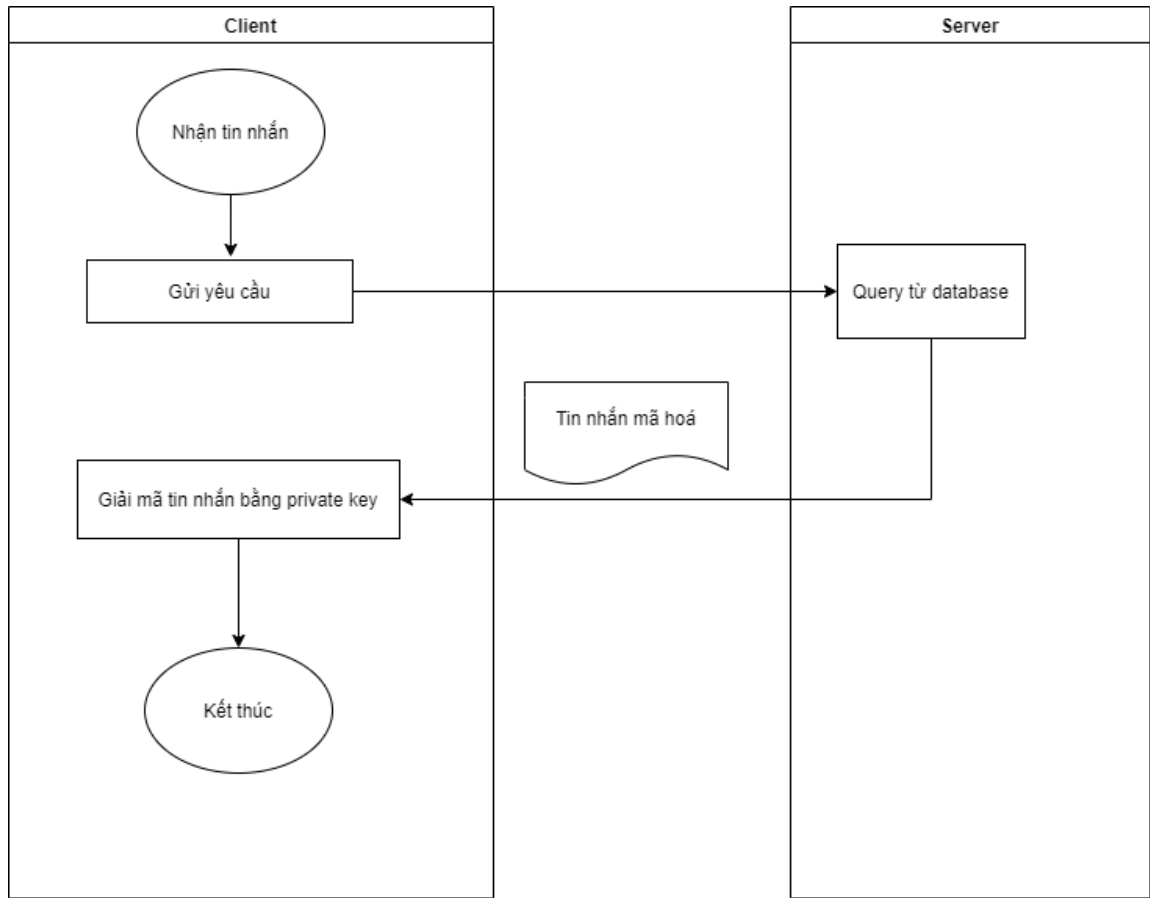
### 3.2.2 Quy trình gửi tin nhắn lưu ở server



Hình 3.2.2 Sơ đồ quy trình gửi tin nhắn

Khi gửi tin nhắn, tin nhắn đó phải được mã hoá bằng các public key của những người dùng sẽ nhận (như ví dụ ở trên là A gửi cho B thì tin nhắn đó được mã hoá bằng public key của cả A và B). Khi tin nhắn đã được mã hoá sẽ được gửi đi cho server rồi lưu vào database, phía server sẽ bắn socket realtime cho người nhận nếu người nhận đang online.

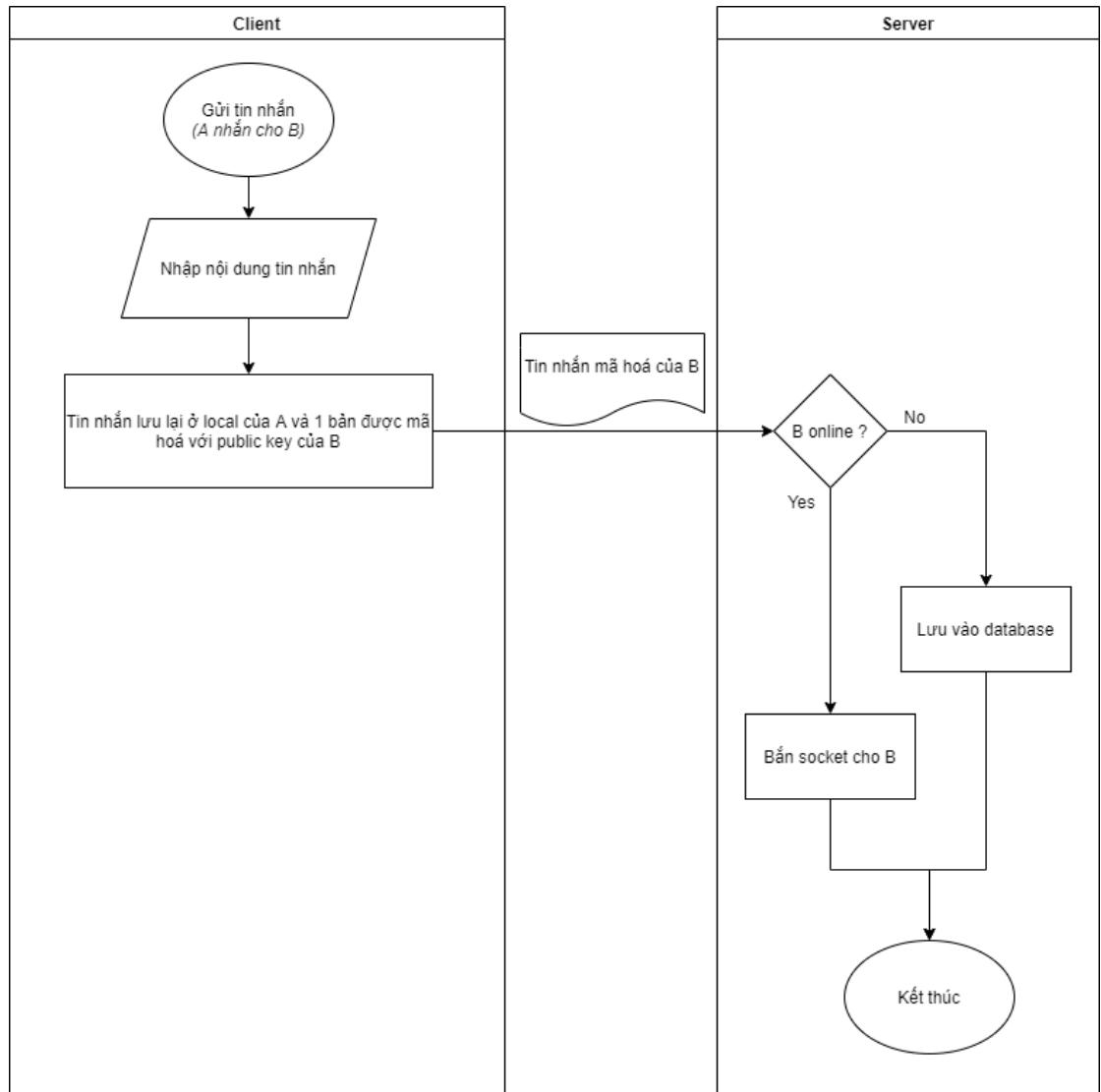
### 3.2.3 Quy trình nhận tin nhắn lưu ở server



*Hình 3.2.3 Sơ đồ quy trình nhận tin nhắn*

Tin nhắn ở dạng mã hoá được nhận từ phía server sẽ được giải mã bằng private key của người dùng.

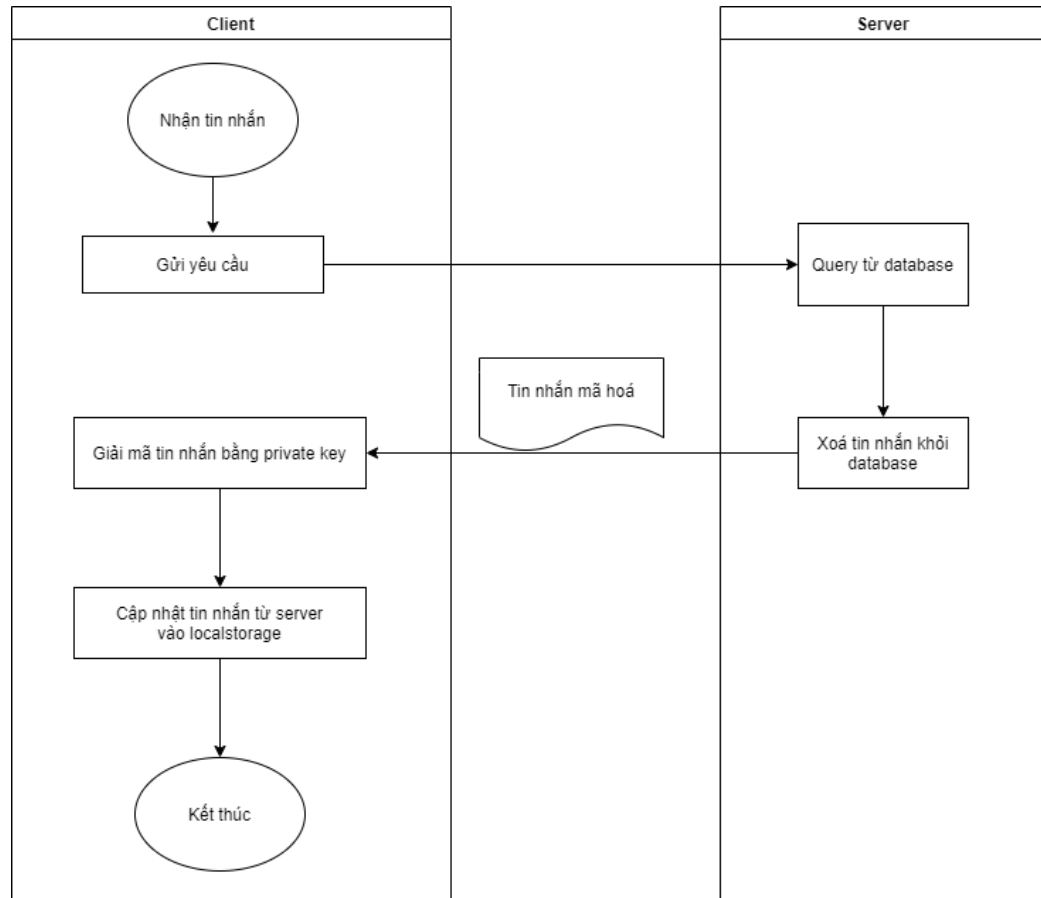
### 3.2.4 Quy trình gửi tin nhắn lưu ở client



*Hình 3.2.4 Sơ đồ quy trình gửi tin nhắn lưu ở client*

Khi gửi tin nhắn, tin nhắn đó được chia làm 2 bản 1 bản được lưu vào local storage và 1 bản được mã hoá bằng các public key của những người dùng sẽ nhận trừ người gửi (như ví dụ ở trên là A gửi cho B thì tin nhắn đó được mã hoá bằng public key của B). Khi tin nhắn đã được mã hoá xong thì gửi đi cho server, phía server sẽ bắn socket realtime cho người nhận nếu người nhận đang online, nếu không sẽ lưu tạm vào database.

### 3.2.5 Quy trình nhận tin nhắn lưu ở client

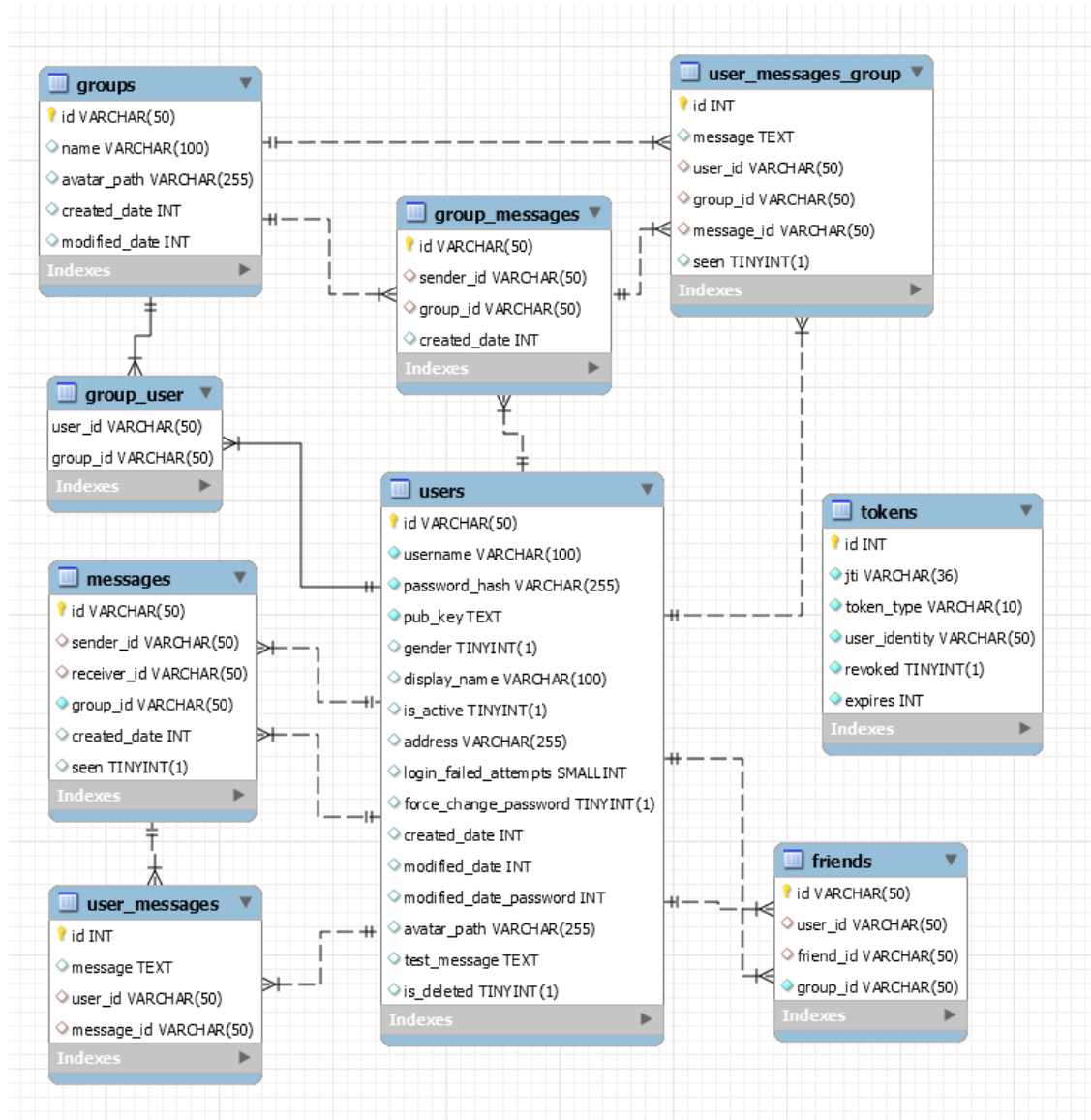


*Hình 3.2.5 Sơ đồ quy trình nhận tin nhắn lưu ở client*

Mỗi lần bật app thì client sẽ gửi yêu cầu đến server để lấy tất cả các tin nhắn trong thời gian offline, khi get dữ liệu thành công thì sẽ bị xoá khỏi database. Các tin nhắn ở dạng mã hoá được nhận từ phía server sẽ được giải mã bằng private key của người dùng rồi cập nhật vào localStorage.

## Chương 4: XÂY DỰNG TRƯỜNG TRÌNH

### 4.1 Mô hình quan hệ database



*Hình 4.1 Mô hình quan hệ database*

Bảng users sẽ lưu lại khoá công khai của từng người dùng, mỗi tin nhắn sẽ được mã hoá bằng khoá công khai của người gửi và những người nhận rồi lưu vào database, khi frontend load tin nhắn thì phải trả đúng tin nhắn được mã hoá bằng khoá công khai tương ứng của người dùng đó.



Ví dụ: A nhắn tin cho B thì đoạn tin nhắn đó phải được mã hoá bằng 2 khoá công khai của A và B rồi lưu 2 đoạn mã đó vào database, khi A load tin nhắn thì backend sẽ trả về đoạn tin nhắn được mã hoá bằng khoá công khai của A và tương tự với B.

## **4.2 Tổng quan về phần mềm**

### **4.2.1 Backend**

Link repo github: <https://github.com/mountain-chan/secure-chat-backend>

Sử dụng Python 3.7

Database server: Mysql 8.0

Các thư viện chính sử dụng:

- Flask
- Flask-sqlalchemy
- Flask-SocketIO

Hỗ trợ build bằng docker

### **4.2.2 Frontend**

Link repo github: <https://github.com/WHKnightZ/RN-Secure-Chat>

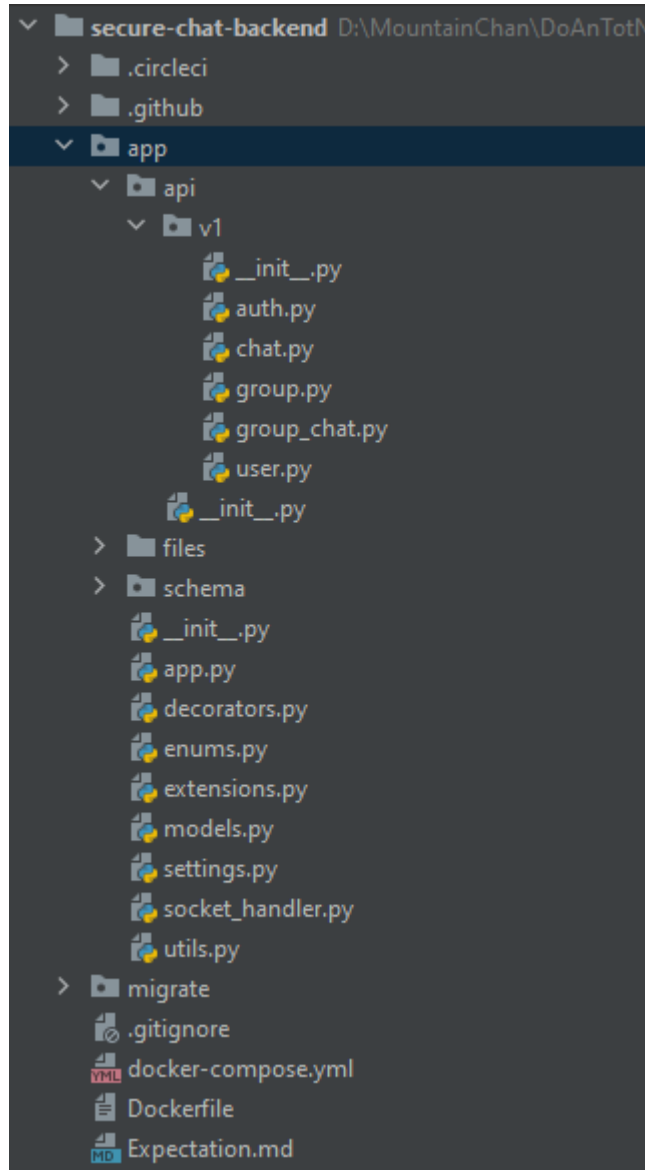
Sử dụng React Native - expo

Các thư viện chính sử dụng:

- expo
- socket.io-client

## 4.3 Giới thiệu về các module của chương trình

### 4.3.1 Backend



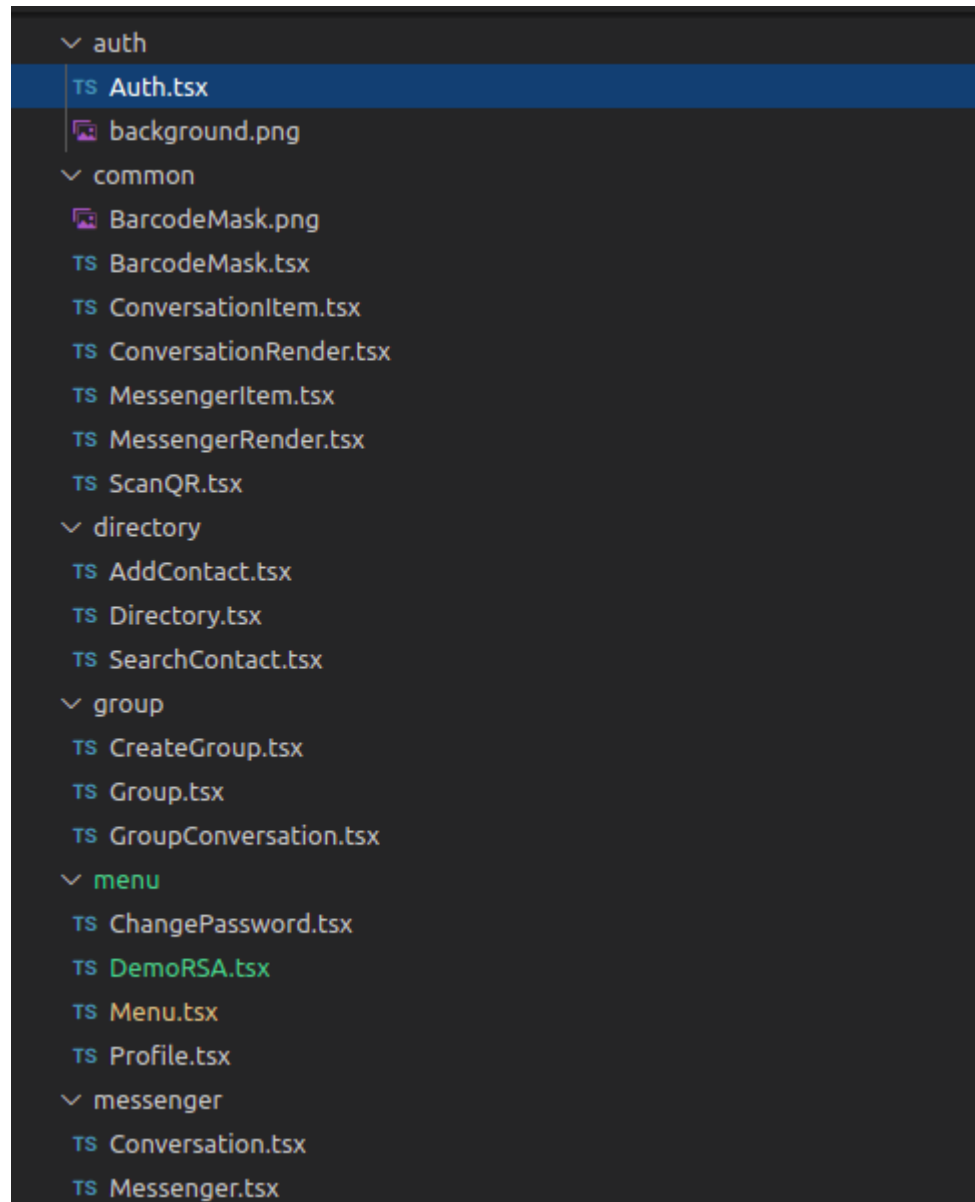
*Hình 4.3.1.1 Kiến trúc project backend*

Backend gồm các services chính sau:

- Auth.py gồm các API quản lý đăng nhập, đăng xuất và làm mới token

- User.py gồm các API quản lý người dùng như thêm, sửa, xoá, cập nhật người dùng, đổi mật khẩu, thêm bạn bè...
- Chat.py bao gồm các API cho việc chat 2 người: Tạo chat mới 2 người, đếm số tin nhắn chưa đọc trong các cuộc hội thoại, bắn socket cho frontend.
- Group.py bao gồm các API quản lý các nhóm chat như: thêm, sửa, xoá nhóm chat và cập nhật thành viên nhóm chat.
- Group\_chat.py bao gồm các API cho việc chat nhóm: Tạo tin nhắn mới theo nhóm, đếm số tin nhắn chưa đọc trong các cuộc hội thoại, bắn socket cho frontend.
- Socket\_handler.py xử lý, lắng nghe và chuyển tiếp các sự kiện socket từ phía client.

### 4.3.2 Frontend



*Hình 4.3.2.1 Kiến trúc project frontend*

Frontend gồm các màn hình chính:

- Auth.tsx: Màn hình đăng nhập
- Messenger: Màn hiển thị các cuộc hội thoại giữa 2 người
- Group: Màn hiển thị các cuộc hội thoại cho nhóm 3 người trở lên
- Directory: Màn hiển thị danh bạ, bạn bè

- Menu: Các chức năng phụ: View Profile, Update thông tin User
- ScanQR: Bật chức năng quét QR để tìm kiếm bạn bè

```
export const registerAction = async (dispatch: any, payload: any) => {
  const { username, password } = payload;

  const bits = 1024;
  // 65537 is commonly used as a public exponent in the RSA cryptosystem.
  // Because it is the Fermat number  $F_n = 2^{(2^n)} + 1$  with  $n = 4$ 
  const exponent = '10001'; // 0x10001 => 65537
  rsa.generate(bits, exponent);
  const { publicKey, privateKey } = getKey();
  const testMessage = rsa.encrypt('SC');

  const response: any = await callApi({
    api: rest.register(),
    method: 'post',
    body: { username, password, pub_key: publicKey, test_message: testMessage },
  });
  const { status } = response;
  if (status) {
    await AsyncStorage.setItem(`${username}-private`, privateKey);
    await loginAction(dispatch, { username, password });
  }
}
```

*Hình 4.3.2.2 Hàm sinh khoá công khai và khoá bí mật*

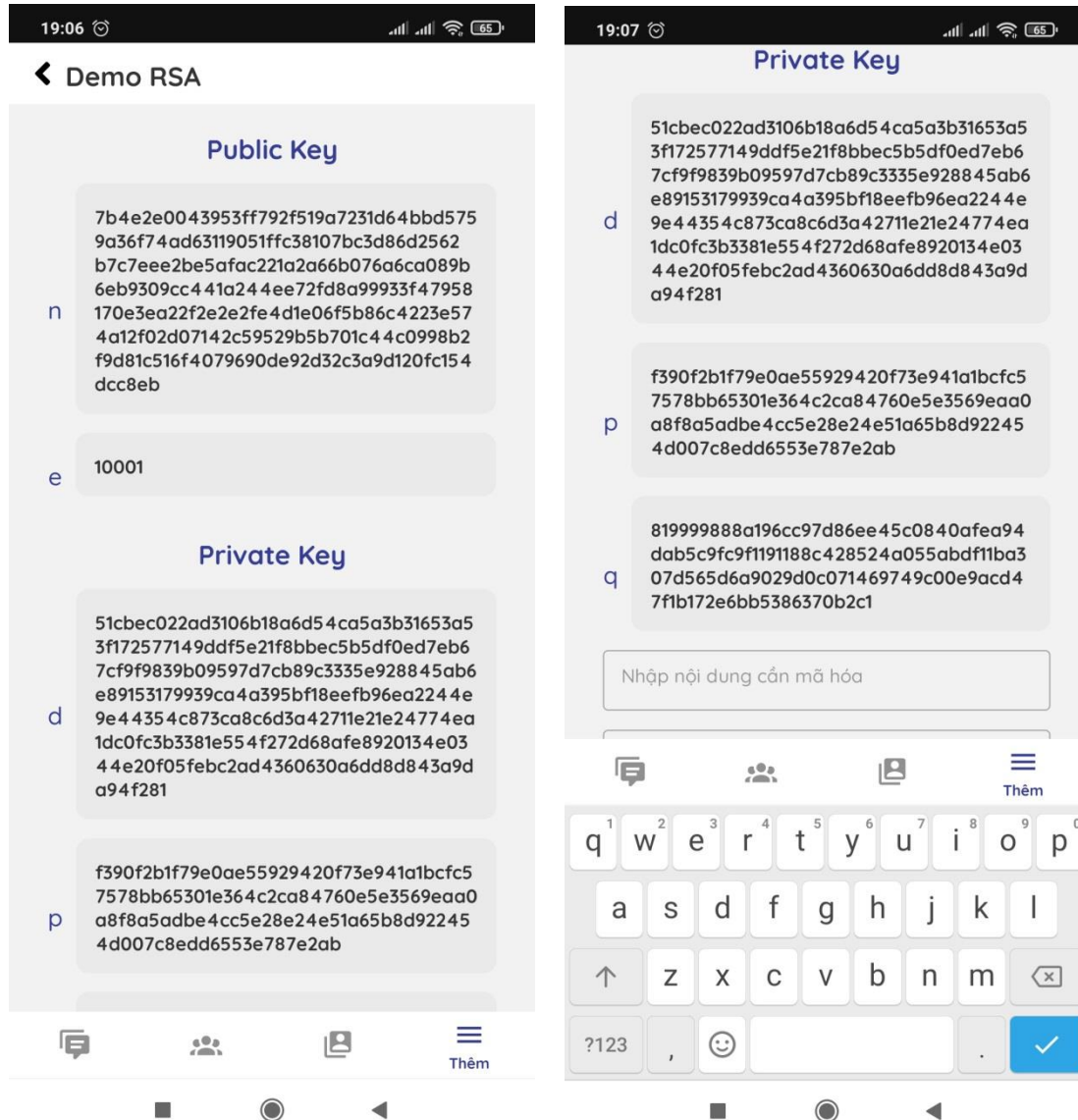
Trong React Native, việc sinh khóa hoàn toàn có thể được rút gọn bằng cách sử dụng các thư viện có sẵn.

Bằng việc áp dụng các thư viện phong phú của JS, ta tiết kiệm được thời gian và công sức trong việc tạo các module mã hóa.

Khi đăng ký tài khoản, khóa sẽ được sinh ra tại máy client, bao gồm 2 thành phần public key và private key, private key sẽ được lưu lại tại local, public key cùng một đoạn demo encrypt message sẽ được gửi về server để phục vụ cho việc mã hóa, giải mã về sau.

## 4.4 Thử nghiệm thực tế

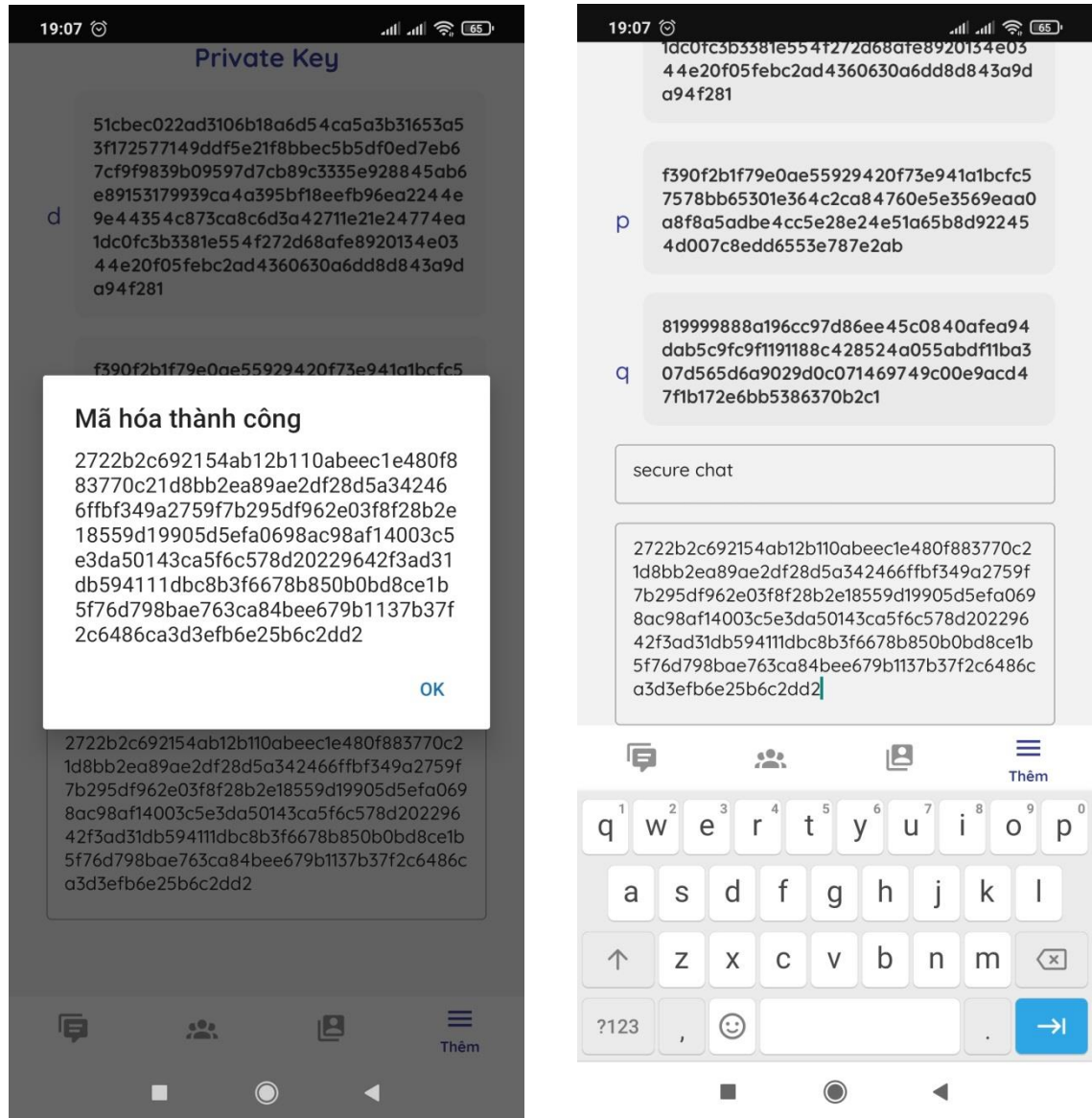
### 4.4.1 Demo mã hoá và giải mã RSA



Hình 4.4.1.1 Màn hình thị private key và public key

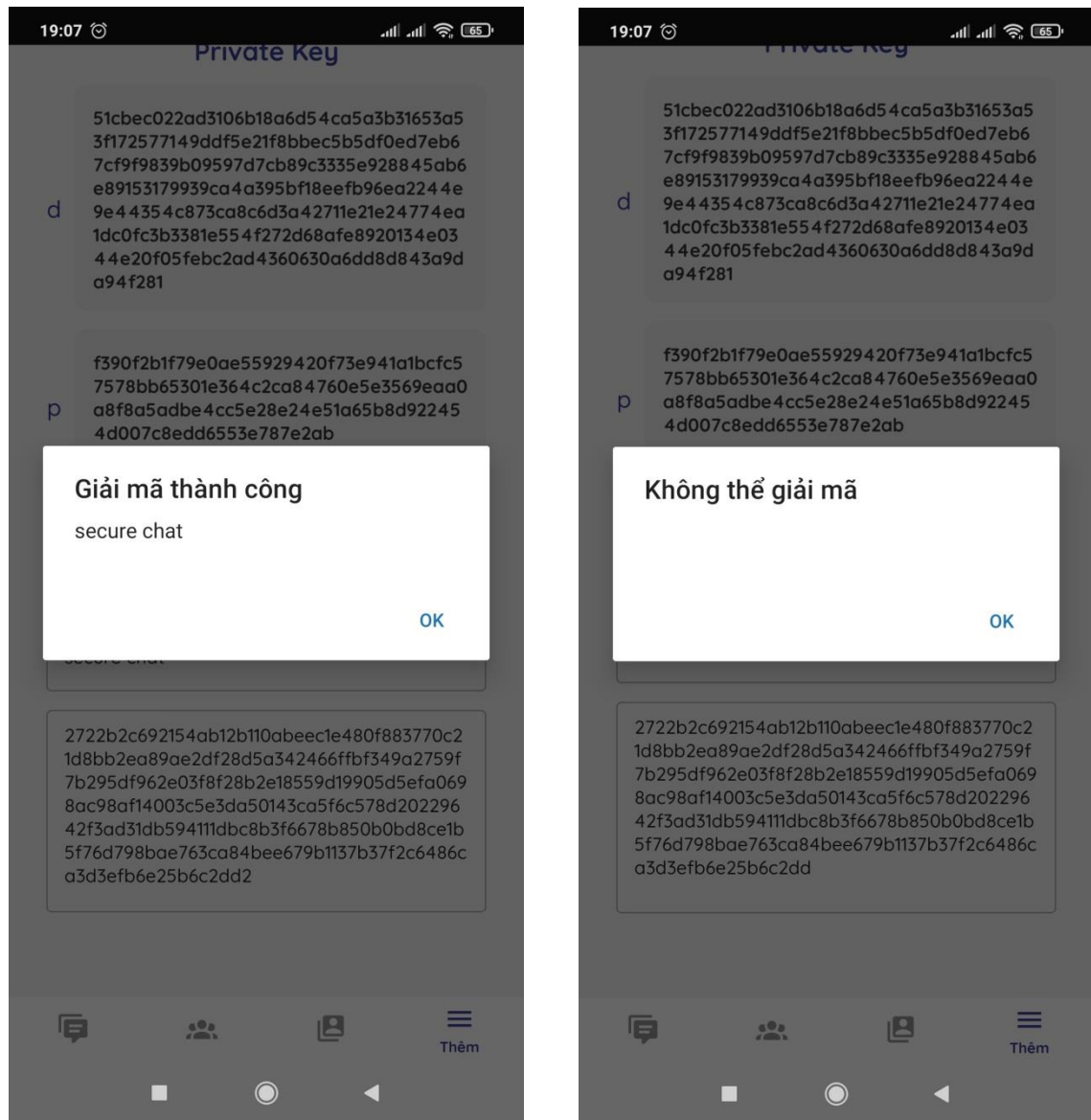
Sau khi đăng nhập, có thể vào mục DemoRSA để test chức năng mã hóa, giải mã tin nhắn RSA.

Ở màn hình này, có thể thấy public key và private key của người dùng hiện tại, đồng thời có thể nhập tin nhắn vào các ô input để xem tin nhắn sau khi mã hóa/giải mã sẽ như thế nào.



Hình 4.4.1.2 Hiện thị chuỗi sau khi đã mã hoá

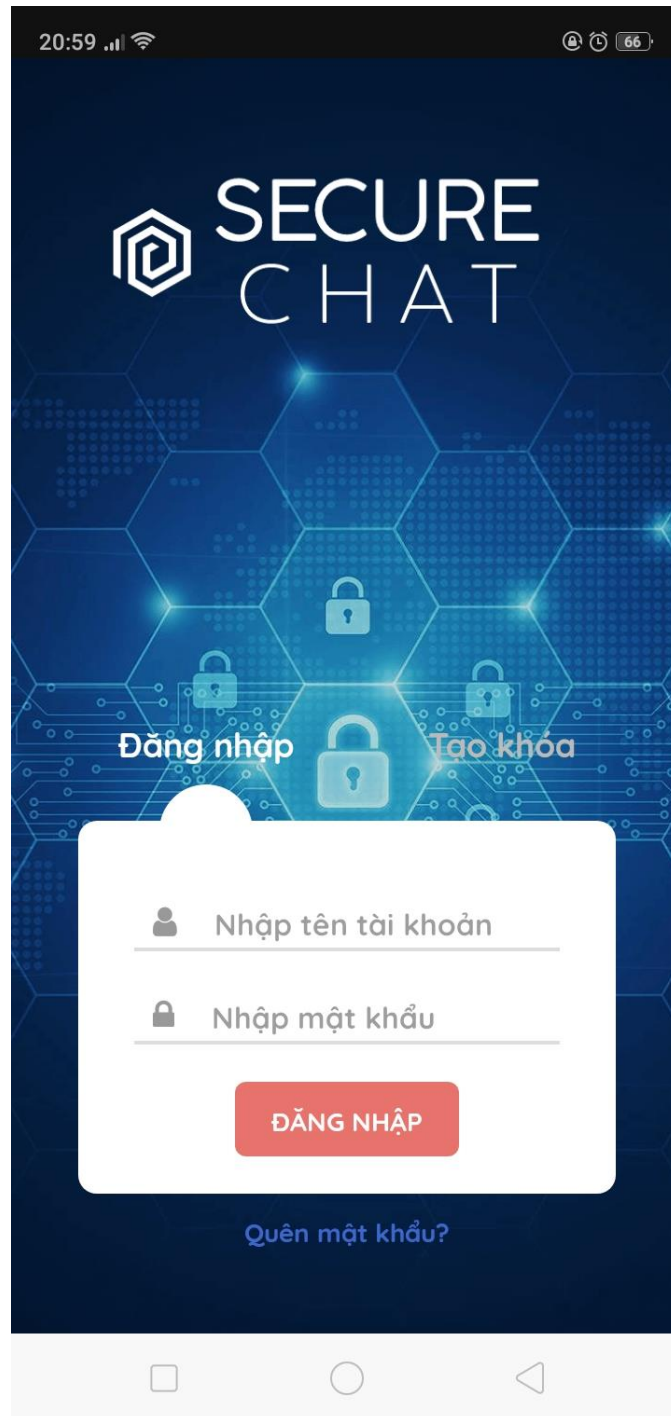
Nếu tin nhắn nhận được mà ko phải được mã hóa từ chính public key của người dùng hiện tại thì đương nhiên sẽ không thể giải mã (bằng private key)



Hình 4.4.1.3 Màn hiển thị kết quả giải mã

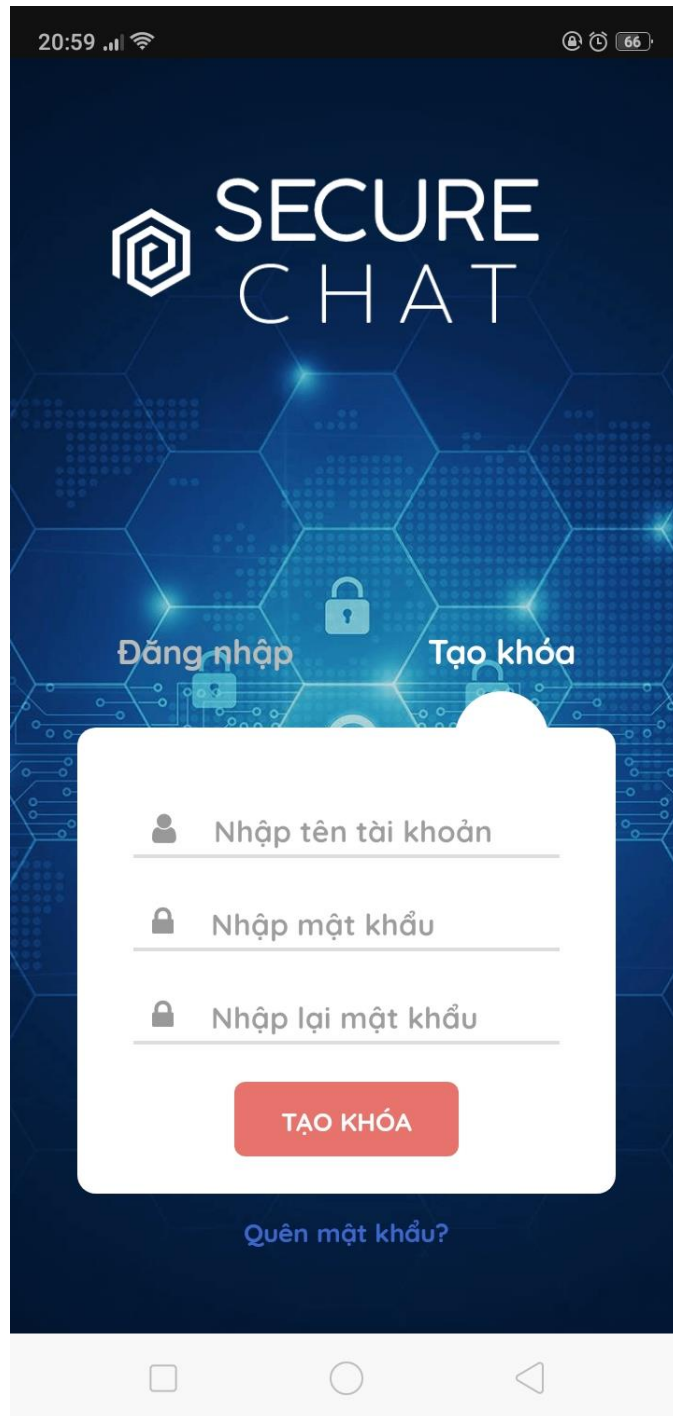


#### 4.4.2 Các màn hình chính của app



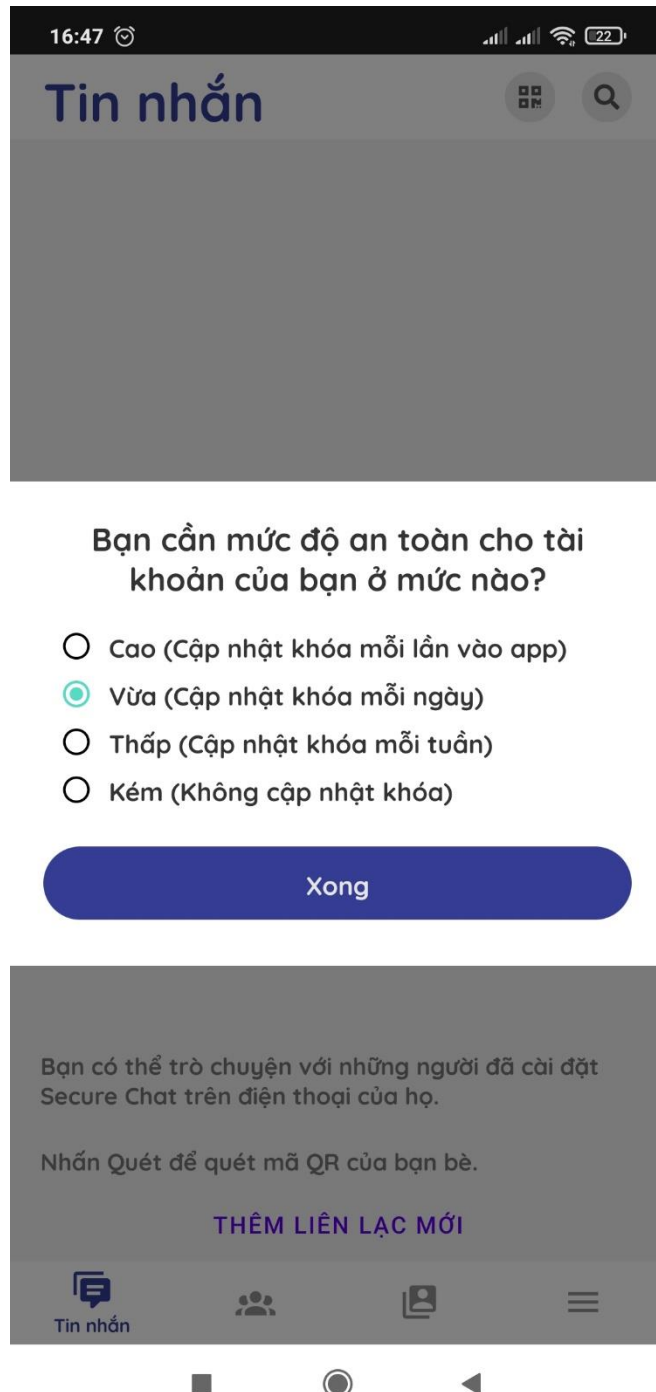
*Hình 4.4.2.1 Màn đăng nhập*

Đây là màn hình đầu tiên cho phép người dùng nhập thông tin tài khoản mật khẩu để đăng nhập vào app với các tài khoản đăng ký trên thiết bị này.



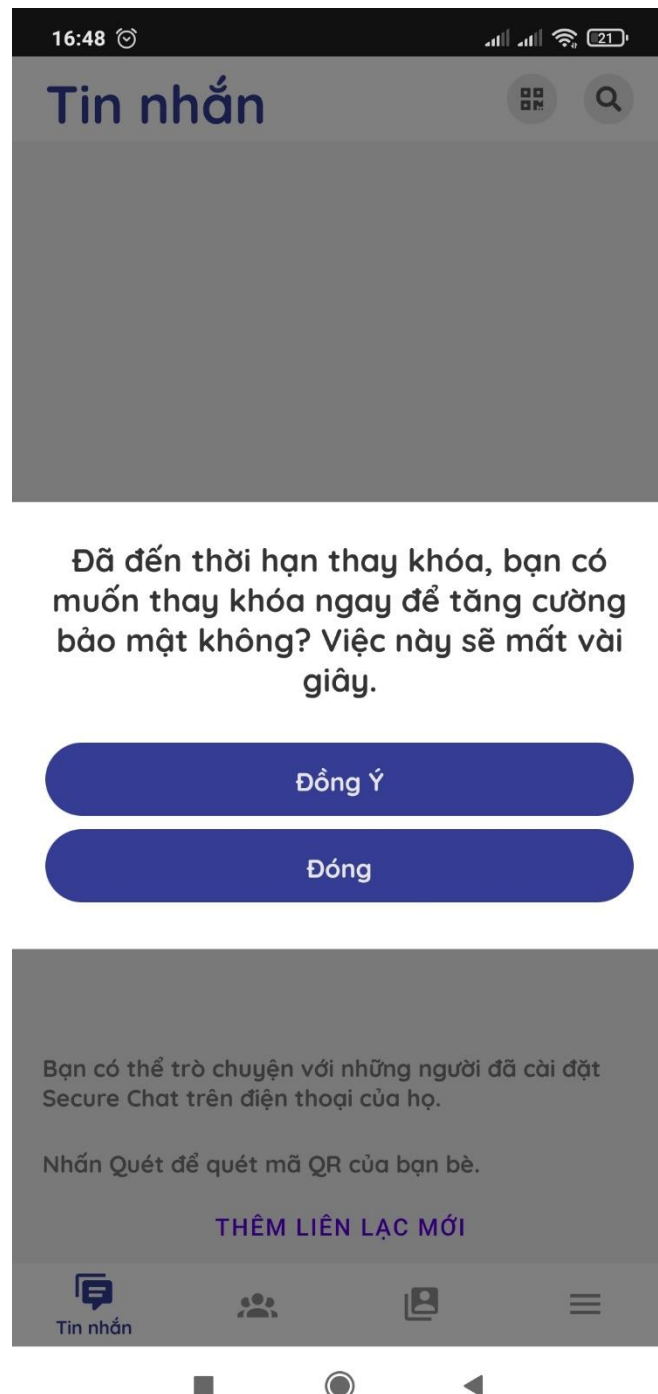
Hình 4.4.2.2 Màn tạo khoá

Màn hình cho phép người dùng đăng ký tài khoản mới, tên tài khoản không được trùng với các tài khoản đã tồn tại, khi ấn tạo khoá các khoá RSA được tạo, public key được gửi cho server và private key được lưu lại phía client.



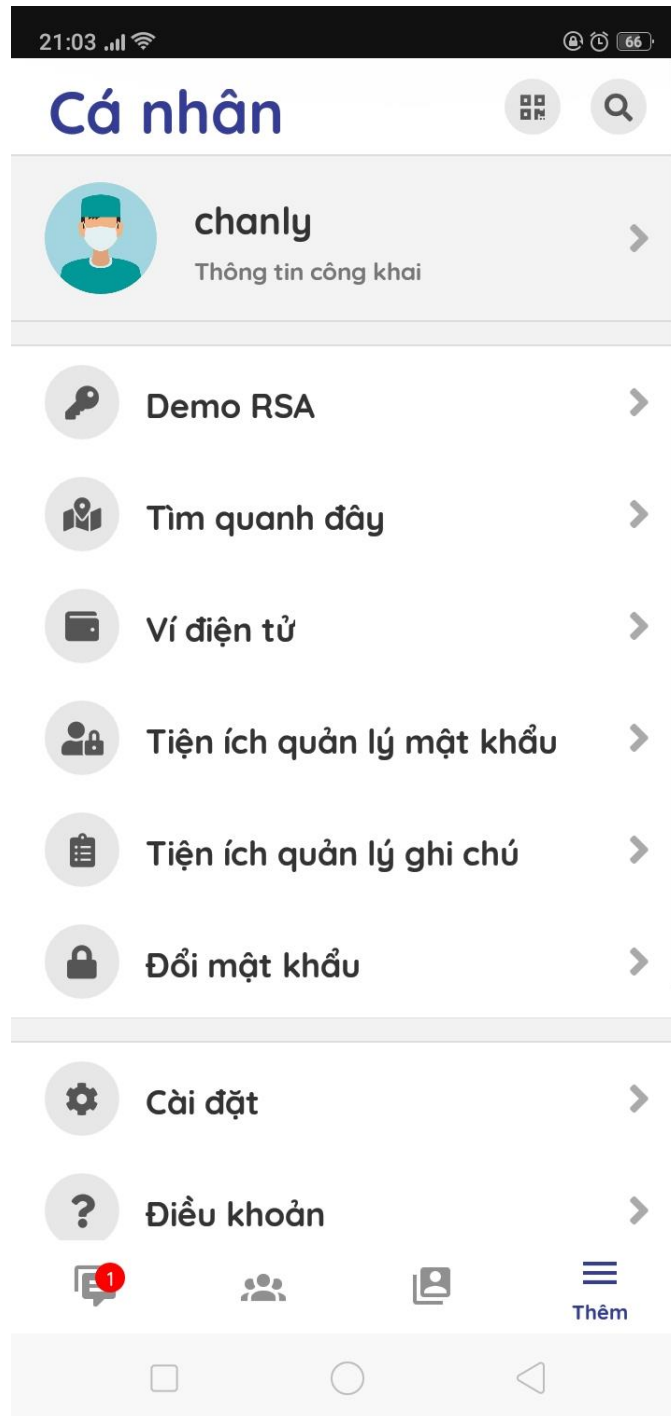
*Hình 4.4.2.3 Màn thiết lập mức độ an toàn*

Sau khi tạo khoá xong sẽ cho phép người dùng lựa chọn 4 cấp độ an toàn cho tài khoản: cao (cập nhật khoá mỗi lần mở app), vừa (cập nhật khoá mỗi ngày), thấp (cập nhật khoá mỗi tuần) và kém (không cập nhật khoá). Người dùng có thể vào chỉnh sửa thiết lập này ở phần cài đặt tài khoản.



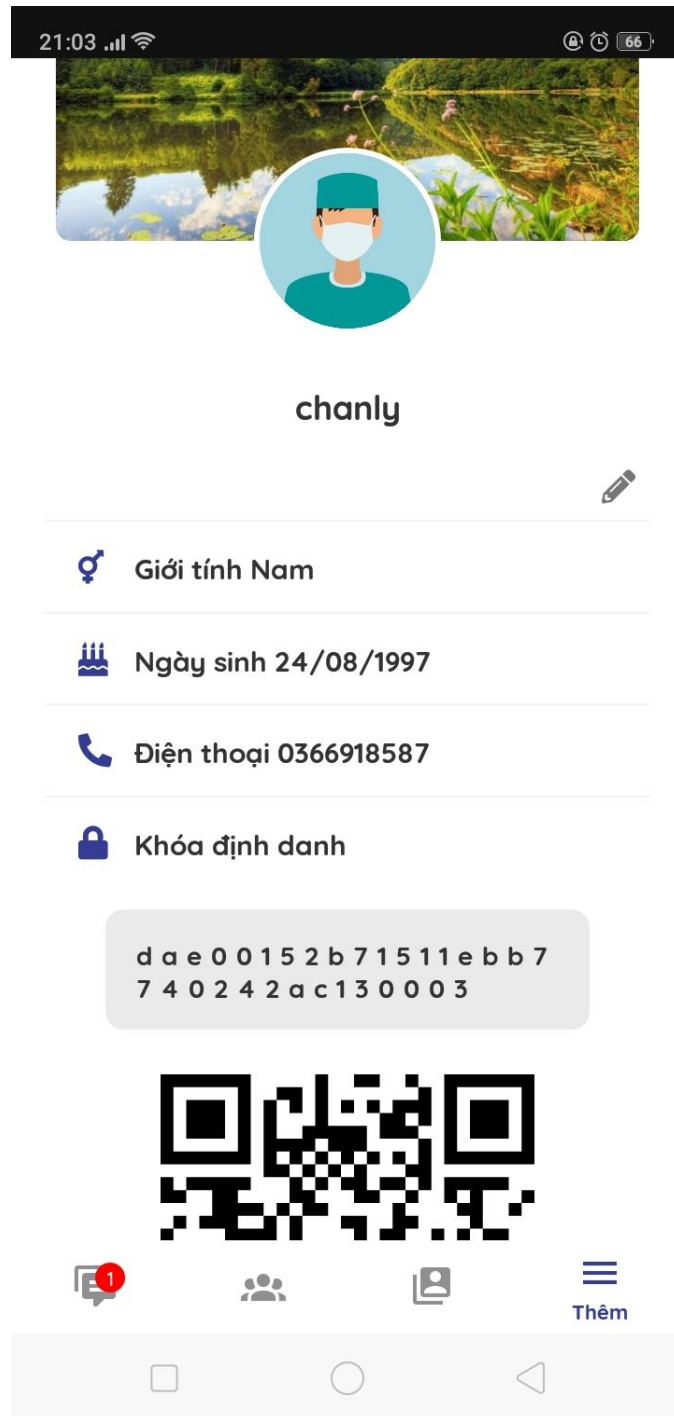
*Hình 4.4.2.4 Màn thông báo cập nhật khoá*

Khi đến thời hạn cập nhật khoá, app sẽ hiển thị thông báo cho người dùng đồng ý thay đổi khoá mới hay không? Nếu đồng ý thì app sẽ tiến hành tạo bộ khoá mới ngược lại sẽ vẫn dùng khoá cũ.



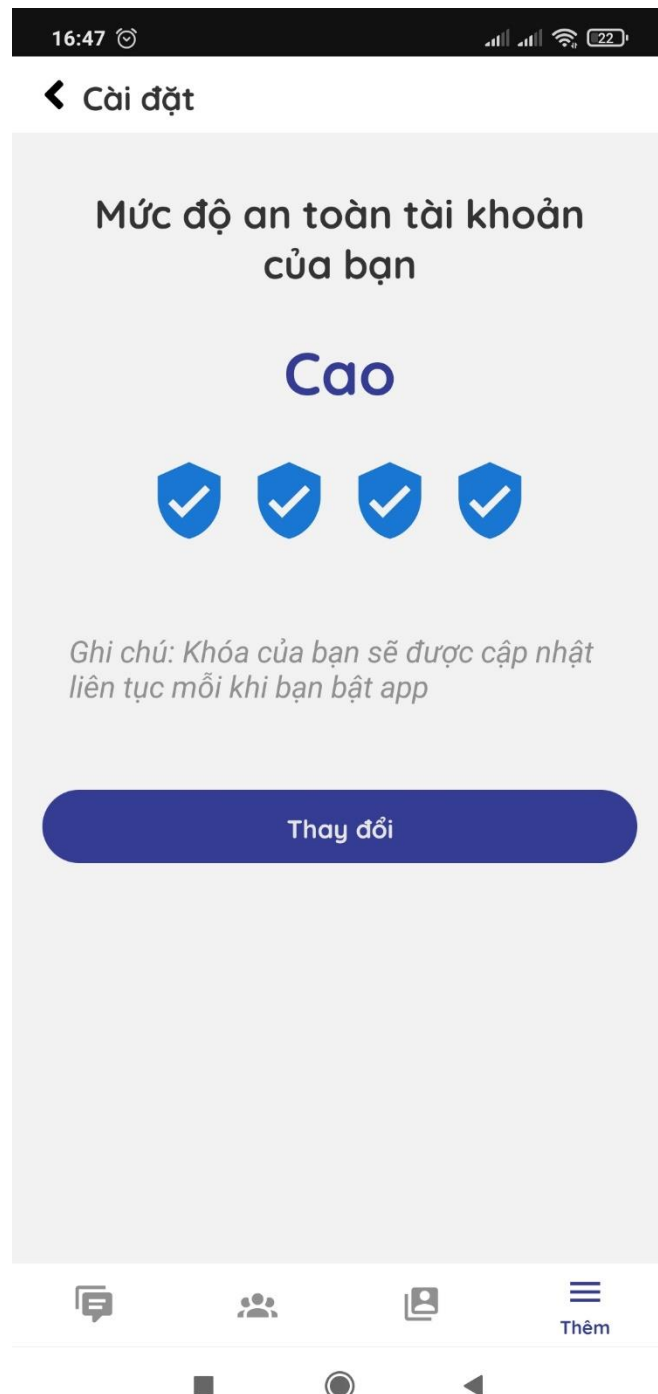
*Hình 4.4.2.5 Màn menu*

Màn menu cho phép người dùng chuyển trang đến các chức năng như: cài đặt, thay đổi mật khẩu, đăng xuất, tìm bạn quanh đây, vào trang cá nhân...



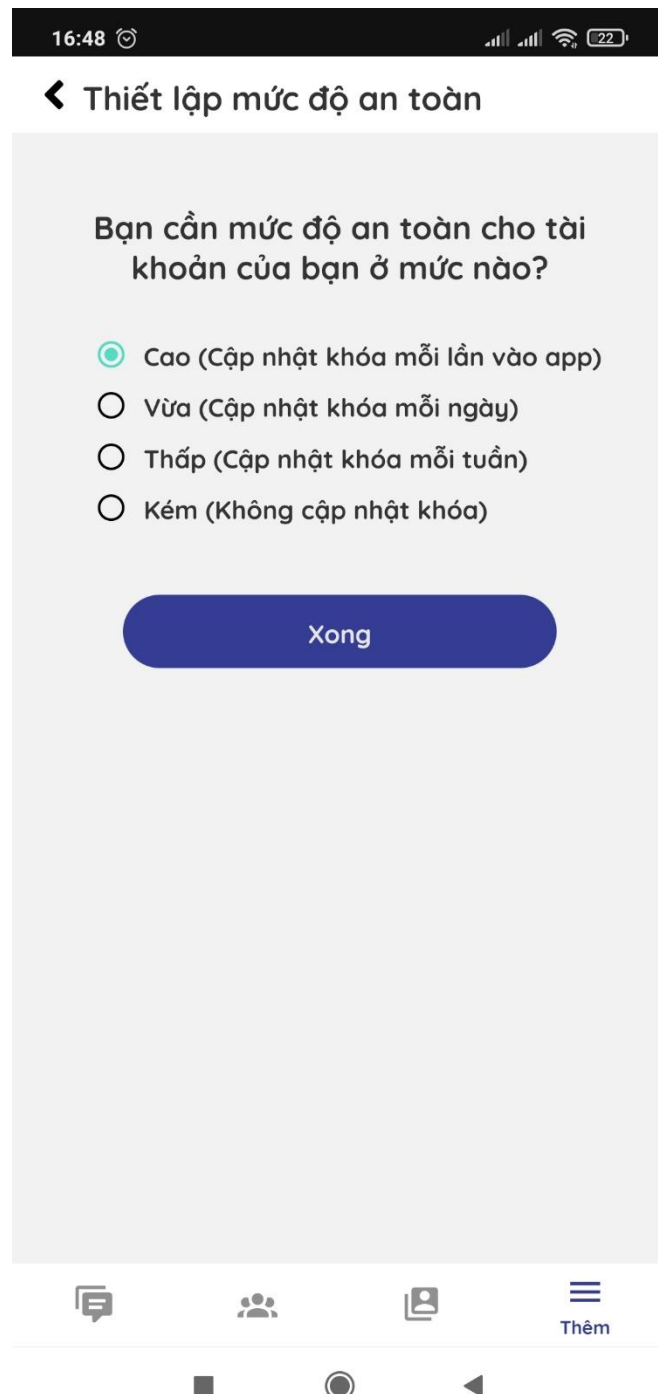
*Hình 4.4.2.6 Màn trang cá nhân*

Màn hình trang cá nhân hiển thị các thông tin các nhân như ngày sinh, số điện thoại, khoá định danh, ảnh đại diện...



*Hình 4.4.2.7 Màn hiển thị mức độ an toàn hiện tại của tài khoản*



Màn sẽ hiển thị mức độ an toàn hiện tại của tài khoản, khi click vào button thay đổi sẽ chuyển sang màn thiết lập mức độ an toàn của tài khoản.






*Hình 4.4.2.8 Màn thiết lập mức độ an toàn*

Màn cho phép người dùng thiết lập lại mức độ an toàn của tài khoản với 4 mức độ đã nêu ở trên.



21:04  


  66

 **Đổi mật khẩu**


## Mật khẩu mới

Mật khẩu mới có thể bao gồm ký tự viết thường, hoa, số và ký tự đặc biệt


Mật khẩu hiện tại







Mật khẩu mới






Nhập lại mật khẩu mới



Đổi mật khẩu

Thêm

*Hình 4.4.2.9 Màn đổi mật khẩu*

Màn đổi mật khẩu cho phép người dùng đổi mật khẩu mới khi cần thiết.



Hình 4.4.2.10 Màn danh sách các cuộc trò chuyện

Màn hiển thị danh sách các cuộc trò chuyện 2 người.



*Hình 4.4.2.11 Màn chat 2 người*

Màn hiển thị các tin nhắn trò chuyện của 2 người sau khi các tin nhắn đã được giải mã ở phía client.



*Hình 4.4.2.12 Màn chat nhóm*

Màn hiển thị các tin nhắn trò chuyện của nhóm sau khi các tin nhắn đã được giải mã.

## KẾT LUẬN

### 1. Kết luận

Những kết quả nghiên cứu và những đóng góp cụ thể trong quá trình làm đồ án đã đạt được như sau:

- Đã tìm hiểu các loại thuật toán mã hóa.
- Xây dựng được ứng dụng chat an toàn

Tuy nhiên đề tài vẫn còn tồn tại một số điểm hạn chế, khó khăn như:

Kích thước mã hóa dữ liệu có hạn chế

Các tin nhắn

### 2. Hướng phát triển

Hướng phát triển của đề tài:

Tiếp tục tìm hiểu nghiên cứu các phương pháp mã hóa khác để nâng cao về độ bảo mật và tốc độ mã hóa và giải mã dữ liệu cũng như là tăng kích thước mã hóa dữ liệu.

Phát triển ứng dụng cả trên pc và mobile.