

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG ĐIỆN – ĐIỆN TỬ

ĐỒ ÁN TỐT NGHIỆP

**Ứng dụng lý thuyết hỗn loạn vào mã hoá đầu
cuối trong kỹ thuật truyền thông**

NGUYỄN VĂN LONG
long.nv182665@sis.hust.edu.vn

Giảng viên hướng dẫn: PGS. TS. Đỗ Trọng Tuấn

Khoa: Kỹ thuật truyền thông

Chữ ký của GVHD

Hà Nội, 08-2022

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG ĐIỆN – ĐIỆN TỬ

**ĐÁNH GIÁ ĐỒ ÁN TỐT NGHIỆP
(DÀNH CHO CÁN BỘ HƯỚNG DẪN)**

Tên đề tài: Ứng dụng lý thuyết hỗn loạn vào mã hoá đầu cuối trong kỹ thuật truyền thông

Họ tên SV: Nguyễn Văn Long

MSSV: 20182665

Cán bộ hướng dẫn: PGS. TS. Đỗ Trọng Tuấn

STT	Tiêu chí (Điểm tối đa)	Hướng dẫn đánh giá tiêu chí	Điểm tiêu chí
1	Thái độ làm việc (2,5 điểm)	Nghiêm túc, tích cực và chủ động trong quá trình làm ĐATN	
		Hoàn thành đầy đủ và đúng tiến độ các nội dung được GVHD giao	
2	Kỹ năng viết quyển ĐATN (2 điểm)	Trình bày đúng mẫu quy định, bố cục các chương logic và hợp lý: Bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến trong đồ án, có căn lề, dấu cách sau dấu chấm, dấu phẩy, có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn, v.v.	
		Kỹ năng diễn đạt, phân tích, giải thích, lập luận: Cấu trúc câu rõ ràng, văn phong khoa học, lập luận logic và có cơ sở, thuật ngữ chuyên ngành phù hợp, v.v.	
3	Nội dung và kết quả đạt được (5 điểm)	Nêu rõ tính cấp thiết, ý nghĩa khoa học và thực tiễn của đề tài, các vấn đề và các giả thuyết, phạm vi ứng dụng của đề tài. Thực hiện đầy đủ quy trình nghiên cứu: Đặt vấn đề, mục tiêu đề ra, phương pháp nghiên cứu/ giải quyết vấn đề, kết quả đạt được, đánh giá và kết luận.	
		Nội dung và kết quả được trình bày một cách logic và hợp lý, được phân tích và đánh giá thỏa đáng. Biện luận phân tích kết quả mô phỏng/ phần mềm/ thực nghiệm, so sánh kết quả đạt được với kết quả trước đó có liên quan.	
		Chỉ rõ phù hợp giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai. Hàm lượng khoa học/ độ phức tạp cao, có tính mới/tính sáng tạo trong nội dung và kết quả đồ án.	
4	Điểm thành tích (1 điểm)	Có bài báo KH được đăng hoặc chấp nhận đăng/ đạt giải SV NCKH giải 3 cấp Trường trở lên/ Các giải thưởng khoa học trong nước, quốc tế từ giải 3 trở lên/ Có đăng ký bằng phát minh sáng chế. (1 điểm)	
		Được báo cáo tại hội đồng cấp Trường trong hội nghị SV NCKH nhưng không đạt giải từ giải 3 trở lên/ Đạt giải khuyến khích trong cuộc thi khoa học trong nước, quốc tế/ Kết quả đồ án là sản phẩm ứng dụng có tính hoàn thiện cao, yêu cầu khối lượng thực hiện lớn. (0,5 điểm)	
		Điểm tổng các tiêu chí:	
		Điểm hướng dẫn:	

Cán bộ hướng dẫn

(Ký và ghi rõ họ tên)

Điểm từng tiêu chí cho lẻ đến 0,5. Nếu Điểm tổng các tiêu chí > 10 thì Điểm hướng dẫn làm tròn thành 10

ĐÁNH GIÁ ĐỒ ÁN TỐT NGHIỆP (DÀNH CHO CÁN BỘ PHẢN BIỆN)

Tên đề tài: Ứng dụng lý thuyết hỗn loạn vào mã hoá đầu cuối trong kỹ thuật truyền thông

Họ tên SV: Nguyễn Văn Long

MSSV: 20182665

Cán bộ phản biện:

STT	Tiêu chí (Điểm tối đa)	Hướng dẫn đánh giá tiêu chí	Điểm tiêu chí
1	Trình bày quyển ĐATN (4 điểm)	Đồ án trình bày đúng mẫu quy định, bố cục các chương logic và hợp lý: Bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến trong đồ án, có căn lề, dấu cách sau dấu chấm, dấu phẩy, có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn, v.v.	
		Kỹ năng diễn đạt, phân tích, giải thích, lập luận: Cấu trúc câu rõ ràng, văn phong khoa học, lập luận logic và có cơ sở, thuật ngữ chuyên ngành phù hợp, v.v.	
2	Nội dung và kết quả đạt được (5,5 điểm)	Nêu rõ tính cấp thiết, ý nghĩa khoa học và thực tiễn của đề tài, các vấn đề và các giả thuyết, phạm vi ứng dụng của đề tài. Thực hiện đầy đủ quy trình nghiên cứu: Đặt vấn đề, mục tiêu đề ra, phương pháp nghiên cứu/ giải quyết vấn đề, kết quả đạt được, đánh giá và kết luận.	
		Nội dung và kết quả được trình bày một cách logic và hợp lý, được phân tích và đánh giá thỏa đáng. Biện luận phân tích kết quả mô phỏng/ phần mềm/ thực nghiệm, so sánh kết quả đạt được với kết quả trước đó có liên quan.	
		Chỉ rõ phù hợp giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai. Hàm lượng khoa học/ độ phức tạp cao, có tính mới/tính sáng tạo trong nội dung và kết quả đồ án.	
3	Điểm thành tích (1 điểm)	Có bài báo KH được đăng hoặc chấp nhận đăng/ đạt giải SV NCKH giải 3 cấp Trường trở lên/ Các giải thưởng khoa học trong nước, quốc tế từ giải 3 trở lên/ Có đăng ký bằng phát minh sáng chế. (1 điểm)	
		Được báo cáo tại hội đồng cấp Trường trong hội nghị SV NCKH nhưng không đạt giải từ giải 3 trở lên/ Đạt giải khuyến khích trong cuộc thi khoa học trong nước, quốc tế/ Kết quả đồ án là sản phẩm ứng dụng có tính hoàn thiện cao, yêu cầu khối lượng thực hiện lớn. (0,5 điểm)	
		Điểm tổng các tiêu chí:	
		Điểm phản biện:	

Cán bộ phản biện

(Ký và ghi rõ họ tên)

ĐÁNH GIÁ ĐỒ ÁN TỐT NGHIỆP
(DÀNH CHO CÁN BỘ THÀNH VIÊN HỘI ĐỒNG)

Tên đề tài: Ứng dụng lý thuyết hỗn loạn vào mã hoá đầu cuối trong kỹ thuật truyền thông

Họ tên SV: Nguyễn Văn Long

MSSV: 20182665

Cán bộ thành viên HĐ:

STT	Tiêu chí (Điểm tối đa)	Hướng dẫn đánh giá tiêu chí	Điểm tiêu chí
1	Chất lượng slides/Bản vẽ kỹ thuật (1,5 điểm)	Sử dụng các minh họa hỗ trợ: Hình ảnh, biểu đồ rõ nét và phù hợp, dễ hiểu	
		Không quá nhiều từ, biết sử dụng từ khoá; bố cục logic, có đánh số trang	
2	Kỹ năng thuyết trình (1,5 điểm)	Tự tin, làm chủ nội dung trình bày, đúng thời gian quy định	
		Dễ hiểu, dễ theo dõi, lô-gic, lôi cuốn.	
3	Nội dung và kết quả đạt được (4 điểm)	Nêu rõ tính cấp thiết, ý nghĩa khoa học và thực tiễn của đề tài, các vấn đề và các giả thuyết, phạm vi ứng dụng của đề tài. Thực hiện đầy đủ quy trình nghiên cứu: Đặt vấn đề, mục tiêu đề ra, phương pháp nghiên cứu/ giải quyết vấn đề, kết quả đạt được, đánh giá và kết luận.	
		Nội dung và kết quả được trình bày một cách logic và hợp lý, được phân tích và đánh giá thỏa đáng. Biện luận phân tích kết quả mô phỏng/ phần mềm/ thực nghiệm, so sánh kết quả đạt được với kết quả trước đó có liên quan.	
		Chỉ rõ phù hợp giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai. Hàm lượng khoa học/ độ phức tạp cao, có tính mới/ tính sáng tạo trong nội dung và kết quả đồ án.	
4	Trả lời câu hỏi (2,5 điểm)	Trả lời ngắn gọn, chính xác, đi thẳng vào vấn đề của câu hỏi.	
		Nắm vững kiến thức cơ bản liên quan đến lĩnh vực nghiên cứu/ công việc của đồ án.	
5	Điểm thành tích (1 điểm)	Có bài báo KH được đăng hoặc chấp nhận đăng/ đạt giải SV NCKH giải 3 cấp Trường trở lên/ Các giải thưởng khoa học trong nước, quốc tế từ giải 3 trở lên/ Có đăng ký bằng phát minh sáng chế. (1 điểm)	
		Được báo cáo tại hội đồng cấp Trường trong hội nghị SV NCKH nhưng không đạt giải từ giải 3 trở lên/ Đạt giải khuyến khích trong cuộc thi khoa học trong nước, quốc tế/ Kết quả đồ án là sản phẩm ứng dụng có tính hoàn thiện cao, yêu cầu khối lượng thực hiện lớn. (0,5 điểm)	
		Điểm tổng các tiêu chí:	
		Điểm bảo vệ:	

Cán bộ thành viên HĐ

(Ký và ghi rõ họ tên)

Điểm từng tiêu chí cho lẻ đến 0,5. Nếu Điểm tổng các tiêu chí > 10 thì Điểm hướng dẫn làm tròn thành 10

LỜI CẢM ƠN

Để có thể hoàn thành đồ án tốt nghiệp này, em xin gửi lời cảm ơn chân thành tới PGS.TS. Đỗ Trọng Tuấn, giảng viên tại Trường Điện – Điện tử, Đại học Bách Khoa Hà Nội và anh Vũ Văn Mạnh, sinh viên khoá 62 của Trường Điện – Điện tử đã định hướng nghiên cứu, động viên và hướng dẫn tận tình và tạo điều kiện tốt nhất cho em trong suốt quá trình thực hiện đề tài.

Em cũng xin gửi lời cảm ơn tới những thầy cô của trường, đã hướng dẫn, dạy dỗ em trong bốn năm đại học, giúp em có những kiến thức nền tảng vững chắc để hoàn thành đồ án và sau này là tham gia thị trường lao động.

Bên cạnh đó, mình/em xin gửi lời cảm ơn tới ASE Lab và những người bạn, những người anh, người chị trong mà mình/em đã có cơ hội gặp gỡ, làm việc và học tập trong những năm qua để giúp mình/em ngày càng tự tin, trưởng thành hơn.

Cuối cùng, con cảm ơn gia đình, những người luôn bên con, khích lệ con khi con gặp khó khăn. Gia đình luôn là nguồn động lực to lớn để con yên tâm học tập, nỗ lực hết mình.

LỜI NÓI ĐẦU

Với sự phát triển nhanh của công nghệ, hàng loạt ứng dụng được nghiên cứu và đưa vào sử dụng, bảo mật thông tin ngày càng trở lên quan trọng. Các hệ mật được tìm thấy xung quanh chúng ta hàng ngày, khi lên mạng, video call, nhắn tin với nhau, giao dịch ngân hàng... Dữ liệu cần bảo mật ngày càng lớn, và các thiết bị xử lý có khả năng xử lý hạn chế. Mặt khác, lý thuyết hỗn loạn đang được nghiên cứu nhiều năm gần đây, tỏ ra có những ưu điểm so với những thuật toán hiện tại.

Từ những tìm hiểu của mình, với mong muốn vận dụng kiến thức đã được học tại Đại học Bách Khoa Hà Nội, em đã lựa chọn đề tài: **“Ứng dụng mật mã hoá hỗn loạn trong mã hoá đầu cuối trong kỹ thuật truyền thông”** để làm đề tài đồ án tốt nghiệp của mình.

Sau một khoảng thời gian nghiên cứu và thực hiện đề tài, sản phẩm đã có những kết quả đạt được bước đầu. Dù đã rất cố gắng nhưng vì kiến thức còn hạn chế cho nên hệ thống không tránh khỏi những thiếu sót và hạn chế nhất định. Vì vậy, em rất mong nhận được sự góp ý, bổ sung của các thầy cô để đề tài được hoàn thiện hơn.

Em xin chân thành cảm ơn!

LỜI CAM ĐOAN

Tôi là Nguyễn Văn Long, mã số sinh viên 20182665, sinh viên lớp Chương trình tài năng Điện tử truyền thông, khóa 63. Người hướng dẫn là PGS.TS Đỗ Trọng Tuấn. Tôi xin cam đoan toàn bộ nội dung được trình bày trong đề án *Ứng dụng lý thuyết hỗn loạn trong mã hoá đầu cuối trong kỹ thuật truyền thông* là kết quả quá trình tìm hiểu và nghiên cứu của tôi. Các dữ liệu được nêu trong đề án là hoàn toàn trung thực, phản ánh đúng kết quả đo đạc thực tế. Mọi thông tin trích dẫn đều tuân thủ các quy định về sở hữu trí tuệ; các tài liệu tham khảo được liệt kê rõ ràng. Tôi xin chịu hoàn toàn trách nhiệm với những nội dung được viết trong đề án này.

Hà Nội, ngày 10 tháng 8 năm 2022

Người cam đoan

Nguyễn Văn Long

MỤC LỤC

DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT	i
DANH MỤC BẢNG BIỂU.....	i
DANH MỤC HÌNH VẼ.....	ii
TÓM TẮT ĐỒ ÁN.....	iii
MỞ ĐẦU.....	1
CHƯƠNG 1. TỔNG QUAN VỀ MẬT MÃ HOÁ.....	3
1.1 Tổng quan về mật mã hoá.....	3
1.1.1 Khái niệm mật mã học	3
1.1.2 Những nguyên lý của bảo mật thông tin	4
1.1.3 Phân loại các thuật toán mật mã hoá.....	6
1.2 Mã hoá đầu cuối.....	9
1.2.1 Khái niệm.....	9
1.2.2 Cách thức hoạt động	10
1.3 Quản lý và phân phối khoá.....	10
1.3.1 Mật mã đường cong Elliptic (Elliptic Curve Cryptography – ECC)	10
1.3.2 Tính bảo mật	18
1.3.3 So sánh ECC và RSA.....	20
1.3.4 Trao đổi khoá bằng thuật toán Elliptic Curve Diffie-Hellman.....	21
1.4 Hàm hỗn loạn và mật mã hoá hỗn loạn.....	22
1.4.1 Khái niệm lý thuyết hỗn loạn.....	22
1.4.2 Một số tính chất quan trọng	22
1.4.3 Phân loại hệ thống hỗn loạn.....	23
1.4.4 Sự phù hợp của hỗn loạn trong mật mã hoá.....	24
1.4.5 So sánh mật mã thông thường và mật mã hỗn loạn	25
1.4.6 Ứng dụng của hỗn loạn trong mật mã hoá.....	26
1.5 Kết luận chương.....	26
CHƯƠNG 2. THIẾT KẾ HỆ THỐNG MẬT MÃ HOÁ.....	27
2.1 Sơ đồ hệ thống.....	27
2.1.1 Khôi hoán vị.....	28
2.1.2 Khôi thay thế.....	29
2.1.3 Khôi khuếch tán	30
2.2 Hàm hỗn loạn Logistic.....	31
2.3 Quá trình mật mã.....	34
2.4 Quá trình giải mật.....	35
2.5 Quá trình sinh khoá.....	37

2.6 Kết luận chương.....	38
CHƯƠNG 3. TRIỂN KHAI ỨNG DỤNG MÃ HOÁ ĐẦU CUỐI	39
3.1 Phân tích ứng dụng.....	39
3.1.1 Tổng quan ứng dụng.....	39
3.1.2 Phân tích đối tượng người dùng	39
3.1.3 Yêu cầu chức năng	39
3.1.4 Yêu cầu phi chức năng	40
3.2 Thiết kế hệ thống phía máy chủ	40
3.2.1 Xây dựng CSDL	41
3.2.2 Xây dựng API.....	42
3.2.3 Xây dựng Socket	48
3.3 Thiết kế ứng dụng phía máy khách.....	48
3.3.1 Sơ đồ hệ thống.....	48
3.3.2 Cấu trúc ứng dụng	50
3.3.3 Giao diện người dùng	53
3.4 Kết luận chương.....	55
CHƯƠNG 4. KIỂM THỬ VÀ ĐÁNH GIÁ	56
4.1 Kiểm thử và đánh giá khối mã mã hoá	56
4.1.1 Kiểm thử.....	56
4.1.2 Đánh giá hệ mật.....	56
4.2 Kiểm thử các chức năng phía máy chủ.....	57
4.2.1 Kiểm thử trên máy local	57
4.2.2 Kiểm thử khi hệ thống triển khai trên Heroku Server	66
4.3 Kiểm thử các chức năng trên ứng dụng Android.....	66
4.3.1 Kiểm thử chức năng trao đổi khoá	66
4.3.2 Kiểm thử các chức năng liên quan đến đăng nhập, đăng ký	67
4.3.3 Kiểm thử chức năng nhắn tin	68
4.4 Đánh giá	69
4.5 Kết luận chương.....	69
KẾT LUẬN.....	70
TÀI LIỆU THAM KHẢO	71
PHỤ LỤC.....	73
Phụ lục 1. Các phần mềm, framework sử dụng trong đề án	73

DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT

Ký hiệu	Viết đầy đủ	Ý nghĩa
ECC	Elliptic Curve Cryptography	Mật mã đường cong Elliptic
ECDH	Elliptic Curve Diffie–Hellman Key Exchange	Trao đổi khóa Elliptic Curve Diffie – Hellman
ECDLP	Elliptic Curve Discrete Logarithm Problem	Bài toán lôgarit rời rạc đường cong Elliptic
E2EE	End-to-End encryption	Mã hoá đầu cuối
CSDL	Cơ sở dữ liệu (database)	Cơ sở dữ liệu
MVVM	Model View ViewModel	

DANH MỤC BẢNG BIỂU

Bảng 1.1 So sánh kích thước khoá của RSA và ECC cùng mức độ bảo mật (bit) ..	20
Bảng 1.2 So sánh tính chất mật mã và tính chất hỗn loạn	25
Bảng 1.3 So sánh thuật toán lập mật mã chuẩn và mật mã hỗn loạn.....	25
Bảng 2.1 Bảng S-Box sinh ra bởi hàm Logistic với $x_0 = 0.12$ và $\mu = 3.99$	35
Bảng 4.1 Kịch bản kiểm thử khối mật mã hoá	56
Bảng 4.2 Thời gian mật mã và giải mật của hệ mật.....	57
Bảng 4.3 Kịch bản kiểm thử máy chủ.....	57
Bảng 4.4 Bảng kiểm thử chức năng trao đổi khoá.....	66
Bảng 4.5 Bảng kiểm thử chức năng đăng nhập, đăng kí	67
Bảng 4.6 Kiểm thử chức năng nhắn tin	69

DANH MỤC HÌNH VẼ

Hình 1.1 Mô hình truyền tin mật	4
Hình 1.2 Mô hình hệ mật mã hoá đối xứng [4]	7
Hình 1.3 Phân loại mã khối theo cấu trúc.....	8
Hình 1.4 Mô hình hệ mật mã hoá công khai	9
Hình 1.5 Các hình dạng của các đường cong elliptic khác nhau.....	12
Hình 1.6 Các loại điểm kỳ dị.....	12
Hình 1.7 Tổng của 3 điểm thẳng hàng bằng 0	13
Hình 1.8 Một số dạng đồ thị của đường cong trong trường hữu hạn	15
Hình 1.9 Phép nhân của P với 5 điểm khác biệt	16
Hình 1.10 Quá trình trao đổi khoá bằng ECDH	21
Hình 2.1 Quá trình mật mã và giải mật	27
Hình 2.2 Khối hoán vị của hệ mật	28
Hình 2.3 Khối thay thế của hệ mật	29
Hình 2.4 Khối khuếch tán của hệ mật	31
Hình 2.5 Phụ thuộc điều kiện đầu vào của hàm Logistic với $\mu = 3.89$	32
Hình 2.6 Hệ số Lyapunov của hàm Logistic phụ thuộc vào μ	32
Hình 2.7 Lược đồ phân nhánh của hàm Logistic.....	33
Hình 2.8 Sự chuyển động của hàm Logistic trong mặt phẳng pha.....	33
Hình 2.9 Phân bố của chuỗi giá trị được tạo từ hàm Logistic	34
Hình 3.1 Kiến trúc hệ thống	39
Hình 3.2 CSDL của ứng dụng	41
Hình 3.3 Quá trình trao đổi khoá.....	49
Hình 3.4 Quá trình trao đổi tin nhắn.....	49
Hình 3.5 Kiến trúc Clean Architecture	51
Hình 3.6 Cấu trúc của ứng dụng thử nghiệm.....	52
Hình 3.7 Cấu trúc thư mục của project.....	52
Hình 3.8 Module chaotic	53
Hình 3.9 Màn hình đăng nhập, đăng ký của ứng dụng.....	54
Hình 3.10 Màn hình liên quan đến nhắn tin	55
Hình 4.1 Kiến trúc hệ điều hành Android.....	77

TÓM TẮT ĐỒ ÁN

Đồ án nhằm mục đích nghiên cứu về ứng dụng của lý thuyết hỗn loạn trong mật mã hoá. Từ đó, ứng dụng mật mã hoá hỗn loạn vào trong mã hoá đầu cuối với mục đích tăng tính bảo mật của dữ liệu trong quá trình trao đổi thông tin. Cuối cùng, đồ án triển khai một ứng dụng nhắn tin trên hệ điều hành Android có sử dụng những kiến thức đã được nghiên cứu.

Trong đồ án này, em đã tìm hiểu những lý thuyết về mật mã hoá như mã hoá đầu cuối, các kỹ thuật trao đổi khoá, mật mã hoá hỗn loạn, thiết kế một hệ mật sử dụng lý thuyết hỗn loạn. Sau đó, xây dựng một ứng dụng nhắn tin bảo mật sử dụng mã hoá đầu cuối có ứng dụng lý thuyết hỗn loạn. Cuối cùng, em kiểm thử và đánh giá hệ mật, kiểm thử ứng dụng ở máy chủ và ở trên điện thoại Android.

Kết quả của đồ án đạt được những yêu cầu đặt ra: Hệ mật mật mã hoá và giải mật mã hoá đúng; Quá trình trao đổi khoá đúng nên trong quá trình nhắn tin, tin nhắn được mật mã hoá và giải mật mã hoá đúng. Tuy nhiên, quá trình mật mã hoá và giải mật bằng hệ mật hỗn loạn chưa được tối ưu, cần nghiên cứu, phát triển thêm, và nghiên cứu, thực hiện thêm mật mã hoá với ảnh, video, hoặc theo thời gian thực như gọi điện, video call.

ABSTRACT

The thesis is aimed to research the application of chaos theory in cryptography. After that, chaotic cryptography is applied to end-to-end encryption to grow up the security of data in the communication. Finally, the thesis is implemented a messaging application on the android device using the researched knowledge.

In this thesis, I learned about cryptographic theories such as end-to-end encryption, key exchange techniques, chaotic cryptography, and designing a cryptosystem using chaos theory. Then build a secure messaging application using end-to-end encryption that has chaos theory applications. Finally, I test and evaluate the cryptosystem, test the application on the server and on the android phone.

The results of the thesis meet the requirements set forth: The correct encryption and decryption cryptosystem; The key exchange process is correct so during the messaging process, the message is encrypted and decrypted properly. However, the process of encryption and decryption using chaotic cryptosystems has not been optimized, it is necessary to research, develop further, and research and implement more encryption with photos, videos, or in real time such as: call, video call.

MỞ ĐẦU

Trong thời đại công nghệ phát triển nhanh chóng như hiện nay, dữ liệu được lưu trữ, trao đổi rất nhiều. Do đó, vấn đề bảo mật thông tin trong quá trình truyền thông trở nên vô cùng quan trọng. Mặt khác, dữ liệu có kích thước ngày càng lớn, đặc biệt là ảnh và video, cùng với việc các thiết bị dùng để mật mã và giải mật có sức mạnh xử lý hạn chế như smartphone, các thiết bị Internet of Things... Khi đó, các kỹ thuật mật mã phổ biến hiện nay như AES, RSA thể hiện một số hạn chế và điểm yếu trong việc mật mã hoá và giải mật. Đó là thời gian tính toán kéo dài và mức độ tính toán cao cho các dữ liệu có kích thước khổng lồ. Điều này không phù hợp với những ứng dụng cần mã hoá theo thời gian thực như hội nghị từ xa, phát video trực tiếp và nhiều ứng dụng khác cần tính toán rất cao.

Trong khi đó, một số thuật toán dựa trên sự hỗn loạn cung cấp sự kết hợp tốt giữa tốc độ, độ phức tạp bảo mật cao, chi phí tính toán thấp. Hơn nữa, một số thuật toán dựa trên sự hỗn loạn có nhiều đặc điểm quan trọng: phụ thuộc nhạy cảm vào các tham số ban đầu, đặc tính giả ngẫu nhiên, sai lệch, không tuần hoàn. Mật mã hóa dựa trên hỗn loạn là một phương tiện thiết yếu của mã hóa kỹ thuật số hiện đại. Đặc tính thống kê của dữ liệu gốc được biến đổi, do đó làm tăng độ khó của việc phá mã trái phép.

Từ nhu cầu trên, với sự hướng dẫn của thầy Đỗ Trọng Tuấn, em đã lựa chọn đề tài **“Ứng dụng lý thuyết hỗn loạn trong mã hoá đầu cuối trong kỹ thuật truyền thông”**. Trong đề tài này, em đã tìm hiểu về những lý thuyết liên quan đến mật mã hoá như mã hoá đầu cuối, quá trình trao đổi khoá, lý thuyết hỗn loạn trong mật mã hoá. Sau đó, dựa vào những ưu điểm của lý thuyết hỗn loạn, xây dựng thuật toán mật mã và giải mật, triển khai vào trong mã hoá đầu cuối. Cuối cùng xây dựng một ứng dụng nhắn tin sử dụng lý thuyết hỗn loạn trong mã hoá đầu cuối.

Nội dung báo cáo đồ án được trình bày cụ thể theo 4 chương sau:

- **Chương 1. Tổng quan về mật mã hoá** – Trình bày các kiến thức cơ sở liên quan đến mật mã hoá. Mã hoá đầu cuối và những cách triển khai; quản lý, phân phối khoá và ứng dụng vào mã hoá đầu cuối; những kiến thức cơ sở liên quan đến lý thuyết hỗn loạn, sự phù hợp của lý thuyết hỗn loạn trong mật mã học.
- **Chương 2. Thiết kế hệ thống mật mã hoá** – Trình bày về hệ thống mật mã hoá sử dụng lý thuyết hỗn loạn, giải thích chi tiết các khối trong hệ thống hoạt động,

quá trình mật mã và giải mật, quá trình sinh khoá cho hệ thống từ khoá được chia sẻ trong quá trình trao đổi khoá.

- **Chương 3. Triển khai ứng dụng** – Trình bày về cách triển khai ứng dụng nhắn tin có sử dụng hệ mật hỗn loạn được thiết kế ở Chương 2. Ứng dụng bao gồm máy chủ và ứng dụng android dùng để nhắn tin. Máy chủ dùng để lưu thông tin người dùng, các đoạn hội thoại đã được mã hoá. Còn ứng dụng android dùng để nhắn tin, mã hoá và giải mã sẽ ngay trên ứng dụng, sử dụng khoá trao đổi trong quá trình trao đổi khoá.
- **Chương 4. Kiểm thử và đánh giá** – Kiểm thử các chức năng và đánh giá hệ mật và ứng dụng được thiết kế ở chương 2 và chương 3 có đạt những yêu cầu đã được đề ra trong đề án không.

CHƯƠNG 1. TỔNG QUAN VỀ MẬT MÃ HOÁ

Trong chương này, đề án sẽ giới thiệu tổng quan về mật mã học, lý thuyết hỗn loạn. Trình bày các cơ sở lý thuyết nền tảng như các loại thuật toán mật mã hoá, mã hoá đầu cuối, những phương pháp triển khai mã hoá đầu cuối. Từ đó lựa chọn phương án phù hợp nhất với đề án. Trình bày những khái niệm ban đầu của lý thuyết hỗn loạn, sự phù hợp của hỗn loạn trong mật mã hoá, lựa chọn phương pháp ứng dụng hỗn loạn trong đề án.

1.1 Tổng quan về mật mã hoá

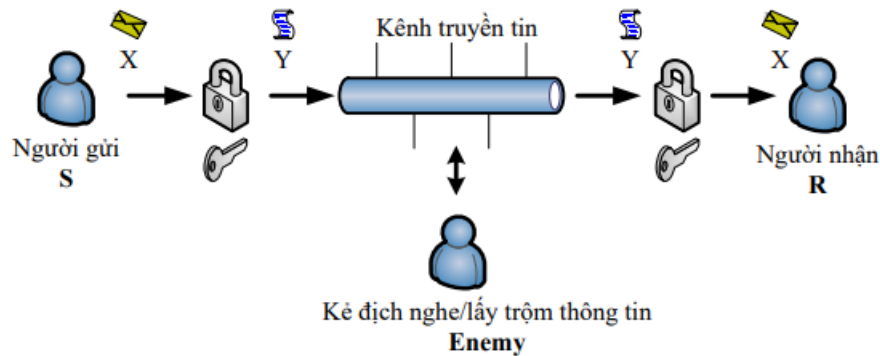
1.1.1 Khái niệm mật mã học

Mật mã học là ngành khoa học nghiên cứu về an toàn thông tin [1]. Mật mã học bao gồm quá trình mật mã, nghĩa là chuyển đổi thông tin từ dạng có thể hiểu được sang dạng không thể hiểu được và ngược lại là quá trình giải mật. Mật mã là công cụ quan trọng để che giấu thông tin trong quá trình trao đổi thông tin, khi mà luôn có sự tồn tại của những đối tượng muốn ăn cắp thông tin nhằm mục đích để lợi dụng hoặc phá hoại.

Mật mã học sử dụng các phép biến đổi theo một quy luật toán học nào đó để biến thông tin có thể hiểu được ban đầu thành thông tin mà kẻ địch không thể hiểu được và đồng thời cũng có khả năng khôi phục lại thành thông tin ban đầu để người trong cuộc có thể đọc được. Trong Lý thuyết truyền thông của các hệ thống bảo mật (Communication theory of secrecy systems), Shannon lần đầu tiên đưa ra khái niệm an toàn thông tin bằng mô hình toán học [2], là nền tảng của lý thuyết mật mã hiện đại.

Mô hình truyền tin bảo mật được mô tả trên Hình 1.1, ở mô hình này xuất hiện kẻ địch ẩn giấu muốn lấy cắp, phá hoại thông tin. Để chống lại điều này, khối xử lý mật mã và giải mật được thêm vào theo nguyên tắc sau: Người gửi S khi muốn gửi một thông điệp X đến người nhận R qua một kênh truyền tin và kẻ địch (Enemy) muốn nghe trộm thông điệp X này. Khi đó, để bảo mật, người gửi S sử dụng những phép biến đổi để biến đổi thông điệp X thành thông điệp Y không thể đọc được, rồi gửi đến cho người nhận R. Tại đây, người nhận R sử dụng những quy tắc của S để biến đổi lại thông điệp Y thành thông điệp X ban đầu. Khi đó, ta gọi X là bản rõ, Y là bản mật và bản mật Y đã che giấu nội dung từ bản rõ X ban đầu. Quá trình biến đổi từ X thành Y được gọi là quá trình mật

mã hoá. Ngược lại, quá trình người nhận thu được bản rõ X từ bản mã Y được gọi là quá trình giải mật mã hoá [3].



Hình 1.1 Mô hình truyền tin mật

Định nghĩa 1.1: Một hệ thống mật mã là một bộ 5 (P, C, K, E, D) thoả mãn các điều kiện sau:

1. P (Plaintext) là một tập hữu hạn các kí tự bản rõ (thông điệp gốc)
2. C (Ciphertext) là một tập hữu hạn các kí tự bản mã (thông điệp được mã hoá)
3. K (Key) là một tập hữu hạn các khóa mã hóa
4. E/D (Encryption/Decryption) là phép lập mã, giải mật, trong đó E là một ánh xạ từ $K \times P$ vào C , D là một ánh xạ từ $K \times C$ vào P . Với mỗi khóa $k \in K$, giả thiết $E_k: P \rightarrow C$ và $D_k: C \rightarrow P$ là hai hàm cho bởi $E_k = E(k, X)$, $D_k = D(k, Y)$ được gọi là hàm mật mã hoá và giải mật mã hoá tương ứng, thoả mãn hệ thức $D_k(E_k(X)) = X$

Trong đó tính chất 4 là tính chất quan trọng của một hệ thống mật mã hoá. Tính chất này đảm bảo một mẫu tin $x \in P$ được mật mã bằng luật mật mã hoá E_k có thể được giải mật chính xác bằng luật giải mật D_k tương ứng.

Ngược lại của quá trình mật mã là quá trình thám mã. Thám mã là quá trình khôi phục lại bản rõ hoặc khoá khi chỉ có bản mật tương ứng cho trước (không biết khoá và quy tắc mã/dịch). Một giải pháp mật mã là đảm bảo bí mật, nếu mọi thuật toán thám mã (nếu có) đều phải được thực hiện với độ phức tạp tính toán và thời gian tính toán cực lớn.

1.1.2 Những nguyên lý của bảo mật thông tin

Một hệ thống thông tin được coi là bảo mật khi hệ thống đó tuân thủ theo 5 nguyên lý sau: Nguyên lý bí mật, riêng tư, Nguyên lý toàn vẹn, Nguyên lý xác thực, Nguyên lý

không chối bỏ và Nguyên lý nhận dạng [4].

1.1.2.1 Nguyên lý bí mật, riêng tư

Giả sử A gửi một “vật mang tin” đến cho B. Nguyên lý đầu tiên của lý thuyết bảo mật là phải đảm bảo tính bí mật và tính riêng tư cho quá trình truyền tin. Điều này có nghĩa là việc truyền tin phải đảm bảo rằng chỉ có hai đối tác A và B khi tiếp cận vật mang tin mới nắm bắt được nội dung thông tin được truyền. Trong quá trình truyền tin, nếu có kẻ thứ ba C (vì một nguyên nhân nào đó) có thể tiếp cận được vật mang tin thì phải đảm bảo rằng kẻ đó vẫn không thể nắm bắt được, không thể hiểu được nội dung “thực sự” của thông tin chứa trong vật mang tin đó.

1.1.2.2 Nguyên lý toàn vẹn

Trong quá trình truyền tin, có thể vì lý do khách quan của môi trường, nhất là do sự xâm nhập phá hoại của kẻ thứ ba, nội dung của thông tin ban đầu chứa trong vật mang tin có thể bị mất mát hay bị thay đổi. Nguyên lý này không yêu cầu đến mức phải đảm bảo rằng thông tin không bị thay đổi trong quá trình truyền tin, nhưng phải đảm bảo được là mỗi khi thông tin bị thay đổi thì người nhận (và tất nhiên là cả người gửi) đều phát hiện được.

1.1.2.3 Nguyên lý xác thực

Trong một quá trình truyền tin, người nhận tin (và có khi cả người gửi tin nữa) có biện pháp để chứng minh với đối tác rằng “họ chính là họ” chứ không phải là một người thứ ba nào khác.

1.1.2.4 Nguyên lý không chối bỏ

Nguyên lý này đòi hỏi rằng khi quá trình truyền tin kết thúc, A đã gửi cho B một thông tin và B đã nhận thông tin thì A không thể chối bỏ rằng thông tin đó không do mình gửi (hoặc mình không gửi tin) mặt khác B cũng không thể chối bỏ rằng mình chưa nhận được.

1.1.2.5 Nguyên lý nhận dạng

Giả sử một hệ thống tài nguyên thông tin chung có nhiều người sử dụng (users) với những mức độ quyền hạn khác nhau. Nguyên lý này yêu cầu phải có biện pháp để hệ thống có thể nhận dạng được các người sử dụng với quyền hạn kèm theo của họ.

Trong vấn đề bảo mật còn có một điều cần lưu ý: đó là “sự tin tưởng”. Khi chia sẻ một bí mật cho một người, bạn phải tin tưởng vào khả năng bảo vệ bí mật của người đó. Nhưng một điều khó khăn ở đây là: “tin tưởng” là một phạm trù có tính tâm lý, xã hội không có các đặc trưng của một loại quan hệ toán học nào. Chính vì vậy, trong các vấn đề bảo mật nhiều khi chúng ta không thể hoàn toàn dùng các phương pháp suy luận logic thông thường mà phải chú ý đến việc tuân thủ các nguyên lý bảo mật thông tin.

1.1.3 Phân loại các thuật toán mật mã hoá

Ta có thể phân chia các thuật toán mã hoá thành 2 loại: mật mã hiện đại và mật mã cổ điển. Mật mã cổ điển là các loại mật mã xuất hiện trước năm 1970 còn mật mã hiện đại là các mật mã xuất hiện trong khoảng thời gian từ 1970 đến nay.

1.1.3.1 Mật mã cổ điển

Mật mã cổ điển có thể được chia thành 2 nhánh: Chuyển vị và Thay thế với các tính chất khác nhau:

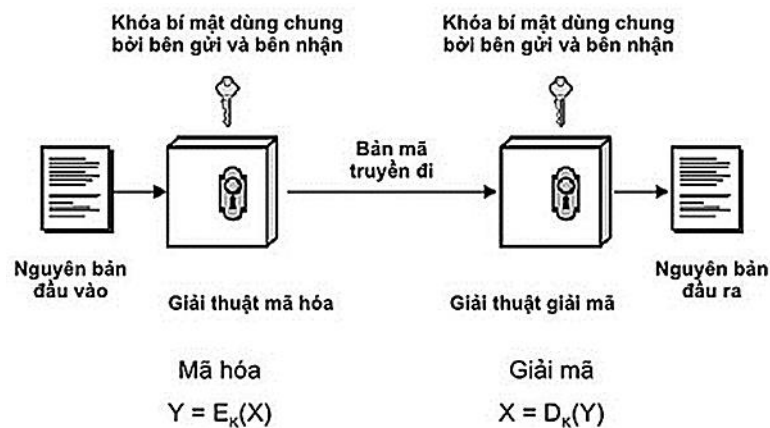
- Chuyển vị: thay đổi vị trí các thành phần (thường là các kí tự) của bản rõ. Bản mật và bản rõ có các kí tự giống nhau nhưng chỉ khác nhau về vị trí của các kí tự
- Thay thế: thay thế một kí tự của bản rõ thành một kí tự khác trong bản mật. Bản mật và bản rõ có các kí tự khác nhau về cả số lượng và vị trí

1.1.3.2 Mật mã hiện đại

Có nhiều cách phân loại các thuật toán mã hóa hiện đại hiện đang sử dụng. Ở đây, đồ án phân loại dựa theo tính chất của khoá [5]:

a) Mật mã hoá đối xứng

Mật mã hoá đối xứng là một thuật toán mà trong đó cả hai quá trình mật mã hóa và giải mật đều dùng một khóa [3]. Mô hình của hệ mật mã hoá đối xứng được thể hiện ở Hình 1.2. Để đảm bảo tính an toàn, khóa này phải được giữ bí mật. Một điều cần lưu ý là khi một người mật mã một bản rõ (plaintext) thành bản mật (ciphertext) bằng một khóa K (thuật toán mật mã) rồi gửi thông điệp này cho đối tác thì đối tác muốn giải mật cũng cần phải có khóa K, nghĩa là trước đó hai đối tác đã phải trao đổi cho nhau chia sẻ để cùng biết được khóa K.



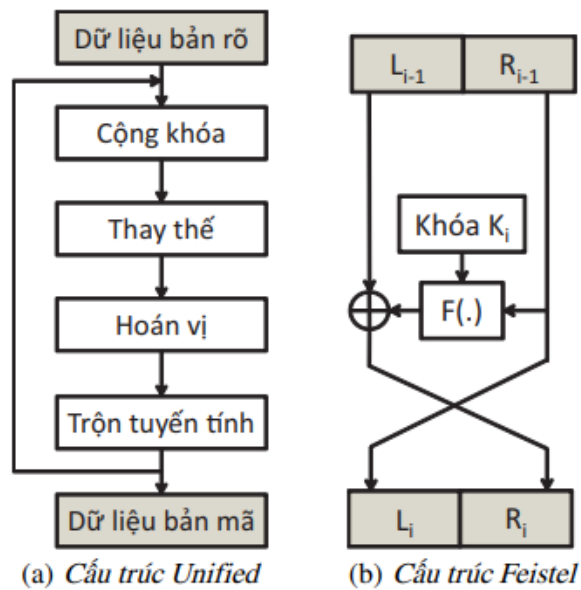
Hình 1.2 Mô hình hệ mật mã hoá đối xứng [4]

Mật mã hóa đối xứng có thể phân thành hai nhóm phụ:

- Thuật toán mã hóa theo khối (Block ciphers): trong đó từng khối dữ liệu trong văn bản gốc ban đầu được thay thế bằng một khối dữ liệu khác có cùng độ dài. Độ dài mỗi khối gọi là kích thước khối (block size), thường được tính bằng đơn vị bit. Ví dụ thuật toán 3-Way có kích thước khối bằng 96 bit. Một số thuật toán khối thông dụng là: DES, 3DES, RC5, RC6, 3-Way, CAST, Camelia, Blowfish, MARS, Serpent, Twofish, GOST...
- Thuật toán mã hóa dòng (Stream ciphers): trong đó dữ liệu đầu vào được mã hóa từng bit một. Các thuật toán dòng có tốc độ nhanh hơn các thuật toán khối, được dùng khi khối lượng dữ liệu cần mã hóa chưa được biết trước, ví dụ trong kết nối không dây. Có thể coi thuật toán dòng là thuật toán khối với kích thước mỗi khối là 1 bit. Một số thuật toán dòng thông dụng: RC4, A5/1, A5/2, Chameleon...

Mức độ bảo mật của các hệ thống mật mã hóa đối xứng sẽ phụ thuộc vào độ khó trong việc suy đoán ngẫu nhiên ra khóa đối xứng theo hình thức tấn công bruteforce. Ví dụ: Dò ra mã hóa của 1 khóa 128 bit – mất vài tỷ năm với máy tính thông thường. Thông thường, khóa độ dài 256 bit được xem là bảo mật tuyệt đối [6].

Trong thuật toán mã hoá theo khối, có thể phân loại theo 2 cấu trúc: Cấu trúc Feistel và cấu trúc Unified (Cấu trúc SPN) như được thấy ở Hình 1.3. Trong đó, cấu trúc Feistel thực hiện chia dữ liệu thành hai nửa rồi mật mã hoá, sau đó đảo các nửa này, còn cấu trúc Unified thực hiện qua nhiều bước mà ở đó điển hình nhất là có bước hoán vị (permutation) và bước thay thế (substitution). Quá trình mật mã có thể được lặp lại nhiều lần để tăng độ bảo mật cho bản mật.



Hình 1.3 Phân loại mã khối theo cấu trúc

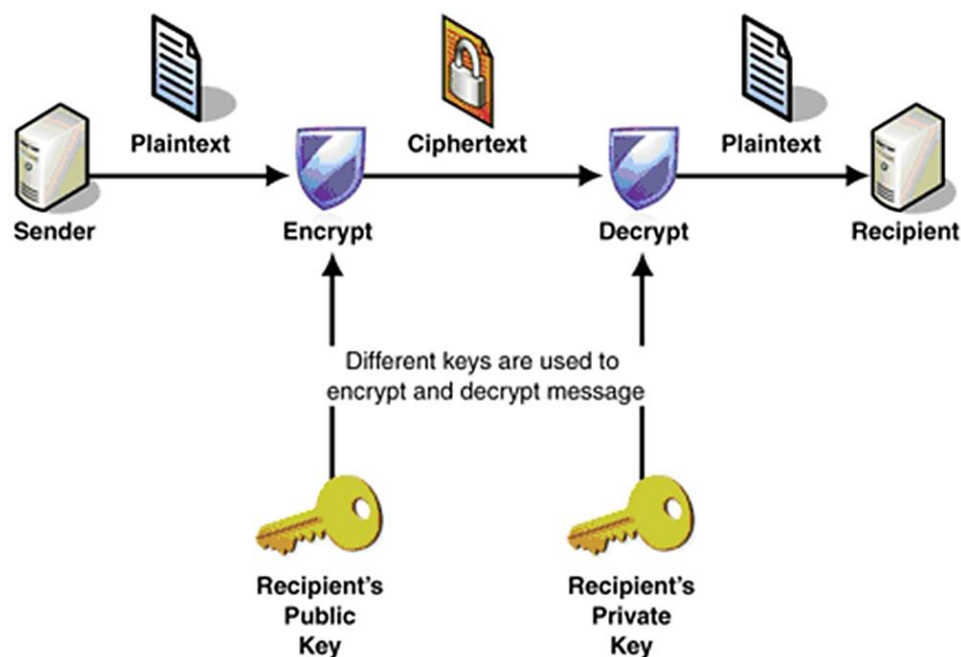
b) Mật mã hoá công khai

Mật mã hoá bất đối xứng hay mật mã hoá công khai là một dạng mật mã hóa cho phép người sử dụng trao đổi các thông tin mật mà không cần phải trao đổi các khóa bí mật trước đó. Điều này được thực hiện bằng cách sử dụng một cặp khóa có quan hệ toán học với nhau là khóa công khai (Public key) và khóa riêng (Private key) hay khóa bí mật (Secret key).

Trong mật mã hóa công khai, khóa riêng cần phải được giữ bí mật trong khi khóa công khai được phổ biến công khai. Trong 2 khóa, một dùng để mã hóa và khóa còn lại dùng để giải mã như Hình 1.4. Điều quan trọng đối với hệ thống là không thể (hoặc rất khó) tìm ra khóa bí mật nếu chỉ biết khóa công khai.

Hệ thống mật mã hóa công khai có thể sử dụng với các mục đích:

- Mật mã ho: giữ bí mật thông tin và chỉ có người có khóa bí mật mới giải mã được.
- Tạo chữ ký số: cho phép kiểm tra một văn bản xem nó có phải đã được tạo với một khóa bí mật nào đó hay không.
- Thỏa thuận khóa: cho phép thiết lập khóa để trao đổi thông tin mật giữa hai bên.



Hình 1.4 Mô hình hệ mật mã hoá công khai

Thông thường, các kỹ thuật mật mã hóa công khai đòi hỏi khối lượng tính toán nhiều hơn các kỹ thuật mã hóa đối xứng nhưng do những ưu điểm nổi bật nên chúng được sử dụng nhiều.

2.1.3.2.2 Mã hoá một chiều

Mã hoá một chiều là mật mã không có khoá nào trong quá trình mật mã. Trong Mã hoá một chiều, điển hình nhất là hàm băm (hash function). Giá trị băm có độ dài cố định được tính theo văn bản thuần túy, khiến cho nội dung của văn bản thuần túy không thể được khôi phục. Hàm băm cũng được nhiều hệ điều hành sử dụng để mã hóa mật khẩu, và được sử dụng phổ biến trong chữ ký điện tử [7].

1.2 Mã hoá đầu cuối

1.2.1 Khái niệm

Mã hóa đầu cuối (E2EE) là một phương pháp giao tiếp an toàn ngăn các bên thứ ba truy cập vào dữ liệu khi dữ liệu được chuyển từ hệ thống hoặc thiết bị đầu cuối này sang thiết bị đầu cuối khác. Nhiều nhà cung cấp dịch vụ nhắn tin phổ biến sử dụng mã hóa đầu cuối, bao gồm Facebook, WhatsApp và Zoom.

Trong E2EE, dữ liệu được mật mã hoá ở thiết bị người gửi và chỉ những người nhận là có thể giải mật được chúng. Khi dữ liệu di chuyển đến đích, nó không thể bị đọc hoặc giả mạo bởi nhà cung cấp dịch vụ internet (ISP), nhà cung cấp dịch vụ ứng dụng, tin tặc

hoặc bất kỳ tổ chức hoặc dịch vụ nào khác [8].

1.2.2 Cách thức hoạt động

Các khóa mật mã được sử dụng để mật mã hóa và giải mật các thông điệp được lưu trữ trên các thiết bị đầu cuối. Cách tiếp cận này sử dụng mật mã hóa công khai. Trong đó khóa công khai được chia sẻ với người khác. Sau khi được chia sẻ, những người khác có thể sử dụng khóa công khai để mã hóa tin nhắn và gửi đến chủ sở hữu của khóa công khai. Tin nhắn chỉ có thể được giải mã bằng khóa riêng tương ứng, còn được gọi là khóa giải mã.

Ngoài ra, ta có thể sử dụng mật mã hoá công khai để trao đổi khoá giữa các thiết bị đầu cuối. Sau đó dùng khoá trao đổi đó là khoá mật cho các hệ mật mã hoá đối xứng. Khi đó, khối lượng tính toán và thời gian mật mã và giải mật sẽ được rút ngắn.

Từ những ưu điểm từ phương pháp trao đổi khoá, sau đó dùng khoá trao đổi để thực hiện mật mã hoá và giải mật mã hoá, đồ án sẽ sử dụng phương pháp này để thực hiện mã hoá đầu cuối.

1.3 Quản lý và phân phối khoá

Một điểm yếu của mật mã hoá đối xứng đó là vấn đề trao đổi khoá mật giữa các đối tác với nhau trong môi trường không tin cậy. Nghĩa là, trong quá trình trao đổi khoá, kẻ địch lấy được khoá bí mật thì họ có thể giải mật được thông điệp truyền đi. Do đó, để đảm bảo độ an toàn của hệ mật, phải xác lập những phương thức chuyển giao khoá mật một cách an toàn.

Hiện nay, có nhiều phương pháp để quản lý và phân phối khoá, trong đó có trao đổi khoá sử dụng thuật toán Elliptic Curve Diffie–Hellman (ECDH)

1.3.1 Mật mã đường cong Elliptic (Elliptic Curve Cryptography – ECC)

ECC là một họ hệ thống mật mã công khai hiện đại ECC dựa trên cấu trúc đại số của các đường cong elliptic trên các trường hữu hạn và dựa trên độ khó của Bài toán Logarit rời rạc Đường cong Elliptic (Elliptic Curve Discrete Logarithm Problem – ECDLP). ECC có tất cả các khả năng chính của hệ thống mật mã hoá công khai: mật mã, giải mật, tạo chữ ký số và trao đổi khoá [9].

ECC là một trong những loại mật mã mạnh nhất hiện nay. Nhiều công ty như

CloudFlare,... đã sử dụng ECC để bảo mật mọi thứ từ kết nối HTTPS của khách hàng đến cách thức chuyển dữ liệu giữa các trung tâm truyền dữ liệu [10].

Mật mã ECC được coi là sự kế thừa tốt của hệ thống mật mã RSA, vì ECC sử dụng các khóa và chữ ký nhỏ hơn RSA ở cùng mức độ bảo mật và cung cấp khả năng tạo khóa, thỏa thuận khóa, tạo chữ ký nhanh [9].

Các thuật toán ECC có thể sử dụng các đường cong elliptic khác nhau. Các đường cong khác nhau cung cấp mức độ bảo mật khác nhau, hiệu suất khác nhau và độ dài khóa khác nhau và cũng có thể liên quan đến các thuật toán khác nhau.

1.3.1.1 Đường cong Elliptic trong trường số thực

a) Định nghĩa

Trong toán học, đường cong elliptic là một đường cong đại số phẳng, bao gồm tất cả các điểm $\{x, y\}$ thỏa mãn Phương trình (1.2) [11]:

$$Ax^3 + Bx^2y + Cxy^2 + Dy^3 + Ex^2 + Fxy + Gy^2 + Hx + Iy + J = 0 \quad (1.1)$$

Trong mật mã học, đường cong elliptic được sử dụng ở dạng đơn giản hơn (Dạng Weierstras) được biểu diễn như Phương trình (1.2):

$$y^2 = x^3 + ax + b \quad (1.2)$$

Với $4a^3 + 27b^2 \neq 0$. Ví dụ, đường cong NIST *secp256k1* dựa trên đường cong elliptic ở dạng $y^2 = x^3 + 7$ (với $a = 0, b = 7$)

Mỗi đường cong elliptic đều có những hình dạng, đặc điểm khác nhau với $b = 1$ và a trong khoảng 2 đến -3 , được biểu diễn ở Hình 1.5. Tùy thuộc vào giá trị của a và b , các đường cong elliptic sẽ có những hình dạng khác nhau, nhưng chúng đều có điểm chung là đối xứng qua trục x .



Hình 1.5 Các hình dạng của các đường cong elliptic khác nhau

Hình 1.6 là các loại điểm kỳ dị. Trong đó, hình bên trái là đường cong với một đỉnh ($y^2 = x^3$), còn hình bên phải là đường cong tự giao ($y^2 = x^3 - 3x + 2$). Tất cả chúng đều không hợp lệ, không thoả mãn điều kiện của đường cong.



Hình 1.6 Các loại điểm kỳ dị

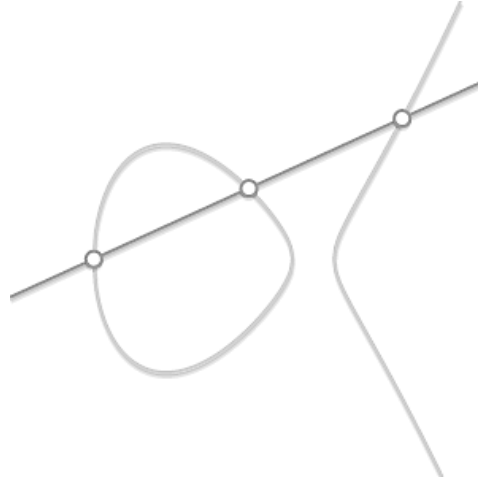
Ta định nghĩa thêm một “điểm vô cực” mà không thể biểu diễn trên đồ thị, được kí hiệu là 0. Khi đó, đường cong được định nghĩa như Phương trình (1.3)

$$\{(x, y) \in \mathbb{R}^2 | y^2 = x^3 + ax + b, 4a^3 + 27b^2 \neq 0\} \cup \{0\} \quad (1.3)$$

Khi đó, đường cong Elliptic có đầy đủ tính chất của một nhóm:

- Các phần tử của nhóm là các điểm của một đường cong elliptic

- Phần tử trung hoà là điểm vô cực 0
- Phần tử đối xứng của điểm P là điểm đối xứng với nó qua trục x trên đồ thị
- Ngoài ra, khi 3 điểm thẳng hàng khác điểm 0 cùng nằm trên một đường thẳng, thì tổng của chúng bằng 0: $P + Q + R = 0$ (Hình 1.7)



Hình 1.7 Tổng của 3 điểm thẳng hàng bằng 0

Do P, Q, R không cần tuân theo thứ tự nên $P + (Q + R) = (P + Q) + R = \dots = 0$, có thêm tính chất kết hợp, nên đường cong này là một nhóm abel. Khi đó, các phép toán của đường cong elliptic được biểu diễn như sau:

b) Các phép toán trong trường số thực

- Phép cộng:

- $P + (-P) = 0$ và $P + 0 = 0 + P = P$
- $P \neq Q$: Đường thẳng nối P và Q có độ dốc

$$m = \frac{y_P - y_Q}{x_P - x_Q} \quad (1.4)$$

Khi đó $R = (x_R, y_R)$:

$$\begin{aligned} x_R &= m^2 - x_P - x_Q \\ y_R &= y_P + m(x_R - x_P) = y_Q + m(x_R - x_Q) \end{aligned} \quad (1.5)$$

- $P = Q$: Đường thẳng có độ dốc

$$m = \frac{3x_P^2 + a}{2y_P} \quad (1.6)$$

Thay m vào Phương trình (1.5) để tính điểm R

- Phép nhân với số nguyên: $nP = P + P + \dots + P$. Có nhiều thuật toán để thực hiện phép nhân, một trong số đó là nhân đôi và cộng (“double and add”). Ví dụ với $n = 151$ được biểu diễn dưới dạng nhị phân là 10010111_2 . Dạng nhị phân này được biểu diễn dưới dạng tổng của 2 là $151 = 2^7 + 2^4 + 2^2 + 2^1 + 2^0$. Khi đó phép nhân $151P = 2^7P + 2^4P + 2^2P + 2P + P$. Từ P ta sẽ tính được $2P, 2^2P, 2^4P$, sau đó cộng tổng lại với nhau. Thuật toán trên được biểu diễn dưới dạng mã Python như sau:

```
def bits(n):
    """
    Generates the binary digits of n, starting
    from the least significant bit.

    bits(151) -> 1, 1, 1, 0, 1, 0, 0, 1
    """
    while n:
        yield n & 1
        n >>= 1

def double_and_add(n, x):
    """
    Returns the result of n * x, computed using
    the double and add algorithm.
    """
    result = 0
    addend = x

    for bit in bits(n):
        if bit == 1:
            result += addend
            addend *= 2

    return result
```

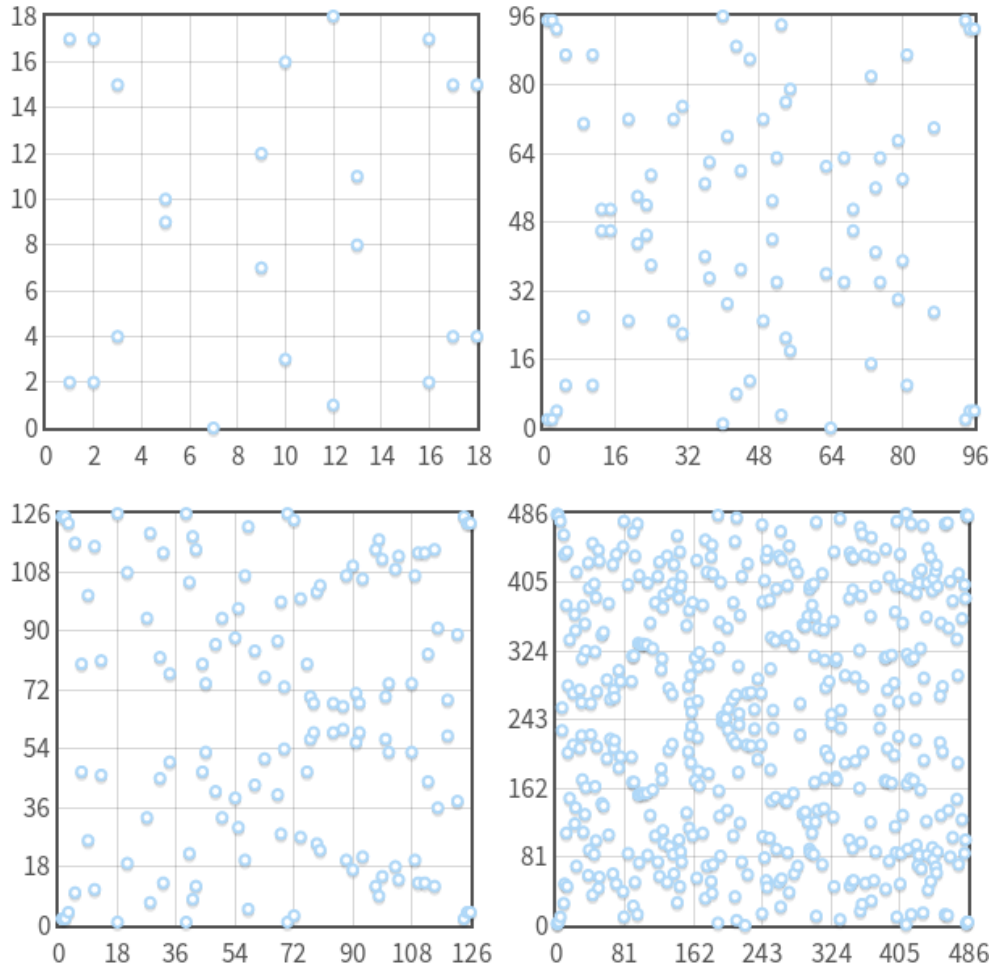
1.3.1.2 Đường cong Elliptic trong trường hữu hạn

a) Định nghĩa

Đường cong Elliptic trong trường hữu hạn GF_p với p là số nguyên tố được biểu diễn như

$$\left\{ (x, y) \in (GF_p)^2 \mid y^2 \equiv x^3 + ax + b \pmod{p}, 4a^3 + 27b^2 \not\equiv 0 \pmod{p} \right\} \cup \{0\} \quad (1.7)$$

với 0 là điểm vô cực, a và b đều là các số nguyên trong trường GF_p



Hình 1.8 Một số dạng đồ thị của đường cong trong trường hữu hạn

Hình 1.8 là đường cong $y^2 \equiv x^3 - 7x + 10 \pmod{p}$ với $p = 29,97,237,487$. Với mỗi giá trị của x sẽ có 2 điểm và đối xứng qua $y = p/2$. Khi đó, 3 điểm nằm trên một đường thẳng khi và chỉ khi $ax + by + c \equiv 0 \pmod{p}$

b) Các phép toán trong trường hữu hạn

- Phép cộng

Các phép cộng trong trường hữu hạn được biểu diễn như sau:

- $P + 0 = 0 + P = P$
- $-P = (x_P, -y_P \pmod{p})$
- $P + (-P) = 0$
- $P + Q = R$ với

$$\begin{aligned} x_R &= (m^2 - x_P - x_Q) \pmod{p} \\ y_R &= [y_P + m(x_R - x_P)] \pmod{p} = [y_Q + m(x_R - x_Q)] \pmod{p} \end{aligned} \quad (1.8)$$

Với $P \neq Q$:

$$m = (y_P - y_Q)(x_P - x_Q)^{-1} \quad (1.9)$$

Với $P = Q$:

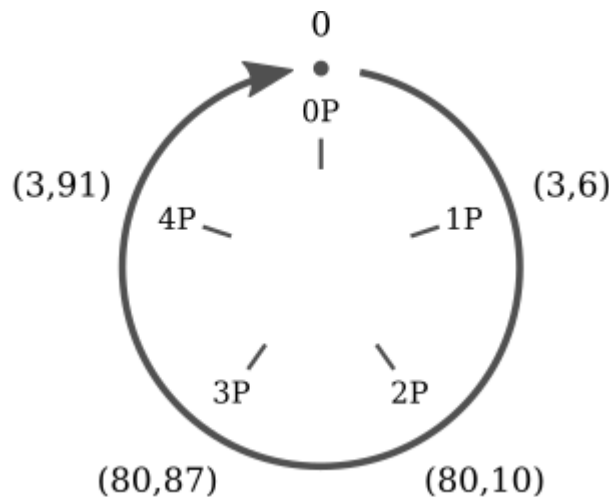
$$m = (3x_P^2 + a)(2y_P)^{-1} \quad (1.10)$$

- Bậc của một nhóm đường cong elliptic là số điểm của đường cong sau khi rời rạc hoá. Để tìm bậc của một nhóm, ta sẽ dùng thuật toán Schoof.
- Phép nhân vô hướng

Khi thực hiện phép nhân với số nguyên, ta vẫn sử dụng thuật toán nhân đôi và cộng. Trong trường hữu hạn, đường cong Elliptic có tính chất đặc biệt, đó là các kết quả sẽ được lặp lại sau một số lần nhất định, có tính chất chu kì.

Ví dụ: Khi thực hiện phép nhân của đường $y^2 \equiv x^3 + 2x + 3 \pmod{97}$ với điểm $P = (3,6)$:

- | | |
|------------------|------------------|
| ○ $0P = 0$ | ○ $5P = 0$ |
| ○ $1P = (3,6)$ | ○ $6P = (3,6)$ |
| ○ $2P = (80,10)$ | ○ $7P = (80,10)$ |
| ○ $3P = (80,87)$ | ○ $8P = (80,87)$ |
| ○ $4P = (3,91)$ | ○ $9P = (3,91)$ |



Hình 1.9 Phép nhân của P với 5 điểm khác biệt

Hình 1.9 là phép nhân của $P = (3,6)$ với 5 điểm khác biệt ($P, 2P, 3P, 4P$) và chúng lặp lại theo vòng tròn. Nhìn vào hình, dễ dàng nhận ra sự giống nhau giữa phép nhân vô hướng trên đường cong elliptic và phép cộng trong số học modulo.

Các điểm sinh ra của phép nhân vô hướng được gọi là nhóm con tuần hoàn của nhóm được tạo thành bởi đường cong elliptic với điểm sinh là điểm P . Để tìm bậc của nhóm con này, ta sử dụng mối liên hệ với bậc của đường cong bởi định lý Lagrange, trong đó nói rằng bậc của một nhóm con là một ước số của bậc của nhóm mẹ. Nói cách khác, nếu một đường cong elliptic chứa n điểm và một trong các nhóm con của nó chứa r điểm, thì r là ước của n .

Một số đường cong tạo thành một nhóm tuần hoàn duy nhất (bao gồm tất cả các điểm EC của chúng), trong khi những đường khác tạo thành một số nhóm con tuần hoàn không chồng lên nhau (mỗi đường chứa một tập con các điểm EC của đường cong). Trong trường hợp thứ hai, các điểm trên đường cong được chia thành h nhóm con tuần hoàn, mỗi nhóm có bậc r (mỗi nhóm con chứa số điểm bằng nhau). Bậc của toàn bộ nhóm là $n = hr$ số nhóm con, nhân với số điểm trong mỗi nhóm con). Số lượng nhóm con h giữ các điểm EC được gọi là đồng yếu tố (cofactor). Nếu đường cong chỉ bao gồm một nhóm con tuần hoàn, hệ số của nó $h = 1$. Nếu đường cong bao gồm một số nhóm con, hệ số của nó lớn hơn 1. Ví dụ: Đường cong *secp256k1* $h = 1$, *Curve25519* có $h = 8$, *Curve448* có $h = 4$

1.3.1.3 Các tham số của đường cong Elliptic

Thuật toán ECC cần những thông số (p, a, b, G, n, h) . Trong đó

- Số nguyên tố p (prime) xác định kích thước của trường hữu hạn.
- Các hệ số a và b (coefficients) của phương trình đường cong elip.
- Điểm sinh G (base point) tạo ra nhóm con
- Bậc n (order) của nhóm
- Đồng hệ số h (cofactor) của nhóm con

1.3.1.4 Khoá của ECC

Khoá bí mật trong ECC là những số nguyên (trong phạm vi kích thước trường của đường cong, thường là số nguyên 256 bit). Ví dụ một khoá bí mật của ECC có độ dài 256 bit được biểu diễn dưới dạng hex [9]:

0x51897b64e85c3f714bba707e867914295a1377a7463a9dae8ea6a8b914246319

Việc tạo khoá trong ECC đơn giản như tạo một cách an toàn một số nguyên ngẫu nhiên trong một phạm vi nhất định, do đó việc này cực kỳ nhanh chóng. Bất kỳ số nào

trong phạm vi đều là khóa bí mật hợp lệ.

Các khóa công khai trong ECC là các điểm EC - các cặp tọa độ số nguyên $\{x, y\}$, nằm trên đường cong. Do các tính chất đặc biệt của chúng, các điểm EC có thể được nén thành chỉ một tọa độ +1 *bit* (lẻ hoặc chẵn). Do đó, khóa công khai được nén, tương ứng với khóa bí mật 256-bit, là một số nguyên 257-bit. Ví dụ về khóa công khai ECC (tương ứng với khóa riêng ở trên, được mã hóa ở định dạng Ethereum, dưới dạng hex với tiền tố 02 hoặc 03) là:

```
0x02f54ba86dc1ccb5bed0224d23f01ed87e4a443c47fc690d7797a13d41d2340e1a
```

Trong định dạng này, khóa công khai thực sự có 33 byte (66 chữ số hex), có thể được tối ưu hóa thành chính xác 257 bit.

Các đường cong ECC, được chấp nhận trong các thư viện mật mã phổ biến và các tiêu chuẩn bảo mật, có tên (đường cong được đặt tên, ví dụ: *secp256k1* hoặc *Curve25519*), kích thước trường (xác định độ dài khóa, ví dụ: 256-bit), cường độ bảo mật (thường là một nửa kích thước trường hoặc ít hơn), hiệu suất (phép toán / giây) và nhiều thông số khác.

Các khóa ECC có độ dài, phụ thuộc trực tiếp vào đường cong bên dưới. Trong hầu hết các ứng dụng (như OpenSSL, OpenSSH và Bitcoin), độ dài khóa mặc định cho khóa bí mật ECC là 256 bit, nhưng tùy thuộc vào đường cong, có thể có nhiều kích thước khóa ECC khác nhau: 192 bit (đường cong *secp192r1*), 233 bit (đường cong *sect233k1*), 224 bit (đường cong *secp224k1*), 256 bit (đường cong *secp256k1* và *Curve25519*), 283 bit (đường cong *sect283k1*), 384 bit (đường cong *p384* và *secp384r1*), 409 bit (đường cong *sect409r1*), 414 bit (đường cong *Curve41417*), 448 bit (đường cong *Curve448-Goldilocks*), 511-bit (đường cong *M-511*), 521-bit (đường cong *P-521*), 571-bit (đường cong *sect571k1*) và nhiều loại khác.

1.3.2 Tính bảo mật

Khoá công khai được tạo từ phép nhân vô hướng với điểm sinh của đường cong. Nên nếu tìm được hệ số nhân là có thể phá được hệ mật. Với ECC, bài toán đó tên là Bài toán logarit rời rạc của đường cong Elliptic (ECDLP) trong khoa học máy tính được định nghĩa như sau: “Bằng đường cong elliptic đã cho trên trường hữu hạn GF_p và điểm sinh G trên đường cong và điểm P trên đường cong, hãy tìm số nguyên k (nếu nó tồn tại), sao cho $P = kG$ ”

Có hai thuật toán hiệu quả nhất để tính toán logarit rời rạc trên đường cong elliptic: thuật toán bước nhỏ, bước khổng lồ (baby-step, giant-step algorithm) và phương pháp rho của Pollard (Pollard's rho method)

1.3.2.1 Thuật toán bước nhỏ, bước khổng lồ [12]

Một số nguyên x luôn có thể viết được dưới dạng $x = am + b$. Áp dụng vào công thức tính khoá công khai:

$$Q = xP$$

$$\Leftrightarrow Q = (am + b)P$$

$$\Leftrightarrow Q = amP + bP$$

$$\Leftrightarrow Q - amP = bP$$

Bước nhỏ, bước khổng lồ là một thuật toán "gặp nhau ở giữa". Trái ngược với cuộc tấn công brute-force (buộc chúng ta phải tính tất cả các điểm xP cho mọi x cho đến khi chúng ta tìm thấy Q), chúng ta sẽ tính vài giá trị cho bP và giá trị cho $Q - amP$ cho đến khi chúng ta tìm thấy sự tương ứng. Thuật toán hoạt động như sau:

- **Bước 1.** Tính $m = \sqrt{n}$
- **Bước 2.** Với mỗi giá trị của b với $b \in [0, m]$ tính bP
- **Bước 3.** Với mỗi giá trị của a với $a \in [0, m]$ tính $Q - amP$
- **Bước 4.** Kiểm tra $Q - amP$ với bP bằng bảng băm
- **Bước 5.** Lặp lại cho đến khi $Q - amP = bP$, ta tìm được giá trị $x = am + b$

Thuật toán có độ phức tạp $O(\sqrt{n})$. Ví dụ một đường cong chuẩn hóa: *prime192v1* (hay còn gọi là *secp192r1*, *ansiX9p192r1*). Đường cong này có bậc $n = 0xffffffff ffffffff ffffffff 99def836 146bc9b1 b4d22831$. Căn bậc hai của n là khoảng $7.922816251426434 \cdot 10^{28}$.

Để lưu trữ \sqrt{n} điểm của bảng băm trong bộ nhớ, giả sử mỗi điểm yêu cầu 32 byte để lưu trữ thì ta cần 2.5×10^{30} byte bộ nhớ, lớn hơn khoảng 10 lần so với tổng bộ nhớ hiện được lưu trữ trên thế giới 79ZB (79×10^{21} byte). Mà đây là đường cong có thể coi là có độ bảo mật thấp nhất.

1.3.2.2 Phương pháp ρ của Pollard [13]

Với rho của Pollard, chúng ta sẽ giải quyết một vấn đề rất khác: P và Q cho trước,

tìm các số nguyên a, b, A và B sao cho $aP + bQ = AP + BQ$.

Ta có:

$$\begin{aligned} aP + bQ &= AP + BQ \\ \Leftrightarrow aP + bxP &= AP + BxP \\ \Leftrightarrow (a + bx)P &= (A + Bx)P \\ \Leftrightarrow (a - A)P &= (B - b)xP \end{aligned}$$

Bây giờ chúng ta có thể loại bỏ P . Do nhóm con của đường cong là tuần hoàn với bậc n , nên các hệ số được sử dụng trong phép nhân điểm là modulo n :

$$\begin{aligned} a - A &\equiv (B - b)x \pmod{n} \\ \Leftrightarrow x &\equiv (a - A)(B - b)^{-1} \pmod{n} \end{aligned}$$

Nguyên tắc hoạt động của Pollard's rho rất đơn giản: tạo ra một chuỗi giả ngẫu nhiên của các điểm X_1, X_2, \dots trong đó mỗi $X = a_iP + b_iQ$. Chuỗi có thể được tạo bằng cách sử dụng một hàm giả ngẫu nhiên f như sau:

$$(a_{i+1}, b_{i+1}) = f(X_i)$$

Nghĩa là: hàm giả ngẫu nhiên f lấy điểm gần nhất X_i trong dãy làm đầu vào và cho các hệ số a_{i+1} và b_{i+1} làm đầu ra. Từ đó tính được $X_{i+1} = a_{i+1}P + b_{i+1}Q$; sau đó nhập X_{i+1} vào f một lần nữa và lặp lại cho đến khi có điểm X được lặp lại.

Certicom đưa ra một thử thách vào năm 1998 để tính toán logarit rời rạc trên các đường cong elliptic với độ dài bit từ 109 đến 359. Cho đến ngày nay, chỉ có đường cong dài 256 bit được phá vỡ thành công với thời gian 13 ngày với sức mạnh tính toán song song của 256 card đồ họa Tesla V100. Do vậy, nếu sử dụng các máy tính cá nhân, thì thời gian để phá mã thành công sẽ rất lớn, và lúc phá được rồi thì thông tin được mật mã có thể đã không còn giá trị.

1.3.3 So sánh ECC và RSA

NIST cung cấp một bảng so sánh kích thước khóa RSA và ECC cần thiết để đạt được cùng một mức độ bảo mật, được biểu diễn ở Bảng 1.1:

Bảng 1.1 So sánh kích thước khoá của RSA và ECC cùng mức độ bảo mật (bit)

RSA	ECC
1024	160

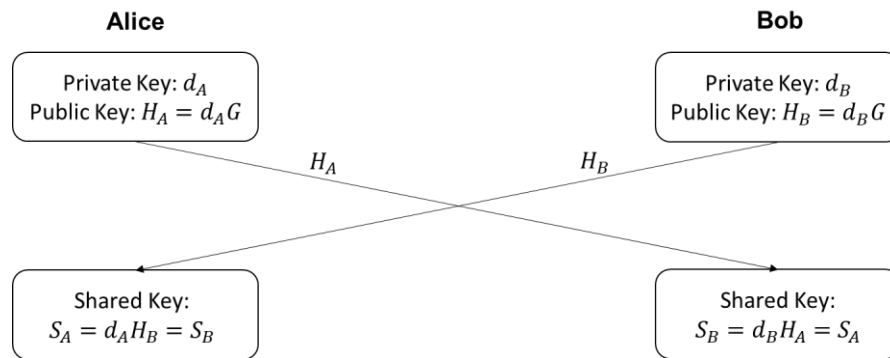
2048	224
3072	256
7680	384
15360	521

Từ Bảng 1.1, để đạt cùng mức độ bảo mật thì ECC yêu cầu độ dài bit thấp hơn so với RSA, do đó thời gian và khối lượng tính toán ít hơn so với RSA.

1.3.4 Trao đổi khoá bằng thuật toán *Elliptic Curve Diffie-Hellman*

ECDH là lược đồ thỏa thuận khóa ẩn danh, cho phép hai bên, mỗi bên có cặp khóa công khai – bí mật theo đường cong elliptic, thiết lập bí mật được chia sẻ trên một kênh không an toàn. ECDH rất giống với thuật toán DHKE (Diffie – Hellman Key Exchange) cổ điển, nhưng nó sử dụng phép nhân điểm ECC thay vì lũy thừa mô-đun [14].

Các bước để thực hiện trao đổi khoá bằng ECDH như Hình 1.10:



Hình 1.10 Quá trình trao đổi khoá bằng ECDH

- **Bước 1.** Alice và Bob sinh cho mình một khoá bí mật và một khoá công khai: Khoá bí mật d_A , khoá công khai $H_A = d_A G$ cho Alice, khoá bí mật d_B và khoá công khai $H_B = d_B G$ cho Bob, với G là điểm sinh của đường cong.
- **Bước 2.** Alice và Bob trao đổi khoá công khai của mình thông qua kênh không an toàn. Người có ý định phá hoại, đánh cắp thông tin, có thể lấy được H_A và H_B nhưng không dễ dàng tìm lại được d_A và d_B .
- **Bước 3.** Alice tính $S = d_A H_B$ (sử dụng khoá bí mật của mình và khoá công khai của Bob) và Bob tính $S = d_B H_A$ (sử dụng khoá bí mật của mình và khoá công khai của Alice). Giá trị S tính được của 2 người là như nhau và có thể được sử dụng để thực hiện mật mã hoá đối xứng như AES.

Ví dụ: Alice chọn đường cong *secp256k1* để trao đổi khoá với các tham số:

- $p=0xffffffff ffffffff ffffffff ffffffff ffffffff ffffffff fffffffe fffffc2f$

- $a=0$
- $b=7$
- $x_G=0x79be667e\ f9dcbbac\ 55a06295\ ce870b07\ 029bfcdb\ 2dce28d9\ 59f2815b\ 16f81798$
- $y_G=0x483ada77\ 26a3c465\ 5da4fbfc\ 0e1108a8\ fd17b448\ a6855419\ 9c47d08f\ fb10d4b8$
- $n=0xffffffff\ ffffffff\ ffffffff\ ffffffff\ baaedce6\ af48a03b\ bfd25e8c\ d0364141$
- $h=1$

Ta thu được kết quả sau:

```
Curve: secp256k1
Alice's private key:
0xe32868331fa8ef0138de0de85478346aec5e3912b6029ae71691c384237a3eeb
Alice's public key:
(0x86b1aa5120f079594348c67647679e7ac4c365b2c01330db782b0ba611c1d677,
0x5f4376a23eed633657a90f385ba21068ed7e29859a7fab09e953cc5b3e89beba)
Bob's private key:
0xcef147652aa90162e1fff9cf07f2605ea05529ca215a04350a98ecc24aa34342
Bob's public key:
(0x4034127647bb7fdab7f1526c7d10be8b28174e2bba35b06ffd8a26fc2c20134a,
0x9e773199edc1ea792b150270ea3317689286c9fe239dd5b9c5cfd9e81b4b632)
Shared secret:
(0x3e2ffbc3aa8a2836c1689e55cd169ba638b58a3a18803fcf7de153525b28c3cd,
0x43ca148c92af58ebdb525542488a4fe6397809200fe8c61b41a105449507083)
```

1.4 Hàm hỗn loạn và mật mã hoá hỗn loạn

1.4.1 Khái niệm lý thuyết hỗn loạn

Hỗn loạn thể hiện trạng thái hỗn độn, thiếu trật tự của các hiện tượng tự nhiên như sự biến đổi của thời tiết khí hậu, trạng thái động học của các hành tinh trong hệ mặt trời,... Về mặt khoa học, lý thuyết hỗn loạn dùng để mô hình hệ thống vận động có vẻ như không có trật tự nhưng lại tuân theo một quy luật hoặc nguyên tắc nào đó [15].

1.4.2 Một số tính chất quan trọng

Theo tác giả L. Kocarev trong Chaos-Based Cryptography Theory, Algorithms and Applications [16], lý thuyết hỗn loạn có ba tính chất quan trọng: Nhạy cảm với điều kiện đầu, Vận động bất quy tắc trong mặt phẳng pha và Hệ thống xác định.

1.4.2.1 Nhạy cảm với điều kiện đầu

Quỹ đạo xuất phát từ các điều kiện khởi tạo sai khác nhau rất nhỏ sẽ phân tách rất nhanh theo luật số mũ tạo ra các quỹ đạo di chuyển hoàn toàn khác nhau. Sự nhạy cảm với điều kiện đầu liên quan đến hàm mũ Lyapunov [17]. Một hệ thống là hệ thống hỗn loạn nếu hàm số mũ này dương.

1.4.2.2 Vận động bất quy tắc trong mặt phẳng pha

Đường di chuyển của hệ thống trong mặt phẳng pha không đi vào bất kỳ điểm cố định hay quỹ đạo có chu kỳ nào khi thời gian vận động tiến tới vô cùng. Vận động bất thường trong hệ thống hỗn loạn được tạo ra do tính phi tuyến bên trong nó chứ không phải do nhiễu. Sự vận động này không thể dự báo dài hạn.

1.4.2.3 Hệ thống xác định

Là hệ thống không có các thông số thống kê xác suất. Đây là điểm khác nhau quan trọng giữa hệ thống hỗn loạn và hệ thống nhiễu với quá trình ngẫu nhiên.

1.4.3 Phân loại hệ thống hỗn loạn

Trong thực tế, hệ thống hỗn loạn được chia thành hai loại gồm liên tục theo thời gian và rời rạc theo thời gian. Hệ hỗn loạn liên tục theo thời gian có biến trạng thái hỗn loạn là một hàm theo thời gian $X(t)$. Ngược lại, các hàm hỗn loạn rời rạc theo thời gian là các hàm được lặp để sinh ra các giá trị X_n . Các hàm hỗn loạn này được ứng dụng vào nhiều lĩnh vực khác nhau.

1.4.3.1 Hệ hỗn loạn liên tục theo thời gian

Các hệ liên tục theo thời gian được biểu diễn bởi hệ phương trình vi phân (1.11):

$$\frac{d\mathbf{X}}{dt} = F(\mathbf{X}) \quad (1.11)$$

Trong đó, $\mathbf{X} = \{x_i, x_i \in \mathbb{R}, i = 1..n\}$ là vector biểu diễn n trạng thái của hệ, t là thời gian. Các hàm liên tục theo thời gian nổi tiếng như Chua's, Lorenz, Rossler,...

Các hệ hỗn loạn liên tục theo thời gian ít được dùng trực tiếp cho mật mã bởi vì việc giải các phương trình vi phân cần nhiều tài nguyên tính toán. Mặt khác, hệ hỗn loạn theo thời gian có thể được mô tả hoặc quan sát thấy theo một số cách khác nhau, như trên mạch điện tử tương tự, trên laser bán dẫn, hoặc trên các hệ cơ khí.

Để sử dụng hệ liên tục theo thời gian vào mục đích mật mã, phương pháp phổ biến là các hàm liên tục được chuyển sang hệ hỗn loạn rời rạc theo một số cách khác nhau. Một trong số cách đó là thực hiện so sánh giá trị của biến trạng thái với các ngưỡng giá trị được đặt ra để chuyển về rời rạc, như trong [18].

1.4.3.2 Hệ hỗn loạn rời rạc theo thời gian

Hàm hỗn loạn rời rạc theo thời gian là các hàm lặp trong quá trình nó hoạt động để tạo ra các giá trị hỗn loạn về biên độ. Một hàm rời rạc theo thời gian được mô tả bởi Phương trình (1.12)

$$\mathbf{X}_{n+1} = \mathbf{F}(\mathbf{X}_n) \quad (1.12)$$

Trong đó, \mathbf{X}_n là véc-tơ biến trạng thái có giá trị tại lần lặp n , hay $\mathbf{X}_n = \{x_n^{(i)}, x_n^{(i)} \in \mathbb{R}, i = 1..m\}$ với m biểu diễn cho số chiều (số phương trình trong hệ). Giá trị \mathbf{X}_0 thường được gọi là vector giá trị điều kiện đầu. Trong trường hợp $m = 1$, biến trạng thái $x_n^{(i)}$ được ký hiệu là x_n .

Một số hàm hỗn loạn được sử dụng trong mật mã hỗn loạn như hàm Logistic, hàm Cat, hàm Chebyshev,...

1.4.4 Sự phù hợp của hỗn loạn trong mật mã hoá

Việc ứng dụng hàm hỗn loạn được dựa trên các yếu tố có lợi để hình thành mật mã, cụ thể là khai thác tính chất phân bố đều của giá trị của biến trạng thái được tạo ra từ hàm hỗn loạn (biến trạng thái) và các tham số của hàm hỗn loạn. Các thuộc tính trên của hàm hỗn loạn đã đưa lại một đặc điểm vô cùng quan trọng đối với hàm hỗn loạn, đó là khả năng không thể dự đoán dài hạn trạng thái của hàm hỗn loạn. Đặc điểm này làm cho hàm hỗn loạn phù hợp với ứng dụng vào mật mã.

Với một hàm hỗn loạn rời rạc, hàm hỗn loạn tạo ra giá trị của biến trạng thái (X_n). Các tham số có thể tác động vào hàm hỗn loạn chính là các điều kiện đầu (X_0), tham số hệ thống, số lần lặp n . Các giá trị của biến đầu ra của hàm hỗn loạn được dùng để tạo ra các chuỗi số và các chuỗi giả ngẫu nhiên cho quá trình mật mã. Việc áp đặt giá trị vào các tham số của hàm hỗn loạn được hiểu như tác động vào đặc tính động của hàm hỗn loạn. Các tham số được tác động sẽ tạo ra giá trị của biến trạng thái và giá trị ấy được dùng cho quá trình mật mã.

Một trong các tính chất quan trọng của chuỗi giá trị được tạo ra từ hàm hỗn loạn đó là tính phân bố đều của chuỗi giả ngẫu nhiên. Tuy nhiên không phải lúc nào hàm hỗn loạn cũng tạo ra các chuỗi có phân bố đều.

1.4.5 So sánh mật mã thông thường và mật mã hỗn loạn

Mối quan hệ chi tiết hơn giữa hỗn loạn và mật mã, được đưa ra trong [19]. Tác giả L. Kocarev trong Chaos-Based Cryptography Theory, Algorithms and Applications [15] cũng đã so sánh tính chất giữa tính chất hỗn loạn của một hệ động học và tính chất mật mã như Bảng 1.2.

Bảng 1.2 So sánh tính chất mật mã và tính chất hỗn loạn

Tính chất hỗn loạn	Tính chất mật mã	Mô tả
Dao động theo quỹ đạo (Ergodicity)	Lộn xộn, hỗn độn (Confusion)	Đầu ra có cùng phân bố với bất kì đầu vào
Nhạy cảm với điều kiện đầu, với các tham số	Tính khuếch tán (Diffusion) khi có một thay đổi rất nhỏ của bản rõ, bản khoá	Sai số nhỏ ở đầu vào có thể là sự thay đổi rất lớn ở đầu ra
Tính trộn lẫn (Mixing)	Tính khuếch tán đối với sự thay đổi nhỏ trong một khối của bản rõ lên toàn bộ bản rõ	Thay đổi trong một khu vực nội bộ là nguyên nhân thay đổi toàn bộ không gian
Đặc tính động tắt định	Tính giả ngẫu nhiên tắt định	Một chu trình tắt định là nguyên nhân thay đổi toàn bộ không gian
Cấu trúc phức tạp	Thuật toán phức tạp	Một chu trình đơn giản có độ phức tạp tính toán lớn

Bảng 1.3 so sánh các đặc trưng cơ bản của hệ mật mã chuẩn và hệ mật mã hỗn loạn. Từ bảng này, có thể thấy mật mã hỗn loạn có không gian biểu diễn giá trị và tham số khoá trong miền số thực, ưu điểm là sẽ dẫn tới không gian khoá và giá trị các tham số là vô hạn, nhưng khi đó xảy ra hạn chế về mặt thực thi và độ phức tạp khi tính toán trên miền số thực. Do đó, việc lựa chọn hệ hỗn loạn thoả mãn bài toán lập mã cũng là một vấn đề cần chú ý để vừa kế thừa được những đặc tính động học hỗn loạn phù hợp với hệ thống lập mã, vừa phải dễ dàng biểu diễn giá trị và thực thi.

Bảng 1.3 So sánh thuật toán lập mật mã chuẩn và mật mã hỗn loạn

Thuật toán lập mật mã chuẩn	Thuật toán lập mật mã hỗn loạn
Không gian pha: là tập hữu hạn các phần tử số nguyên	Không gian pha: Tập (tập con) các phần tử số thực
Phương pháp số học	Phương pháp phân tích
Sử dụng các vòng lặp (rounds)	Sử dụng các bước lặp (iterations)
Khoá (Key) – Hàm Boolean – Không gian khoá rời rạc	Tham số (Parameters) – Số thực – Không gian khoá liên tục

Tính khuếch tán	Nhạy cảm với sự thay đổi của điều kiện đầu, của tham số
Thực hiện số hoá trong miền số nguyên	Thực hiện số hoá bằng hệ thống giá trị xấp xỉ liên tục
Đánh giá bảo mật và sự thực thi theo tiêu chuẩn	Là vấn đề mở cần được nghiên cứu sâu hơn

1.4.6 Ứng dụng của hỗn loạn trong mật mã hoá

Trong thực tế, việc ứng dụng hỗn loạn vào quá trình mật mã được chia ra theo các hướng như sau:

- Tạo chuỗi ngẫu nhiên dùng hỗn loạn: Một trong các đặc trưng của các hàm hỗn loạn là nhạy với điều kiện đầu và nhạy với giá trị của tham số hệ thống. Điều này có nghĩa là với sự thay đổi nhỏ ở giá trị khởi đầu sẽ làm tạo ra các giá trị khác nhau rất lớn ở chuỗi đầu ra. Điều này cũng xảy ra tương tự với giá trị tham số hệ thống của hàm hỗn loạn. Đặc trưng này được dùng để tạo ra chuỗi ngẫu nhiên. Các chuỗi ngẫu nhiên này được dùng trong các phần khác nhau của quá trình mật mã như (i) tạo ra khóa mật cho các bước mật mã, (ii) tạo ra luật hoán vị cho khối P-box (iii) tạo ra giá trị mới dựa trên giá trị của biến trạng thái hỗn loạn, nó tương tự như qua phép thay thế.
- Tạo qui luật hoán vị hoặc thay thế: Các hàm hỗn loạn tham gia trực tiếp vào các khối P-box và S-box theo một số cách khác nhau. Trong mật mã, hàm hỗn loạn có thể nhận điều kiện đầu là vị trí của các giá trị trong chuỗi, sau một loạt các phép lặp, đầu ra của hàm hỗn loạn dùng để xác định vị trí mới của giá trị đó. Tương tự như vậy, giá trị của dữ liệu cần biến đổi được đưa vào hàm hỗn loạn, và đầu ra là giá trị mới ứng với giá trị được thay thế.

Đồ án sử dụng kết hợp cả hai hướng: Hàm hỗn loạn dùng để sinh ra luật hoán vị cho khối P-Box, và dùng để sinh khối S-Box cho bước thay thế.

1.5 Kết luận chương

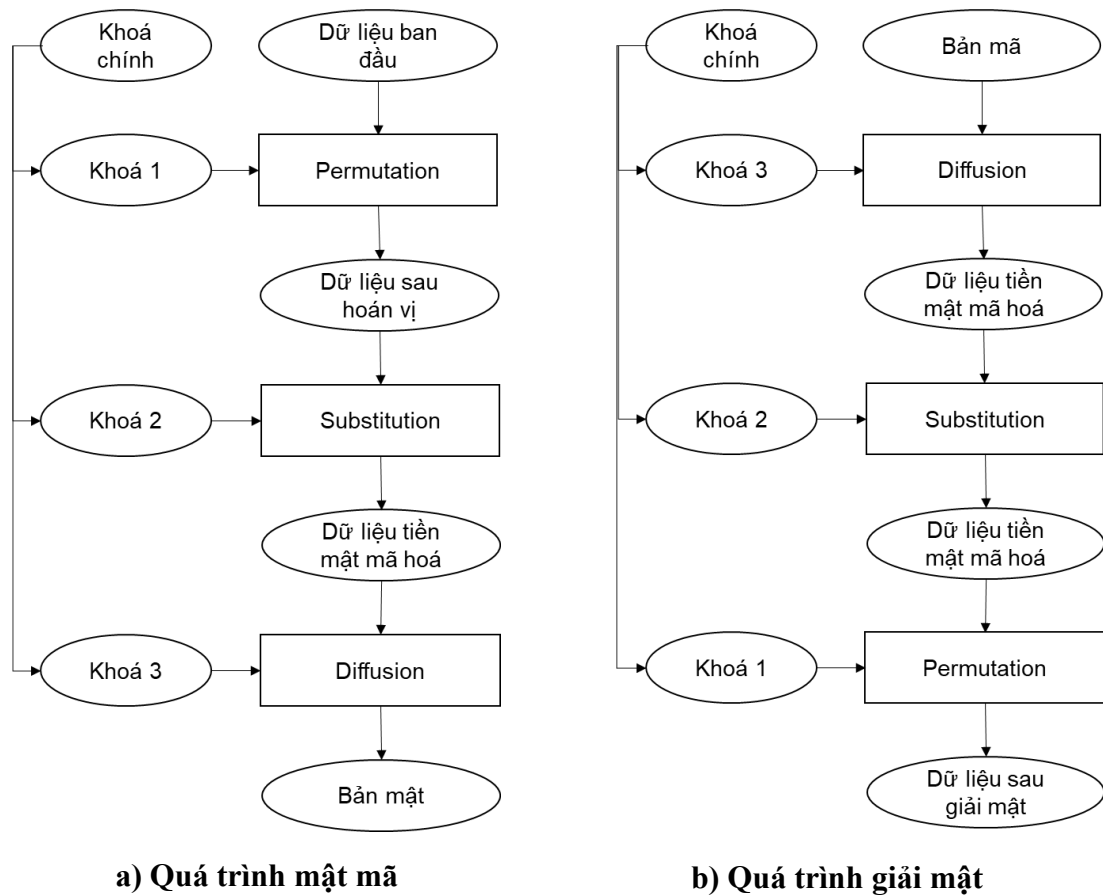
Trong Chương 1, đồ án đã trình bày các cơ sở lý thuyết nền tảng liên quan đến mật mã hoá như mã hoá đầu cuối trao đổi khoá, lý thuyết hỗn loạn, những cách triển khai của lý thuyết hỗn loạn trong mật mã hoá. Sau đó lựa chọn phương pháp triển khai mã hoá đầu cuối và mật mã hoá hỗn loạn. Đây sẽ là những kiến thức nền tảng để thiết kế được hệ mật được trình bày ở Chương 2.

CHƯƠNG 2. THIẾT KẾ HỆ THỐNG MẬT MÃ HOÁ

Trong chương này, đồ án sẽ trình bày việc thiết kế hệ thống mật mã hoá hỗn loạn áp dụng những kiến thức về lý thuyết hỗn loạn và kiến thức về bảo mật đã được nêu ở chương trước, trong đó có quá trình sinh khoá cho các hàm hỗn loạn, và các khối thực hiện mật mã hoá và giải mật mã hoá.

2.1 Sơ đồ hệ thống

Hệ thống mật mã hoá mà tôi sử dụng gồm có 3 khối: Khối hoán vị (Permutation), Khối thay thế (Substitution), Khối khuếch tán (Diffusion). Trong đó quá trình giải mật cũng tương tự quá trình mật mã nhưng làm theo thứ tự ngược lại (Hình 2.1). Ngoài ra, có quá trình sinh khoá cho 3 khối này từ một khoá chính được sinh ra từ quá trình trao đổi khoá bằng ECDH sẽ được nêu trong Mục 2.5.

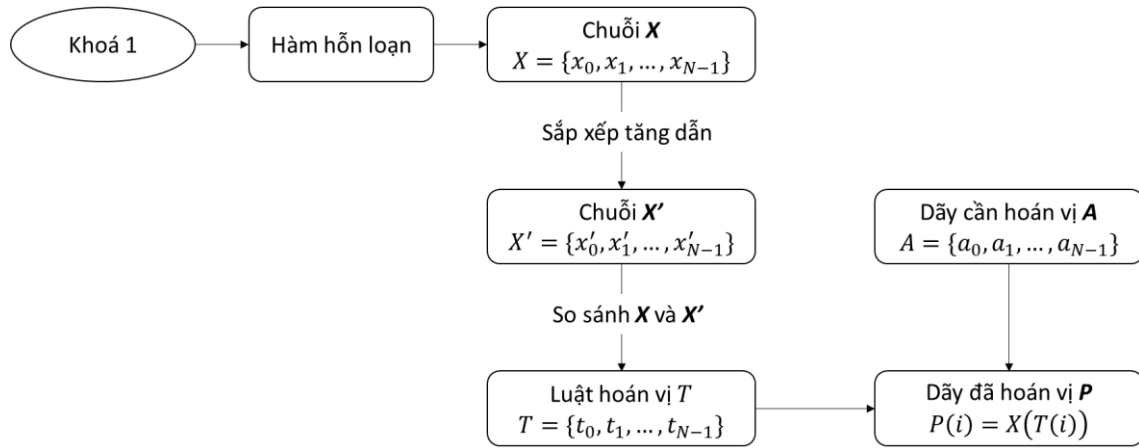


Hình 2.1 Quá trình mật mã và giải mật

2.1.1 Khối hoán vị

Hoán vị là hoán đổi vị trí các đơn vị dữ liệu nhằm phá vỡ cấu trúc thống kê của dữ liệu, tạo ra sự phụ thuộc của bản mã vào khoá mật. Trong mật mã hoá sử dụng hỗn loạn, các hàm hỗn loạn tạo ra luật hoán vị theo một số cách khác nhau. Trong đó phương pháp hoán vị dựa vào biến trạng thái của hàm hỗn loạn là phương pháp đồ án đã sử dụng cho khối hoán vị.

Cơ chế hoán vị có yêu cầu là cho phép khôi phục lại vị trí các điểm dữ liệu ban đầu. Phương pháp tạo ra luật hoán vị ứng dụng hỗn loạn được thực hiện dựa trên dãy các giá trị lấy từ biến trạng thái của hàm hỗn loạn. Các giá trị này được xem là chuỗi giả ngẫu nhiên với giá trị nằm trong phạm vi giá trị của dữ liệu.



Hình 2.2 Khối hoán vị của hệ mật

Các bước thực hiện cho hoán vị như Hình 2.2:

- **Bước 1.** Giả sử chuỗi các giá trị hỗn loạn được tạo ra bởi hàm hỗn loạn là $X = \{x_0, x_1, \dots, x_{N-1}\}$ trong đó N là số kí hiệu cần hoán vị. Do x_i là các giá trị có thể dạng số thực. Do việc hoán vị thực tế là phải là ánh xạ một-một, do vậy các giá trị của X không có các thành phần với giá trị được lặp lại quá một lần. Như vậy, trong quá trình hình thành chuỗi X cần giải quyết các trường hợp giá trị lặp lại ở các phần tử của X .
- **Bước 2.** Thực hiện sắp xếp các giá trị của chuỗi X theo thứ tự tăng dần. Ta được chuỗi $X' = \{x'_0, x'_1, \dots, x'_{N-1}\}$ và luật hoán vị của nó $T = \{t_0, t_1, t_2, \dots, t_{N-1}\}$.
- **Bước 3.** Áp dụng luật hoán vị T cho chuỗi cần hoán vị $Z = \{z_0, z_1, \dots, z_{N-1}\}$ theo công thức $p_i = z_{t_i}$ ta được chuỗi sau hoán vị P .

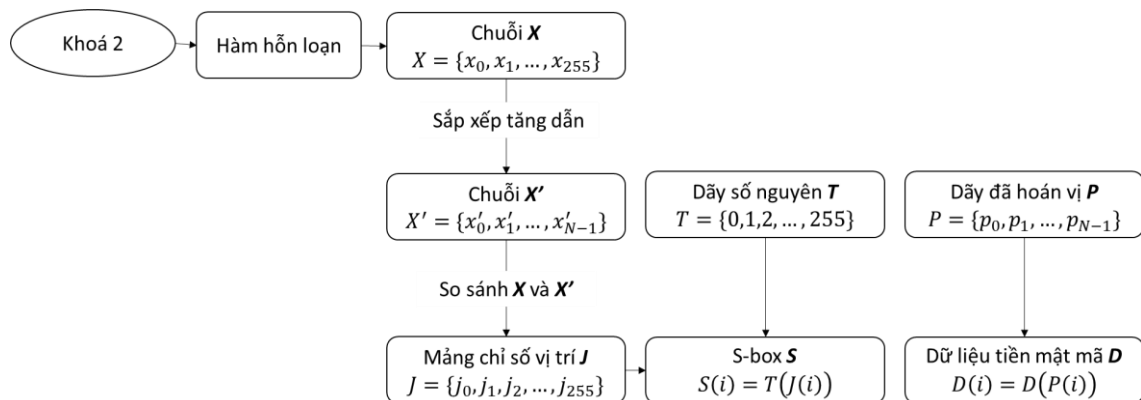
2.1.2 Khối thay thế

Trong khối này, thay thế bằng cách sử dụng một hộp thay thế (S-box) như bảng tra cứu. Kỹ thuật này được sử dụng để ẩn các thuộc tính thống kê của dữ liệu và làm giảm sự tương quan giữa các dữ liệu lân cận.

S-Box được sử dụng làm thành phần phi tuyến tính chính của thuật toán. Mục đích của việc sử dụng S-Box là tạo ra sự phi tuyến tính trong quá trình mật mã và cũng để tạo ra xáo trộn (confusion) và khuếch tán (diffusion) trong bản mật. Việc sử dụng S-Box giúp tăng cường khả năng bảo vệ của hệ thống chống lại sự phá mã tuyến tính và vi phân.

Với tầm quan trọng của S-Box trong hệ thống mật mã khối, việc thiết kế S-Box với hiệu suất mật mã mạnh luôn là mục tiêu của các nhà thiết kế hệ thống mật mã. Nhiều phương pháp xây dựng S-Box đã được đề xuất. Trong quá trình thay thế, mỗi phân tử sẽ được ánh xạ bằng S-Box. S-Box được sử dụng để biến đổi đầu vào bit một cách ngẫu nhiên. Kết quả là, chuỗi bit đầu ra có khả năng chống lại các cuộc tấn công tuyến tính và vi phân. Một số phương pháp tiếp cận như phương pháp phân tích, kỹ thuật đại số, hàm Boolean, phương pháp ánh xạ đa thức bậc ba, và nhóm tam giác đã được áp dụng để xây dựng S-Box.

Trong những năm gần đây, các hệ thống hỗn loạn đã được áp dụng rộng rãi trong thiết kế S-Box vì các đặc điểm mật mã tốt của chúng, chẳng hạn như hành vi giống như ngẫu nhiên, không tuần hoàn và độ nhạy cực cao đối với các điều kiện ban đầu. Thuật toán sinh S-Box của Lu, & Zhu, & Wang được nêu trong [20] là thuật toán sinh S-Box đồ án đã sử dụng.



Hình 2.3 Khối thay thế của hệ mật

Các bước thực hiện để tạo ra S-Box 8×8 như sau:

- **Bước 1.** Đặt dãy $T = \{0,1,2, \dots, 255\}$ chứa 256 số nguyên riêng biệt trong phạm vi $[0,255]$.
- **Bước 2.** Lặp lại hàm hỗn loạn L lần ($L \gg 256$). Loại bỏ $(L - 256)$ phần tử đầu để cải thiện độ nhạy của chuỗi hỗn loạn, kết quả thu được một chuỗi hỗn loạn có độ dài 256 $X = \{x_0, x_1, x_2, \dots, x_{255}\}$.
- **Bước 3.** Sắp xếp chuỗi hỗn loạn X , sau đó thu được mảng chỉ số vị trí $J = \{j_0, j_1, \dots, j_{255}\}$ với $j_i \in \{0,1,2, \dots, 255\}$. Do tính không tuần hoàn và sai lệch của các chuỗi hỗn loạn nên $j_i \neq j_j$ khi $i \neq j$.
- **Bước 4.** Dùng J để tính S : $S(i) = T(J(i))$.

Sau khi sinh được khối thay thế S-Box, ta áp dụng vào lần lượt từng điểm dữ liệu một theo công thức $p_i = S(p_i)$ với p_i là giá trị tại điểm i .

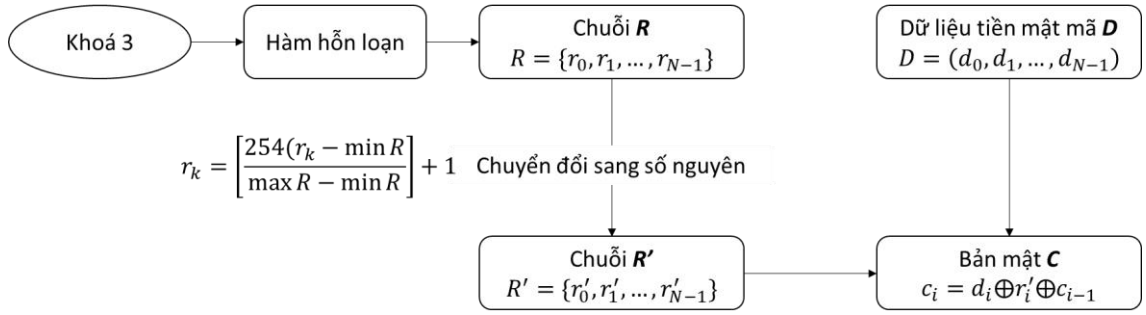
2.1.3 Khối khuếch tán

Các cuộc tấn công lựa chọn bản rõ được thiết kế để phá vỡ hệ thống mật mã hoá bằng cách kiểm tra xem một thay đổi nhỏ trong ảnh rõ có thể ảnh hưởng thế nào đến kết quả mật mã hoá của hệ thống. Một pha khuếch tán hiệu quả có thể làm cho hệ thống mật mã hoá hình ảnh chịu được những kiểu tấn công kiểu này. Để xáo trộn cấu trúc bên trong của dữ liệu, đồ án sử dụng hàm hỗn loạn, phép toán XOR và sử dụng phép thế có lan truyền, nghĩa là kết quả đầu ra phụ thuộc vào tham số điều khiển, giá trị hiện tại và giá trị từ mã trước đó $c_i = f(p_i, \alpha, c_{i-1})$.

Đồ án đã sử dụng hàm hỗn loạn mà giá trị của nó trong phạm vi $[0,1]$. Do đó, sau khi sinh chuỗi hỗn loạn, phải chuyển các giá trị của chuỗi sang phạm vi một byte (từ 0 đến 255). Trong đồ án này, sử dụng hàm hỗn loạn với tham số điều khiển α để sinh ra chuỗi số $R = \{r_i | i = 0..N - 1\}$ và đổi sang chuỗi số nguyên $R' = \{r'_i | i = 0..N - 1\}$ theo công thức (2.1):

$$r'_k = \left\lfloor \frac{254(r_k - \min r_k)}{\max r_k - \min r_k} \right\rfloor + 1 \quad (2.1)$$

Sau đó mật mã từng điểm dữ liệu ở vị trí i là $c_i = f(p_i, r'_i, c_{i-1}) = p_i \oplus r'_i \oplus c_{i-1}$. Quá trình thực hiện khối khuếch tán được biểu diễn như Hình 2.4.



Hình 2.4 Khối khuếch tán của hệ mật

2.2 Hàm hỗn loạn Logistic

Trong 3 khối của hệ mật, đều có sử dụng hàm hỗn loạn. Ba khối có thể sử dụng chung một hàm hỗn loạn, hoặc có thể sử dụng nhiều hàm hỗn loạn khác nhau. Hàm hỗn loạn cũng có thể là một hàm gốc hoặc có thể kết hợp nhiều loại hàm hỗn loạn lại với nhau. Hàm Logistic là một hàm được sử dụng phổ biến trong Mật mã hoá hỗn loạn. Do đó, đồ án này sử dụng hàm Logistic thực hiện hệ mật.

Hàm Logistic có công thức

$$x_{n+1} = f(x_n, \mu) = \mu x_n (1 - x_n) \quad (2.1)$$

Trong đó $\mu \in (0, 4]$ là tham số điều khiển, $x_n \in (0, 1)$ là giá trị của hàm tại vòng lặp n , x_0 là điều kiện đầu của hàm.

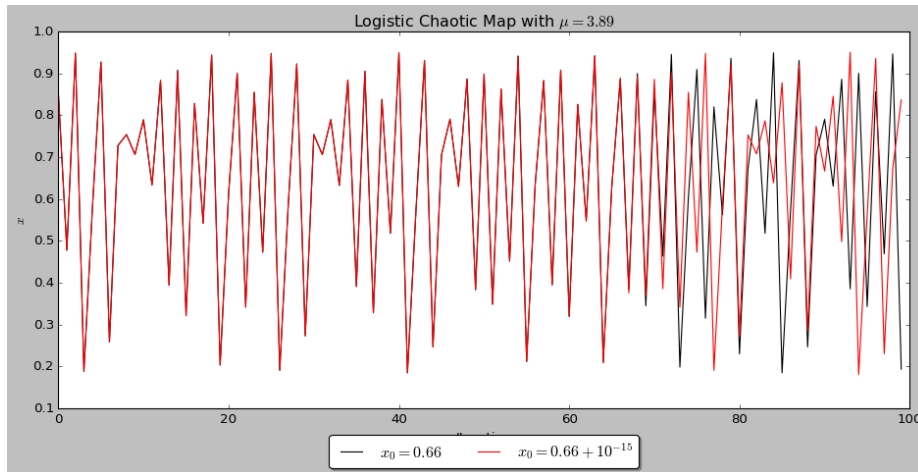
Thông số được dùng để đánh giá mức độ phụ thuộc vào điều kiện đầu của hàm hỗn loạn đó là hệ số lũy thừa Lyapunov.

Hệ số Lyapunov λ được tính bởi công thức (2.2).

$$\lambda = \lim_{n \rightarrow \infty} \frac{1}{n} \left(\sum_{i=1}^n \log |f'(x_i)| \right) \quad (2.2)$$

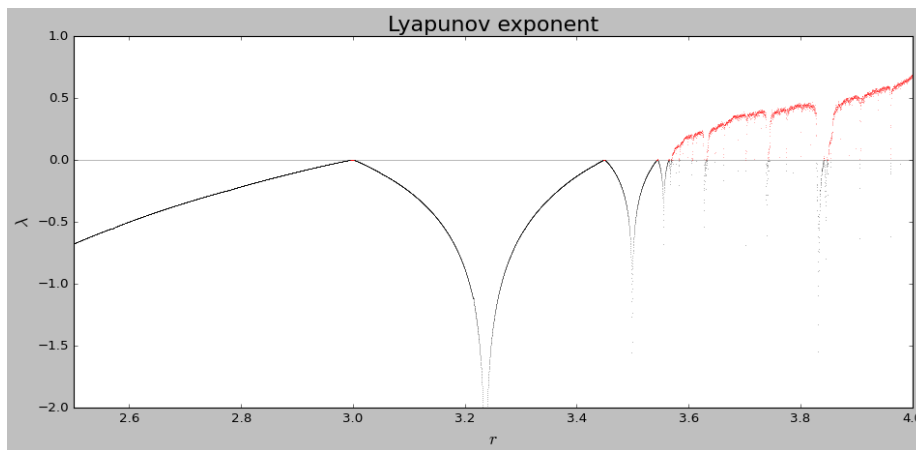
Hệ số lũy thừa Lyapunov âm thể hiện hệ thống ở trạng thái ổn định, hệ số bằng 0 thì hệ dao động có chu kỳ, hệ số này dương cho thấy rằng hiện tượng hỗn loạn tồn tại ở hệ thống động.

Sự phụ thuộc điều kiện đầu của hàm hỗn loạn Logistic có thể được thấy trong Hình 2.5. Ở đó, hai điều kiện đầu vào lần lượt là $x_0 = 0.66$ và $x_0 = 0.66 + 10^{-15}$ sẽ tạo ra các giá trị khác nhau hoàn toàn sau khoảng 70 lần lặp.



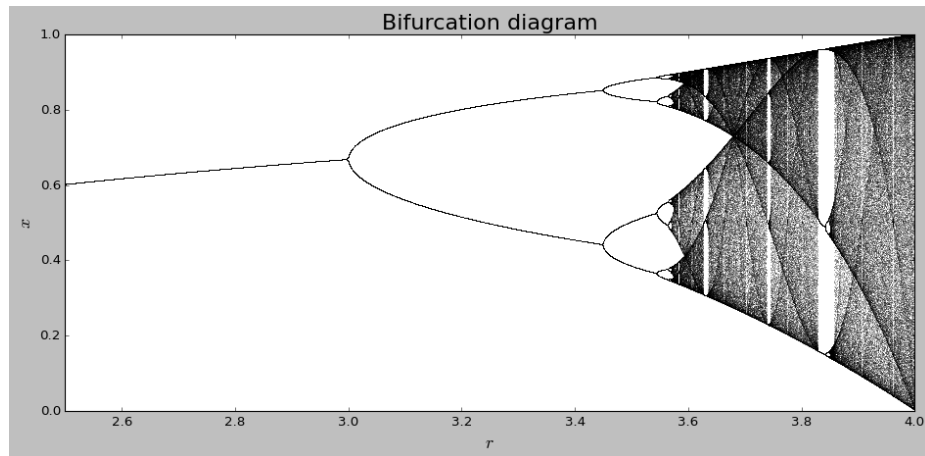
Hình 2.5 Phụ thuộc điều kiện đầu vào của hàm Logistic với $\mu = 3.89$

Để thấy bức tranh tổng quát giá trị của hệ số lũy thừa Lyapunov, việc khảo sát tính toán với từng giá trị của tham số điều khiển được thực hiện. Việc khảo sát các hệ số Lyapunov thường so sánh với giá trị 0 để thấy rõ hàm ở trạng thái hỗn loạn ứng với giá trị của tham số điều khiển. Kết quả cho thấy hệ số Lyapunov của hàm Logistic phụ thuộc vào giá trị tham số điều khiển μ , được biểu thị trong Hình 2.6. Nó cũng cho thấy hàm Logistic xuất hiện hỗn loạn với $\mu > 3.56$.



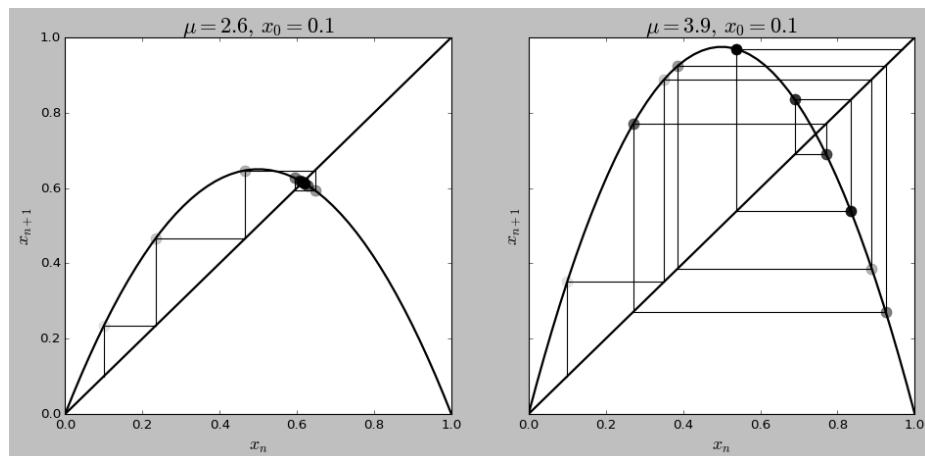
Hình 2.6 Hệ số Lyapunov của hàm Logistic phụ thuộc vào μ

Lược đồ phân nhánh (bifurcation) của hàm Logistic như Hình 2.7 mô tả giá trị của tham số μ thay đổi. Nhìn vào lược đồ, khi thỏa mãn điều kiện hỗn loạn khi $\mu > 3.56$, tại khu vực này phân bố dày đặc các quỹ đạo có chu kỳ cho thấy tồn tại hỗn loạn ứng với giá trị của tham số điều khiển tương ứng.



Hình 2.7 Lược đồ phân nhánh của hàm Logistic

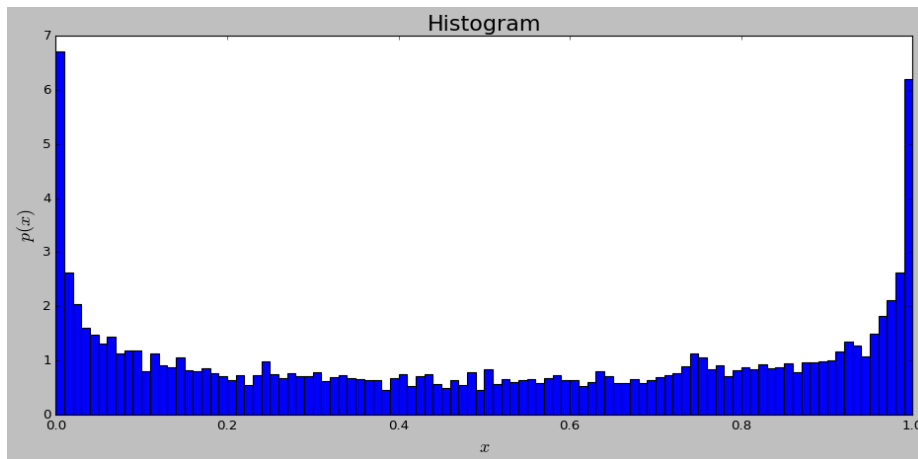
Sự vận động của hàm Logistic trong mặt phẳng pha được biểu diễn trong Hình 2.8. Nhìn vào hình vẽ, ta thấy khi tham số điều khiển là $\mu = 2.6$, giá trị của hàm có xu hướng hội tụ lại một điểm, nhưng với tham số điều khiển là $\mu = 3.9$, giá trị của vận động của hàm hỗn loạn, không có quy tắc, điều này thể hiện đúng tính chất bất quy tắc trong mặt phẳng pha của hàm hỗn loạn.



Hình 2.8 Sự chuyển động của hàm Logistic trong mặt phẳng pha

Một trong các tính chất quan trọng của chuỗi giá trị được tạo ra từ hàm hỗn loạn đó là tính phân bố đều của chuỗi giả ngẫu nhiên. Hình 2.9 cho thấy phân bố giá trị của 10000 giá trị x hàm Logistic được tạo ra với $x_0 = 10^{-15}$ và $\mu = 4$ và phân bố giá trị được tạo ra bởi hàm Logistic là không đồng đều, tỷ lệ giá trị nằm ở gần 0 và 1 nhiều hơn các giá trị khác. Để hàm hỗn loạn có tính phân bố đều, có 2 cách thay đổi hàm Logistic. Đó là tạo ra sự biến đổi tham số điều khiển μ và tạo ra sự thay đổi trong biến hỗn loạn x_n . Trong một số nghiên cứu khác, các hàm Logistic được ghép lại với nhau để tạo ra sự phức tạp. Đồ án này sử dụng hàm Logistic gốc để mật mã hoá mà không

thông qua phép biến đổi nào để giá trị sinh ra được đồng đều.



Hình 2.9 Phân bố của chuỗi giá trị được tạo từ hàm Logistic

2.3 Quá trình mật mã

Đầu vào: Bản rõ $A = \{a_i | i = 0..N - 1\}$, khoá $k = (k_1, k_2, k_3)$ với $k_i = \{x_{0i}, \mu_i\}$

Đầu ra: Bản mật $C = \{c_i | i = 0..N - 1\}$

- **Bước 1.** Sử dụng khoá k_1 để sinh ra luật hoán vị, áp dụng luật hoán vị vào A ta được $P = \{p_0, p_1, \dots, p_{N-1}\}$.
- **Bước 2.** Sử dụng khoá k_2 để sinh khối S-Box, sử dụng khối S-Box, thay thế từng giá trị trong P từ đó tìm ảnh tiền mật mã $D = \{d_0, d_1, \dots, d_{N-1}\}$.
- **Bước 3.** Sử dụng khoá k_3 để thực hiện quá trình khuếch tán, sinh chuỗi số nguyên giả ngẫu nhiên R , sử dụng phép tính XOR cho từng giá trị và giá trị mật mã trước đó: $c_i = d_i \oplus r_i \oplus c_{i-1}$.

Ví dụ 2.1: Mật mã hoá $A = \{5, 1, 6, 2, 10, 15, 6, 4, 3\}$ với khoá $k_1 = \{0.66, 3.89\}$, $k_2 = \{0.12, 3.99\}$, $k_3 = \{0.98, 4\}$.

Sử dụng khoá k_1 sinh chuỗi hỗn loạn X

$X = \{0.8729, 0.4315, 0.9542, 0.1698, 0.5483, 0.9634, 0.1370, 0.4601, 0.9663\}$

Sắp xếp chuỗi X ta được chuỗi X' và quy luật hoán vị T

$X = \{0.1370, 0.1698, 0.4315, 0.4601, 0.5483, 0.8729, 0.9542, 0.9634, 0.9663\}$

$T = \{6, 3, 1, 7, 4, 0, 2, 5, 8\}$

Áp dụng luật hoán vị vào A ta được

$$P = \{6, 2, 1, 4, 10, 5, 6, 15, 3\}$$

Sử dụng khoá k_2 sinh khối S-Box, ta được bảng S-Box được chuyển thành bảng 16×16 như Bảng 2.1. Sử dụng S-Box này để thay thế từng giá trị, ta được

$$D = \{109, 188, 51, 236, 78, 161, 109, 184, 10\}$$

Sử dụng khoá k_3 để sinh ra chuỗi R

$$R = \{0.0784, 0.2890, 0.8219, 0.5854, 0.9708, 0.1133, 0.4020, 0.9616, 0.1478\}$$

Chuyển sang chuỗi số nguyên

$$R' = \{1, 60, 212, 145, 255, 10, 93, 252, 20\}$$

Dùng phép toán XOR, thu được kết quả sau mật mã

$$C = \{0, 236, 91, 145, 41, 166\}$$

Bảng 2.1 Bảng S-Box sinh ra bởi hàm Logistic với $x_0 = 0.12$ và $\mu = 3.99$

183	51	188	10	236	161	109	150	94	124	78	196	56	136	47	184
6	52	87	189	11	237	30	162	110	0	41	151	62	95	125	217
116	79	197	57	25	137	142	176	202	168	84	133	75	121	48	185
7	233	106	147	91	193	53	38	103	88	190	12	15	247	238	18
31	70	163	171	111	1	42	152	63	96	126	250	222	205	241	218
229	117	80	198	58	26	179	157	21	138	143	34	177	203	169	68
101	36	145	73	131	166	174	140	23	114	215	85	4	255	134	76
122	159	181	49	186	8	234	107	148	92	194	54	45	28	39	60
200	82	119	231	104	89	191	13	245	16	248	220	239	227	155	19
32	66	99	71	129	164	172	112	213	2	253	43	243	225	153	64
97	127	211	251	223	209	207	206	208	210	224	242	252	212	128	98
65	154	226	219	244	230	118	81	199	59	27	44	180	158	254	3
214	113	22	139	173	165	130	72	144	35	100	67	33	20	156	178
228	240	204	221	249	170	69	17	246	14	102	37	192	90	146	105
232	120	74	132	83	167	201	175	141	24	115	216	61	40	29	86
5	46	135	55	195	77	123	93	149	108	160	235	9	187	50	182

2.4 Quá trình giải mật

Quá trình giải mật làm tương tự như quá trình mật mã, nhưng làm ngược lại

Đầu vào: Bản mật $C = \{c_i | i = 0..N - 1\}$, khoá $k = (k_1, k_2, k_3)$ với $k_i = \{x_{0i}, \mu_i\}$

Đầu ra: Bản rõ $A = \{a_i | i = 0..N - 1\}$

- **Bước 1.** Sử dụng khoá k_3 để tìm chuỗi số nguyên $R' = \{r'_0, r'_1, r'_2, \dots, r'_{N-1}\}$ bằng hàm hỗn loạn.

- **Bước 2.** Tìm ảnh tiền mã hoá D bằng cách sử dụng phép XOR cho từng giá trị $d_i = c_i \oplus r'_i \oplus c_{i-1}$.
- **Bước 3.** Sử dụng khoá k_2 để sinh khối S-box và tìm nghịch đảo của nó là S^{-1}
- **Bước 4** Sử dụng S^{-1} với từng phần tử của D để tìm ra ảnh sau hoán vị P theo công thức $p_i = S^{-1}(d_i)$.
- **Bước 5** Sử dụng khoá k_1 để sinh ra chuỗi hỗn loạn $X = \{a_0, a_1, a_2, \dots, a_{N-1}\}$, sau đó sắp xếp và tính toán vector hoán vị $T = \{t_0, t_1, t_2, \dots, t_{N-1}\}$, rồi tìm nghịch đảo T^{-1} của T .
- **Bước 6.** Sử dụng T^{-1} để hoán vị P thành vector A ban đầu.

Ví dụ 2.2: Giải mật mã hoá C được mật mã ở ví dụ 2.1 với khoá k_1, k_2, k_3 như ví dụ 2.1

Sử dụng khoá k_3 để sinh ra chuỗi R

$$R = \{0.0784, 0.2890, 0.8219, 0.5854, 0.9708, 0.1133, 0.4020, 0.9616, 0.1478\}$$

Chuyển sang chuỗi số nguyên

$$R' = \{1, 60, 212, 145, 255, 10, 93, 252, 20\}$$

Dùng phép toán XOR để khôi phục lại chuỗi tiền mật mã D

$$D = \{109, 188, 51, 236, 78, 161, 109, 184, 10\}$$

Sử dụng khoá k_2 sinh khối S-Box, ta được bảng S-Box được chuyển thành bảng 16×16 như Bảng 2.1. Tìm nghịch đảo của S-Box này, áp dụng vào D ta được chuỗi đã hoán vị

$$P = \{6, 2, 1, 4, 10, 5, 6, 15, 3\}$$

Sử dụng khoá k_1 sinh chuỗi hỗn loạn A

$$X = \{0.8729, 0.4315, 0.9542, 0.1698, 0.5483, 0.9634, 0.1370, 0.4601, 0.9663\}$$

Sắp xếp chuỗi A ta được chuỗi B và quy luật hoán vị T

$$X' = \{0.1370, 0.1698, 0.4315, 0.4601, 0.5483, 0.8729, 0.9542, 0.9634, 0.9663\}$$

$$T = \{6, 3, 1, 7, 4, 0, 2, 5, 8\}$$

Tìm nghịch đảo của T là T^{-1} , áp dụng vào P

$$E = \{5, 1, 6, 2, 10, 15, 6, 4, 3\} = A$$

2.5 Quá trình sinh khoá

Trong các hệ thống thông tin hiện nay, các dữ liệu đều được biểu diễn dưới dạng các chuỗi byte. Do đó trước khi mật mã hay giải mật, đều phải trải qua quá trình chuyển đổi khoá từ dạng chuỗi byte sang dạng số thực. Trong đề án này cũng vậy, sau quá trình trao đổi, tính toán được khoá chung là một chuỗi 256 bit, sau đó phải chuyển khoá chung 256 bit sang 3 khoá cho khối hoán vị, thay thế và khuếch tán.

- **Bước 1:** Sau quá trình trao đổi khoá, ta thu được khoá chia sẻ giữa 2 bên là S , sau đó sử dụng hàm băm SHA256 để mật mã chuỗi $[S, P_1, P_2]$ (với P_1, P_2 là khoá công khai của 2 bên với nhau. Ta được một số có thể biểu diễn dưới dạng chuỗi byte $F = [f_0, f_1, \dots, f_{31}]$.
- **Bước 2:** Tính khoá $k_1 = \{x_{01}, \mu_1\}$ như công thức (2.2)

$$x_{01} = \frac{f_0 \oplus f_2 \oplus f_4 \dots f_{14} \oplus f_{16}}{255} \quad (2.2)$$

$$\mu_1 = 3.7 + \frac{f_1 \oplus f_3 \oplus f_5 \dots f_{13} \oplus f_{15}}{255} \times 0.3$$

- **Bước 3:** Tính khoá $k_2 = \{x_{02}, \mu_2\}$ như công thức (2.4)

$$x_{02} = \frac{f_{18} \oplus f_{20} \oplus f_{22} \dots f_{28} \oplus f_{30}}{255} \quad (2.3)$$

$$\mu_2 = 3.7 + \frac{f_{17} \oplus f_{19} \oplus f_{21} \dots f_{29} \oplus f_{31}}{255} \times 0.3$$

- **Bước 4:** Tính khoá $k_3 = \{x_{03}, \mu_3\}$ như công thức (2.4)

$$x_{03} = \frac{f_0 \oplus f_2 \oplus f_4 \dots f_{28} \oplus f_{30}}{255} \quad (2.4)$$

$$\mu_2 = 3.7 + \frac{f_1 \oplus f_3 \oplus f_5 \dots f_{29} \oplus f_{31}}{255} \times 0.3$$

```
std::vector<double> key(const std::vector<uint8_t> &digest) {
    doubles key;
    byte x = digest[0];
    byte m = digest[1];
    for (auto i = 0; i < digest.size(); i += 2) {
        x = x ^ digest[i];
    }
}
```

```

        m = m ^ digest[i + 1];
    }

    key.push_back(x / 255.0);
    key.push_back(3.7 + m / 255.0 * 0.3);
    if ( key[0] <= 0 || key[0] >= 1) {
        key[0] = 0.5;
    }
    if (key[1] < 3.7 || key[1] > 4) {
        key[1] = 4;
    }
    return key;
}

key1 = key(bytes(hash.begin(), hash.begin() + hash.size() / 2));
key2 = key(bytes(hash.begin() + hash.size() / 2, hash.end()));
key3 = key(hash);

```

2.6 Kết luận chương

Trong Chương 2, đồ án đã thực hiện thiết kế được hệ mật sử dụng lý thuyết hỗn loạn. Đây là thành phần quan trọng nhất trong một ứng dụng có sử dụng mã hoá đầu cuối. Hệ mật này sẽ được sử dụng để thiết kế một ứng dụng mã hoá đầu cuối được trình bày ở chương tiếp theo.

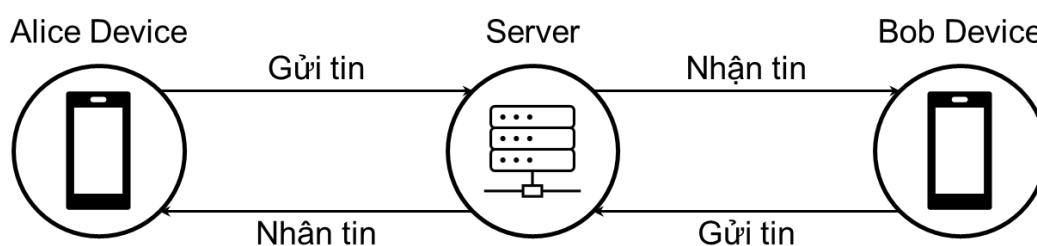
CHƯƠNG 3. TRIỂN KHAI ỨNG DỤNG MÃ HOÁ ĐẦU CUỐI

Sau khi có khối mật mã hoá, ở chương này, đề án triển khai việc thiết kế máy chủ và ứng dụng android của ứng dụng nhắn tin sử dụng mã hoá đầu cuối. Đây là một ứng dụng thử nghiệm. Máy chủ dùng để lưu trữ các thông tin người dùng, các đoạn hội thoại giữa các người dùng. Ứng dụng android thực hiện chức năng sinh khoá, trao đổi khoá, và mật mã, giải mật mã hoá các tin nhắn của người dùng.

3.1 Phân tích ứng dụng

3.1.1 Tổng quan ứng dụng

Ứng dụng chat bao gồm hai phần, đó là phần máy chủ và phần ứng dụng di động như Hình 3.1. Người dùng sẽ gửi yêu cầu đến máy chủ, máy chủ sẽ nhận yêu cầu, xử lý và phản hồi các kết quả về phía ứng dụng do người dùng sử dụng. Trong đề tài này, tôi sử dụng ứng dụng được lập trình dựa trên nền tảng Android, vì Android là một hệ điều hành phổ biến nhất hiện nay nên ta dễ dàng tiếp cận đến.



Hình 3.1 Kiến trúc hệ thống

3.1.2 Phân tích đối tượng người dùng

Người dùng ứng dụng là bất kì ai tải và sử dụng ứng dụng. Người dùng có thể đăng kí, đăng nhập vào ứng dụng, xem các tin nhắn đã nhắn, tìm kiếm người dùng và nhắn tin qua lại cho nhau.

3.1.3 Yêu cầu chức năng

3.1.3.1 Đăng kí

Đây là bước đầu tiên người dùng phải làm khi cài đặt ứng dụng. Để có thể đăng ký người dùng phải nhập họ tên, email và mật khẩu.

3.1.3.2 Đăng nhập

Sau khi người dùng đã được cấp tài khoản có thể đăng nhập vào ứng dụng để sử dụng. Để đăng nhập, người sử dụng phải nhập email, mật khẩu đã đăng kí trước đó. Nếu đăng nhập trên thiết bị cá nhân thành công thì có thể duy trì đăng nhập để không đăng nhập vào lần sử dụng tiếp theo.

3.1.3.3 Nhắn tin

Sau khi đăng nhập, người dùng có thể tìm kiếm, nhắn tin qua lại với nhau

3.1.4 Yêu cầu phi chức năng

3.1.4.1 Về hiệu năng

Về mặt tốc độ xử lý, ứng dụng cần đảm bảo có tốc độ nhanh, thời gian xử lý yêu cầu nhanh chóng, không gây ra sự khó chịu cho người sử dụng. Trong trường hợp có nhiều người cùng truy cập tại cùng một thời điểm, tốc độ của ứng dụng vẫn đạt ở mức chấp nhận được.

Về mặt độ trễ, yêu cầu ứng dụng cần có độ trễ thấp.

Về số lượng người dùng, yêu cầu ứng dụng cần đáp ứng được số lượng người dùng.

3.1.4.2 Về tính bảo mật

Ứng dụng cần phải đảm bảo yếu tố bảo mật tuyệt đối. Khi người dùng thực hiện nhắn tin, lấy những dữ liệu cần thiết, cần dựa vào access token mà máy chủ gửi về lúc đăng nhập để xác định người dùng. Trong quá trình trao đổi tin nhắn, các tin nhắn cần được mã hoá đầu cuối, chỉ đọc được ở phía người dùng ứng dụng.

3.1.4.3 Về tính bảo trì và giao diện

Đối với người lập trình, các chức năng, giao diện đòi hỏi cần phải dễ bảo trì, sửa đổi, bổ sung.

Đối với người dùng, ứng dụng cần có giao diện trực quan, thân thiện, dễ sử dụng, dễ thao tác.

3.2 Thiết kế hệ thống phía máy chủ

Máy chủ được thiết kế bằng framework nestjs. Các đối tượng quản lý được chia

- Bảng **conversations**: Dùng để lưu trữ tin nhắn mới nhất giữa các người dùng.
- Bảng **infomations**: Khi người dùng đăng nhập vào khung chat, quá trình nhắn tin sẽ thông qua socket. Bảng này sẽ lưu thông tin các Socket ID của người dùng để khi nhắn tin, có thể gửi tin nhắn đến chính xác người cần gửi.

3.2.2 Xây dựng API

3.2.2.1 User

API User bao gồm các chức năng như

- Lấy thông tin các người dùng
- Tìm người dùng bằng id
- Tìm người dùng bằng email

a) Lấy thông tin tất cả người dùng

- Action: GET baseurl/v1/user
- Header: Authentication: Bearer {access_token}
- Responses:

Code	Mô tả
200	<p>Thực thi thành công, trả về thông tin tất cả người dùng</p> <pre>[{ "id": 0, "name": "string", "email": "string", "publicKey": "string" }, ...]</pre>
405	Xác thực người dùng thất bại

b) Tìm người dùng bằng id

- Action: GET baseurl/v1/user/{id}
- Header: Authentication: Bearer {access_token}
- Parameters:

Tên	Mô tả
id (integer)	ID của người dùng cần tìm kiếm, có kiểu dữ liệu là số nguyên

- Responses:

Code	Mô tả
200	Thực thi thành công, trả về thông tin người dùng <div> <pre>{ "id": 0, "name": "string", "email": "string", "publicKey": "string" }</pre> </div>
404	Không tìm thấy người dùng
405	Xác thực người dùng thất bại

c) Tìm kiếm người dùng bằng email

- Action: GET baseurl/v1/user/email/email
- Header: Authentication: Bearer {access_token}
- Parameters:

Tên	Mô tả
email (string)	Email của người dùng

- Responses:

Code	Mô tả
200	Thực thi thành công, trả về thông tin người dùng <div> <pre>{ "id": 0, "name": "string", "email": "string", }</pre> </div>

	<pre>"publicKey": "string" }</pre>
404	Không tìm thấy người dùng
405	Xác thực người dùng thất bại

3.2.2.2 Message

API Message bao gồm chức năng lấy tất cả tin nhắn giữa 2 người dùng

- Action: GET baseurl/v1/message/{id}
- Header: Authentication: Bearer {access_token}
- Parameters:

Tên	Mô tả
id (integer)	ID của người dùng cần lấy tất cả tin nhắn

- Responses:

Code	Mô tả
200	Thực thi thành công, trả về thông tin người dùng <pre>[{ "from": 0, "to": 1, "content": "string", "type": "text", "timestamp": "2020-01-01T00:00:00.000Z" }, ...]</pre>
405	Xác thực người dùng thất bại

3.2.2.3 Conversation

API Conversation bao gồm chức năng lấy tất cả đoạn hội thoại giữa người dùng với người dùng khác

- Action: GET baseurl/v1/conversation

- Header: Authentication: Bearer {access_token}
- Responses:

Code	Mô tả
200	Thực thi thành công, trả về thông tin người dùng <pre>[{ "userId": 0, "toUserId": 1, "content": "string", "type": "text", "timestamp": "2020-01-01T00:00:00.000Z" }, ...]</pre>
405	Xác thực người dùng thất bại

3.2.2.4 Auth

API Auth bao gồm các chức năng như

- Đăng kí
- Đăng nhập
- Đăng xuất
- Lấy thông tin người dùng
- Refresh token

a) Đăng kí

- Action: POST baseurl/v1/auth/register
- Parameters:

Tên	Mô tả
body (object)	Các thông tin để đăng kí người dùng <pre>{ "name": "string" "email": "string" "password": "string",</pre>

	<pre> "key": { "publicKey": "string", "privateKey": "string" } </pre>
--	---

- Responses:

Code	Mô tả
201	Đăng kí người dùng thành công
400	Email đăng kí đã tồn tại

b) Đăng nhập

- Action: POST baseurl/v1/auth/login
- Parameters:

Tên	Mô tả
body (object)	<p>Các thông tin để người dùng đăng nhập</p> <pre> { "email": "string" "password": "string", "key": { "publicKey": "string", "privateKey": "string" } } </pre>

- Responses:

Code	Mô tả
200	<p>Đăng nhập thành công</p> <pre> { "email": "string" "password": "string", "token": "string", "refreshToken": "string" } </pre>

	}
401	Sai email hoặc mật khẩu

c) Đăng xuất

- Action: GET baseurl/v1/auth/logout
- Header: Authentication: Bearer {access_token}
- Responses:

Code	Mô tả
200	Đăng xuất thành công
401	Xác thực người dùng thất bại

d) Lấy thông tin người dùng đang đăng nhập

- Action: GET baseurl/v1/auth/profile
- Responses:

Code	Mô tả
200	<p>Lấy thông tin người dùng</p> <pre>{ "id": 0, "name": "string", "email": "string" "key": { "publicKey": "string", "privateKey": "string" }, "refreshToken": "string" }</pre>
401	Xác thực người dùng thất bại

e) Refresh Token

- Action: GET baseurl/v1/auth/profile
- Header: Authentication: Bearer {refresh_token}
- Responses:

Code	Mô tả
200	Refresh token thành công <pre> { "id": 0, "name": "string", "email": "string" "key": { "publicKey": "string", "privateKey": "string" }, "refreshToken": "string" }</pre>
401	Xác thực người dùng thất bại

3.2.3 Xây dựng Socket

3.2.3.1 Connection

Lắng nghe sự kiện có máy khách kết nối đến socket. Khi có máy khách kết nối đến, dựa vào access token mà máy khách gửi tới, xác định được người dùng đang kết nối tới và lưu socket id của người dùng đó vào trong bảng *infomations* trong CSDL

3.2.3.2 Disconnection

Lắng nghe sự kiện có máy khách ngắt kết nối đến socket. Khi máy khách ngắt kết nối, dựa vào access token mà máy khách gửi tới, xác định được người dùng đang ngắt kết nối và xoá socket id của người dùng đó trong bảng *infomations* trong CSDL

3.2.3.3 Send Message và Receive Message

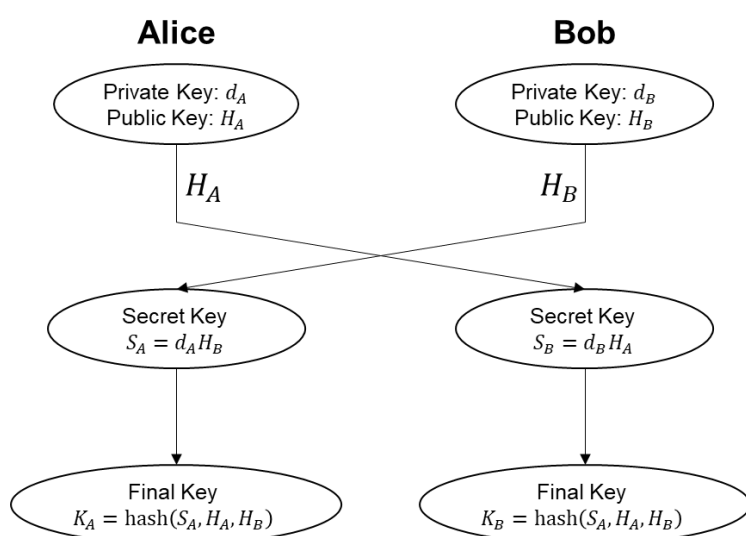
Lắng nghe sự kiện “**send_message**” từ máy khách đang kết nối. Xác thực xem người dùng nào gửi sự kiện này, và người dùng muốn gửi tới ai. Dựa vào 2 biến trên, gửi tin nhắn đến đúng đối tượng thông qua “**receive_message**”, và trả về tin phản hồi cho máy khách gửi sự kiện.

3.3 Thiết kế ứng dụng phía máy khách

3.3.1 Sơ đồ hệ thống

3.3.1.1 Quá trình trao đổi khoá mật

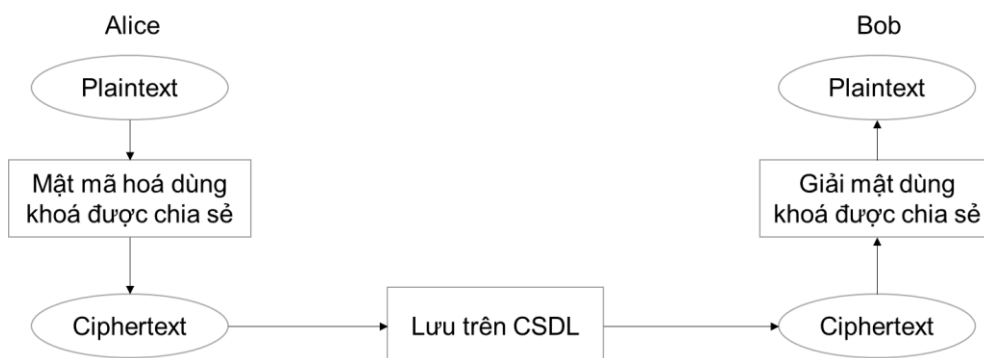
Hình 3.3 là quá trình trao đổi khoá giữa 2 thiết bị. Sau khi đăng nhập, máy chủ sẽ gửi lại khoá công khai và khoá bí mật cho máy khách để thực hiện mã hoá đầu cuối. Để đảm bảo tính an toàn cho khoá bí mật khi được lưu trên CSDL, đồ án đã sử dụng mật khẩu của người dùng để mật mã hoá khoá mật. Trong quá trình trao đổi dữ liệu, máy khách sẽ sử dụng khoá bí mật của mình, và khoá công khai của đối phương để sinh ra một khoá chia sẻ giống nhau ở 2 bên. Tiếp theo sẽ sử dụng hàm băm, để băm khoá bí mật, khoá công khai của mình và đối phương để sinh ra khoá dùng để mật mã hoá tin nhắn (final key).



Hình 3.3 Quá trình trao đổi khoá

3.3.1.2 Quá trình trao đổi tin nhắn

Trong quá trình trao đổi tin nhắn, người dùng sẽ dùng khoá final key giữa 2 người chia sẻ để mật mã hoá và giải mật tin nhắn được gửi và nhận giữa 2 người như Hình 3.4



Hình 3.4 Quá trình trao đổi tin nhắn

3.3.2 Cấu trúc ứng dụng

Ứng dụng thử nghiệm được triển khai theo mô hình MVVM (Model View ViewModel) với Clean Architecture (kiến trúc sạch). Đây là một mô hình thiết kế ứng dụng được sử dụng rộng rãi hiện nay.

3.3.2.1 Giới thiệu về mô hình MVVM và Clean Architecture

a) Mô hình MVVM

MVVM là một mẫu thiết kế cấu trúc phân tách các đối tượng của một project thành ba nhóm riêng biệt [21]:

- Model: là các đối tượng giúp truy xuất và thao tác trên dữ liệu thực sự.
- View: là phần giao diện của ứng dụng để hiển thị dữ liệu và nhận tương tác của người dùng. Một điểm khác biệt so với các ứng dụng truyền thống là View trong mô hình này tích cực hơn. Nó có khả năng thực hiện các hành vi và phản hồi lại người dùng thông qua tính năng binding, command
- ViewModel: Lớp trung gian giữa View và Model. Nó chứa các mã lệnh cần thiết để thực hiện data binding, command.

MVVM có nhiều ưu điểm như [22]:

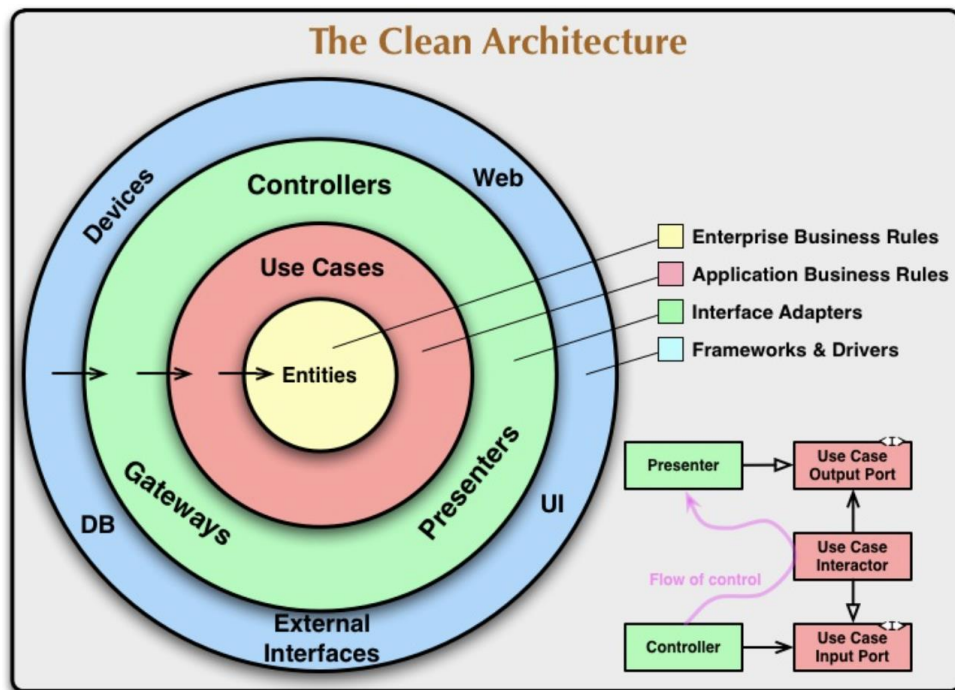
- Khả năng bảo trì: Việc tách biệt rõ ràng các loại mã khác nhau sẽ giúp ta dễ dàng đi vào một hoặc một số phần chi tiết và tập trung hơn và thực hiện các thay đổi mà không cần lo lắng.
- Khả năng kiểm thử: Với MVVM, mỗi đoạn mã chi tiết hơn và nếu nó được triển khai đúng thì các phần phụ thuộc bên ngoài và bên trong của bạn sẽ nằm trong các đoạn mã riêng biệt từ các phần có logic cốt lõi mà bạn muốn kiểm thử. Điều đó làm cho việc viết các unit test dựa trên logic cốt lõi dễ dàng hơn rất nhiều. Đảm bảo rằng ứng dụng hoạt động đúng khi được viết và tiếp tục hoạt động ngay cả khi mọi thứ thay đổi trong quá trình bảo trì.
- Khả năng mở rộng, nâng cấp: MVVM giúp khả năng thay thế hoặc thêm các đoạn mã mới làm những việc tương tự vào đúng vị trí trong kiến trúc mà không ảnh hưởng đến những đoạn mã cũ.

b) Kiến trúc Clean Architecture

Clean Architecture là hướng dẫn kiến trúc hệ thống do Robert C.Martin (Uncle Bob)

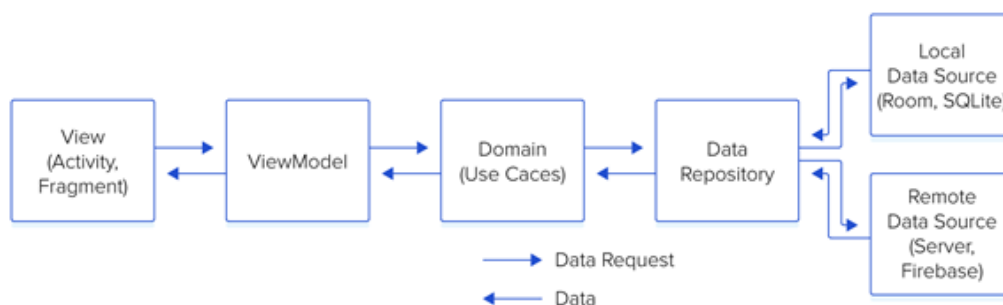
đề xuất xuất từ nhiều hướng dẫn kiến trúc như Hexagonal Architecture, Onion Architecture, v.v... trong nhiều năm. Đây là một trong những nguyên tắc được các kỹ sư phần mềm tuân thủ để xây dựng phần mềm có thể mở rộng, kiểm thử được và có thể bảo trì, giúp giảm thiểu nguồn nhân lực cần thiết để xây dựng, duy trì hệ thống.

Clean Architecture có 4 lớp như Hình 3.5. Quy tắc chính cho Clean Architecture là Quy tắc phụ thuộc Quy tắc phụ thuộc nói rằng các phụ thuộc mã nguồn chỉ có thể trở vào trong. Điều này có nghĩa là không có gì trong một vòng tròn bên trong có thể biết bất cứ điều gì về một cái gì đó trong một vòng tròn bên ngoài. Tức là vòng tròn bên trong không được phụ thuộc vào bất cứ thứ gì ở vòng ngoài. Đây là quy tắc quan trọng làm nên hiệu quả của kiến trúc này.



Hình 3.5 Kiến trúc Clean Architecture

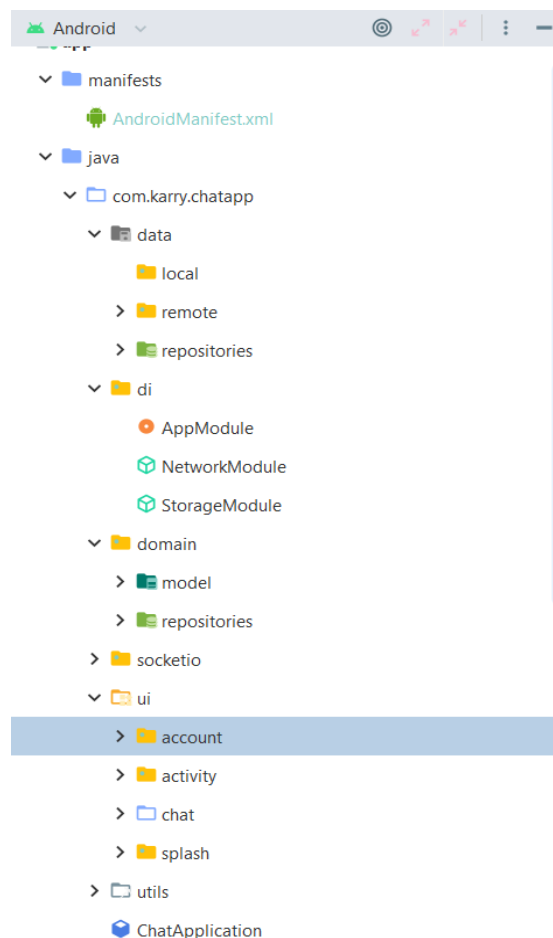
3.3.2.2 Cấu trúc của ứng dụng thử nghiệm



Hình 3.6 Cấu trúc của ứng dụng thử nghiệm

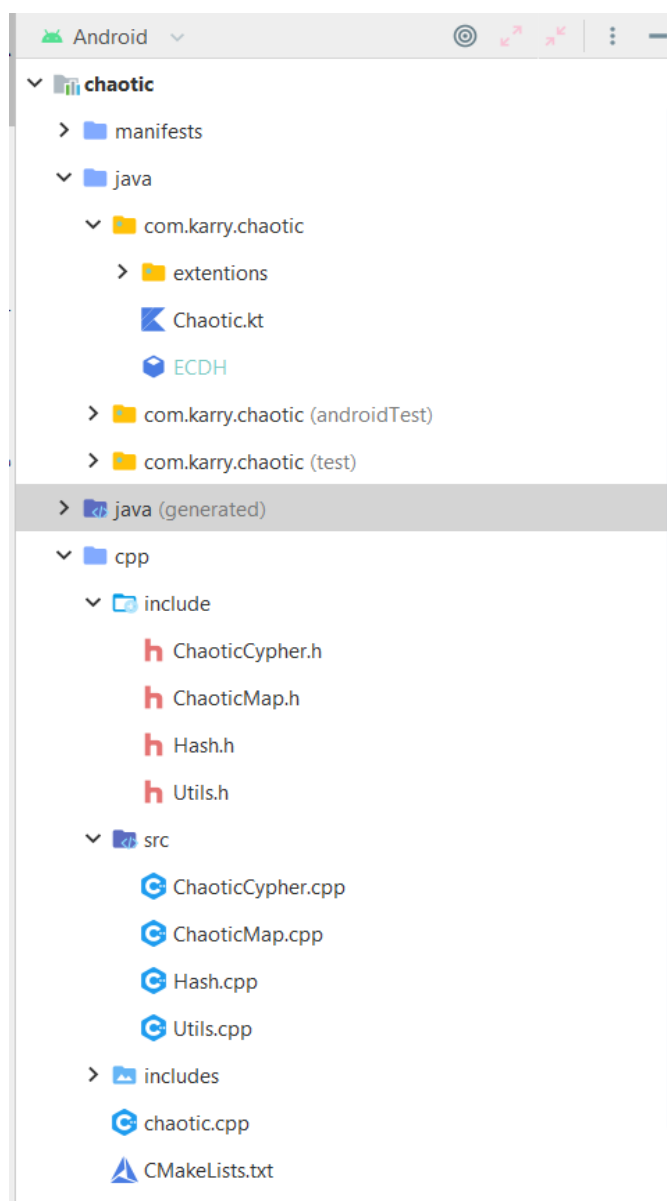
Từ những tìm hiểu về mô hình MVVM và Clean Architecture ở trên, tôi đã sử dụng cấu trúc ứng dụng như Hình 3.6, và có cấu trúc thư mục của project như Hình 3.7. Trong đó:

- Các đối tượng trong package data chịu trách nhiệm lấy dữ liệu từ internet hoặc của bản thân chính ứng dụng đó và triển khai các interface thực hiện lấy dữ liệu trong package domain
- Các đối tượng trong package domain: Chứa các model của các đối tượng được sử dụng trong ứng dụng như User, Message và các interface chức năng ứng dụng.
- Các đối tượng trong package ui: Được chia theo các chức năng nhỏ như Đăng nhập, Đăng Ký,... Ngoài ra, package còn chứa View Model để giao tiếp với UI, các Use Case để thực hiện các chức năng logic của ứng dụng.
- Package utils: Chứa các hàm, các biến cần thiết, được sử dụng ở nhiều nơi



Hình 3.7 Cấu trúc thư mục của project

Ngoài ra, trong project còn có một module tên là chaotic, module này dùng để triển khai hệ mật bao gồm hệ mật hỗn loạn và trao đổi khoá sử dụng ECDH. Trong đó, hệ mật hỗn loạn sử dụng NDK để triển khai code.



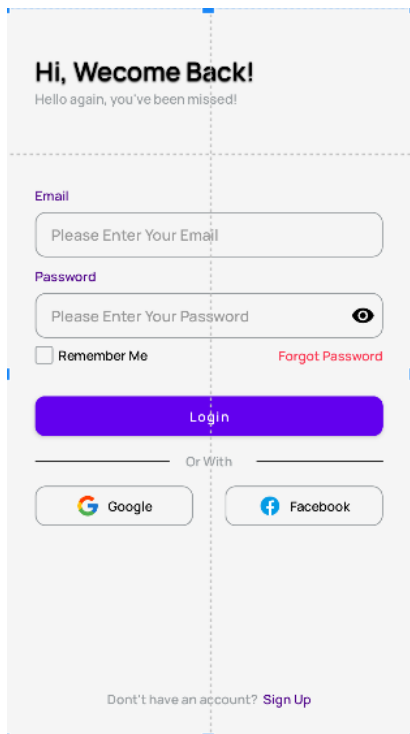
Hình 3.8 Module chaotic

3.3.3 Giao diện người dùng

Sau khi xây dựng được cấu trúc ứng dụng, tiếp theo là việc xây dựng giao diện người dùng và những logic xử lý của ứng dụng.

3.3.3.1 Màn hình đăng nhập, đăng ký

Giao diện đăng nhập và đăng ký cho người dùng được thiết kế như Hình 3.9.



Hi, Welcome Back!
Hello again, you've been missed!

Email
Please Enter Your Email

Password
Please Enter Your Password

☐ Remember Me [Forgot Password](#)

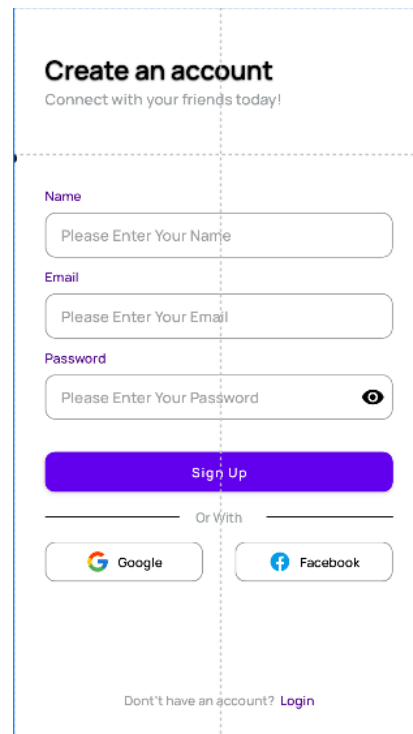
Login

Or With

Google Facebook

Don't have an account? [Sign Up](#)

a) Màn hình đăng nhập



Create an account
Connect with your friends today!

Name
Please Enter Your Name

Email
Please Enter Your Email

Password
Please Enter Your Password

Sign Up

Or With

Google Facebook

Don't have an account? [Login](#)

b) Màn hình đăng kí

Hình 3.9 Màn hình đăng nhập, đăng ký của ứng dụng

3.3.3.2 Các màn hình liên quan đến nhắn tin



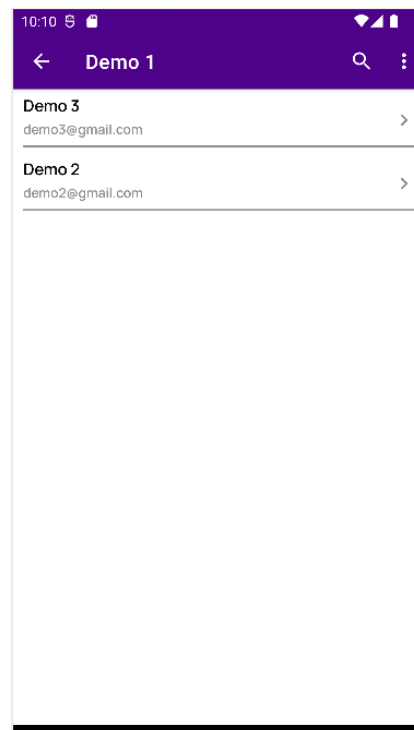
10:09

Demo 1

Demo 2 03/08/22 16:37

Hello

a) Màn hình danh sách hội thoại



10:10

Demo 1

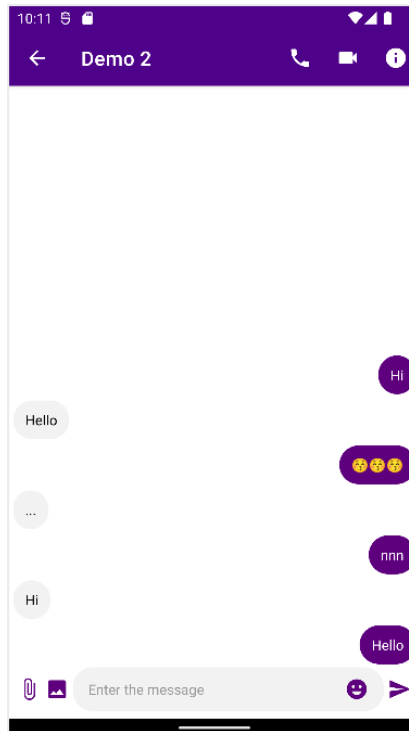
Demo 3

demo3@gmail.com

Demo 2

demo2@gmail.com

b) Màn hình tin nhắn mới



c) Màn hình nhắn tin

Hình 3.10 Màn hình liên quan đến nhắn tin

Hình 3.10 là giao diện người dùng sẽ sử dụng khi nhắn tin.

3.4 Kết luận chương

Trong Chương 3, đề án để triển khai một ứng dụng mã hoá đầu cuối sử dụng những kiến thức đã được tìm hiểu từ Chương 1 và hệ mật được thiết kế ở Chương 2. Sau khi thiết kế được ứng dụng, việc kiểm thử, đánh giá hiệu quả của hệ mật, của ứng dụng là một việc cần thiết. Quá trình kiểm thử và đánh giá được trình bày ở chương tiếp theo.

CHƯƠNG 4. KIỂM THỬ VÀ ĐÁNH GIÁ

Với hệ mật được thiết kế ở Chương 2 và ứng dụng mã hoá đầu cuối được thiết kế ở Chương 3, trong chương này đồ án sẽ trình bày quá trình kiểm thử và đánh giá hệ mật và ứng dụng mã hoá đầu cuối. Trong đó, kịch bản kiểm thử sẽ bao gồm tất cả các khả năng có thể xảy ra khi người dùng sử dụng.

4.1 Kiểm thử và đánh giá khối mật mã hoá

4.1.1 Kiểm thử

Bảng 4.1 là kịch bản kiểm thử của khối mật mã hoá trong ứng dụng.

Bảng 4.1 Kịch bản kiểm thử khối mật mã hoá

ID	Mô tả test case	Thứ tự thực hiện	Test data	Kết quả mong đợi	Kết quả thực tế	Pass hoặc Fail
TC01	Mật mã hoá	1. Nhập vào một chuỗi bất kì làm plaintext 2. Nhập vào một khoá bất kì 3.Ấn Enter để mật mã hoá	Plaintext: Hello World! (Dạng hex: 48 65 6c 00 6f 20 57 00 72 00 64 21) Key: Secret	Plaintext được mã hoá sang dạng không đọc được	Chuỗi dạng hex: a2 4a a2 d4 79 34 27 6a 27 42 b1 33 Chuỗi khác chuỗi ban đầu	Pass
TC02	Giải mật mã hoá với khoá đúng	1. Nhập vào chuỗi đã được mật mã hoá dưới dạng chuỗi hex 2. Nhập vào một khoá đúng 3.Ấn Enter để giải mật mã hoá	Cyphertext: Dạng hex: a2 4a a2 d4 79 34 27 6a 27 42 b1 33) Key: Secret	Chuỗi được giải mật giống chuỗi ban đầu	Chuỗi dạng hex: 48 65 6c 00 6f 20 57 00 72 00 64 21, chuyển sang dạng ASCII: Hello World!. Giống chuỗi ban đầu	Pass
TC03	Giải mật mã hoá với khoá sai	1. Nhập vào chuỗi đã được mật mã hoá dưới dạng chuỗi hex 2. Nhập vào một khoá sai 3.Ấn Enter để giải mật mã hoá	Cyphertext: Dạng hex: a2 4a a2 d4 79 34 27 6a 27 42 b1 33) Key: Hi	Chuỗi được giải mật khác chuỗi ban đầu	Chuỗi dạng hex: 81 56 85 81 7c aa 61 56 86 56 88 e7. Khác chuỗi ban đầu	Pass

4.1.2 Đánh giá hệ mật

4.1.2.1 Thời gian mật mã và giải mật

Để đánh giá thời gian mật mã và giải mật, tôi sinh ra một dãy gồm 10 000 000 phần tử ngẫu nhiên, và thử đi thử lại 10 lần. Thiết bị sử dụng để thực hiện: CPU: Intel Core i7 6700HQ, RAM 16 GB. Kết quả được thể hiện ở Bảng 4.2.

Bảng 4.2 Thời gian mật mã và giải mật của hệ mật

Lần	Thời gian mật mã (ms)	Thời gian giải mật (ms)	Pass hoặc Fail
1	2928	2956	Pass
2	2963	2885	Pass
3	2869	2920	Pass
4	2869	2897	Pass
5	2943	3048	Pass
6	2936	2904	Pass
7	2905	3067	Pass
8	3166	3262	Pass
9	3140	3114	Pass
10	3163	3239	Pass
Trung bình	2988	3029	

4.1.2.2 Không gian khoá

Một hệ thống mã hóa an toàn phải có một không gian khóa lớn có thể chống lại các cuộc tấn công một cách thích hợp. Kích thước không gian khóa bắt buộc phải lớn hơn 2^{100} để cung cấp mức độ bảo mật cao. Đối với mật mã hình ảnh được nhóm trình bày, các tham số và giá trị ban đầu của hàm hỗn loạn là khóa bí mật. Hàm Logistic có một giá trị ban đầu và một tham số. Trong hệ thống, sử dụng 3 lần hàm Logistic để sinh chuỗi hỗn loạn, vì vậy sẽ có 3 giá trị ban đầu và 3 tham số điều khiển thỏa mãn điều kiện của hàm Logistic. Với độ chính xác là 10^{-15} , ta có không gian khóa là $10^{15.6} = 10^{90} \approx 2^{300}$ lớn hơn rất nhiều so với 2^{100} . Do đó, phương pháp được nhóm giới thiệu có không gian khóa đủ để chống lại các kiểu tấn công bruteforce khác nhau.

4.2 Kiểm thử các chức năng phía máy chủ

Để kiểm thử phía máy chủ, tôi sử dụng phần mềm Postman để thực thi các API và kết nối tới socket của máy chủ. Kết quả kiểm thử như Bảng 4.3.

4.2.1 Kiểm thử trên máy local

Bảng 4.3 Kịch bản kiểm thử máy chủ**TC04 – Đăng kí với email không hợp lệ**

Mô tả: Người dùng thực hiện đăng kí, trong đó người dùng nhập vào một email không hợp lệ

Thứ tự thực hiện:

1. Nhập URL đăng kí vào Postman, chế độ POST
2. Chuyển tab sang Body, nhập thông tin đăng kí dạng json với email không hợp lệ

3. Ấn send để gửi yêu cầu lên máy chủ

Dữ liệu kiểm thử:

- URL: <http://localhost:3000/v1/auth/register>
- Body:

```
{
  "email": "dagmaail.com",
  "password": "Aa@gmakdi.123",
  "name": "Demo",
  "key": {
    "privateKey": "ApXROwTagKebt2-
1kLJxd9yi7ywUzJV89WY2u5zknB0fbLPcb1Rr2PQpeWogwatBHUetFR-
FFpX0\nMuTiJUG_8qmQTQmECaqG6Jl04_TYqa-p3melfvMzVKl_3BbcNuevFi0POyr32-
880umhLZSeULrU\nd2tmCxnQSjcnkioiww92nv5zyiwQIEV5GWcbxqm0\n",
    "publicKey":
"MFYwEAYHKOZlZj0CAQYFK4EEAAoDQgAER_aGWYwBnT_yzmsrvIgG07wBWVEvEEQjYlmz_HKeL1ZJ\n
bcpI3UDLFG1nJ3GQ8EXsyr0HJ8-AK0IdMxczBVhLQ\n"
  }
}
```

Kết quả mong đợi: Phản hồi gửi về mã lỗi 400 với message là email không hợp lệ

Kết quả thực tế:

- Mã lỗi: 400
- Phản hồi:

```
{
  "statusCode": 400,
  "message": [
    "email must be an email"
  ],
  "error": "Bad Request"
}
```

Pass hoặc Fail: Pass

TC05 – Đăng kí với mật khẩu yếu

Mô tả: Người dùng thực hiện đăng kí, trong đó người dùng nhập vào một mật khẩu yếu

Thứ tự thực hiện:

1. Nhập URL đăng kí vào Postman, chế độ POST
2. Chuyển tab sang Body, nhập thông tin đăng kí dạng json với mật khẩu yếu
3. Ấn send để gửi yêu cầu lên máy chủ

Dữ liệu kiểm thử:

- URL: <http://localhost:3000/v1/auth/register>
- Body:

```
{
  "email": "a@gmail.com",
  "password": "12345678",
  "name": "Demo",
  "key": {
    "privateKey": "ApXROwTagKebt2-
1kLJxd9yi7ywUzJV89WY2u5zknB0fbLPcb1Rr2PQpeWogwatBHUetFR-
FFpX0\nMuTiJUG_8qmQTQmECaqG6Jl04_TYqa-p3melfvMzVKl_3BbcNuevFi0POyr32-
880umhLZSeULrU\nd2tmCxnQSjcnkioiww92nv5zyiwQIEV5GWcbxqm0\n",
    "publicKey":
"MFYwEAYHKOZlZj0CAQYFK4EEAAoDQgAER_aGWYwBnT_yzmsrvIgG07wBWVEvEEQjYlmz_HKeL1ZJ\n
bcpI3UDLFG1nJ3GQ8EXsyr0HJ8-AK0IdMxczBVhLQ\n"
  }
}
```

Kết quả mong đợi: Phản hồi gửi về mã lỗi 400 với message là mật khẩu yếu

Kết quả thực tế:

- Mã lỗi: 400
- Phản hồi:


```
{
  "statusCode": 400,
  "message": [
    "Password at least 8 characters, must contain at least 1 uppercase
    letter, 1 lowercase letter, 1 number and 1 special characters"
  ],
  "error": "Bad Request"
}
```

Pass hoặc Fail: Pass

TC06 – Đăng kí với email hợp lệ và mật khẩu mạnh

Mô tả: Người dùng thực hiện đăng kí, trong đó người dùng nhập vào email hợp lệ và mật khẩu mạnh

Thứ tự thực hiện:

1. Nhập URL đăng kí vào Postman, chế độ POST
2. Chuyển tab sang Body, nhập thông tin đăng kí dạng json
3. Ấn send để gửi yêu cầu lên máy chủ

Dữ liệu kiểm thử:

- URL: <http://localhost:3000/v1/auth/register>
- Body:

```
{
  "email": "a@gmail.com",
  "password": "Demo@hd123456",
  "name": "Demo",
  "key": {
    "privateKey": "ApXROwTagKebt2-
    1kLJxd9yi7ywUzJV89WY2u5zknB0fbLPcb1Rr2PQpeWogwatBHUetFR-
    FFpX0\nMuTiJUG_8qmQTQmECaqG6Jl04_TYqa-p3melfvMzVKl_3BbcNuevFi0POyr32-
    88OumhLZSeULrU\nd2tmCxnQSjcnkioiww92nv5zyiwQIEV5GWcbxqmO\n",
    "publicKey":
    "MFYwEAYHKoZIzj0CAQYFK4EEAAoDQgAER_aGWYwBnT_yzmsrvIgG07wBWVEvEEQjYlmz_HKeL1ZJ\n
    bcpI3UDLFG1NjJ3GQ8EXsyr0HJ8-AK0IdMxczBVhLQ\n"
  }
}
```

Kết quả mong đợi: Phản hồi gửi về mã 201, và thông tin người dùng đăng kí thành công được tạo mới trong bảng **users** trong CSDL

Kết quả thực tế:

- Mã: 201
- CSDL: User được thêm vào CSDL thành công

Pass hoặc Fail: Pass

TC07 – Đăng nhập với email sai

Mô tả: Người dùng thực hiện đăng nhập, trong đó người dùng nhập vào sai email

Thứ tự thực hiện:

1. Nhập URL đăng nhập vào Postman, chế độ POST
2. Chuyển tab sang Body, nhập thông tin đăng nhập dạng json
3. Ấn send để gửi yêu cầu lên máy chủ

Dữ liệu kiểm thử:

- URL: <http://localhost:3000/v1/auth/login>
- Body:

```
{
  "email": "agmail.com",
  "password": "123456"
}
```

Kết quả mong đợi: Phản hồi gửi về mã lỗi 401

Kết quả thực tế:

- Mã lỗi: 401

- **Phản hồi:**

```
{
  "statusCode": 401,
  "message": "Email or password incorrect"
}
```

Pass hoặc Fail: Pass

TC08 – Đăng nhập với mật khẩu sai

Mô tả: Người dùng thực hiện đăng nhập, trong đó người dùng nhập vào sai mật khẩu

Thứ tự thực hiện:

1. Nhập URL đăng nhập vào Postman, chế độ POST
2. Chuyển tab sang Body, nhập thông tin đăng nhập dạng json
3. Ấn send để gửi yêu cầu lên máy chủ

Dữ liệu kiểm thử:

- URL: <http://localhost:3000/v1/auth/login>
- Body:

```
{
  "email": "a@gmail.com",
  "password": "123456"
}
```

Kết quả mong đợi: Phản hồi gửi về mã lỗi 401

Kết quả thực tế:

- Mã lỗi: 401
- Phản hồi:

```
{
  "statusCode": 401,
  "message": "Email or password incorrect"
}
```

Pass hoặc Fail: Pass

TC09 – Đăng nhập với thông tin hợp lệ

Mô tả: Người dùng thực hiện đăng nhập, trong đó người dùng nhập vào đúng email và mật khẩu

Thứ tự thực hiện:

1. Nhập URL đăng nhập vào Postman, chế độ POST
2. Chuyển tab sang Body, nhập thông tin đăng nhập dạng json
3. Ấn send để gửi yêu cầu lên máy chủ

Dữ liệu kiểm thử:

- URL: <http://localhost:3000/v1/auth/login>
- Body:

```
{
  "email": "a@gmail.com",
  "password": "Demo@hd123456"
}
```

Kết quả mong đợi: Phản hồi gửi về mã 200 và thông tin người đăng nhập, cập nhật key vào CSDL nếu key là null

Kết quả thực tế:

- Mã: 200
- Phản hồi:

```
{
  "id": 16,
  "name": "Demo",
  "email": "a@gmail.com",
  "key": {
    "publicKey":
      "MFYwEAYHKoZIzj0CAQYFK4EEAAoDQgAER aGWYwBnT yzmsrvIgG07wBWVEvEEQjYlmz HKeL1ZJ\n"
  }
}
```

Pass hoặc Fail: Pass

Pass hoặc Fail: Pass

<pre> "informations": [] } </pre>
Pass hoặc Fail: Pass
TC14 – Get access token từ refresh token không hợp lệ Thứ tự thực hiện: <ol style="list-style-type: none"> 1. Nhập URL lấy thông tin người dùng hiện tại vào Postman, chế độ POST 2. Chuyển sang tab Authorization, mục Type, chọn Bearer Token, nhập refresh token vào mục Token 3. Ấn send để gửi yêu cầu lên máy chủ Dữ liệu kiểm thử: <ul style="list-style-type: none"> • URL: http://localhost:3000/v1/auth/refresh • Token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MTYsIm1hdCI6MTY1OTQzNTcwMywiZmxwIjoxNjU5Njk0OTAzfQ.Fo0tX8WF1HurlJTPOfG1RoVzsk-79n8QhPPIsdC6m5M Kết quả mong đợi: Phản hồi gửi về mã 401 Kết quả thực tế: <ul style="list-style-type: none"> • Mã: 401 • Phản hồi: <pre> { "statusCode": 401, "message": "Unauthorized" } </pre>
Pass hoặc Fail: Pass
TC15 – Get access token từ refresh token hợp lệ Thứ tự thực hiện: <ol style="list-style-type: none"> 1. Nhập URL lấy thông tin người dùng hiện tại vào Postman, chế độ POST 2. Chuyển sang tab Authorization, mục Type, chọn Bearer Token, nhập refresh token vào mục Token 3. Ấn send để gửi yêu cầu lên máy chủ Dữ liệu kiểm thử: <ul style="list-style-type: none"> • URL: http://localhost:3000/v1/auth/refresh • Token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MTYsIm1hdCI6MTY1OTQzNTcwMywiZmxwIjoxNjYwMDQwNTAzfQ.gqH7lk4ZZRm2vg-AF-KqxjKfY7vKfjxWVvKHKKTi87J8 Kết quả mong đợi: Phản hồi gửi về mã 200 và thông tin người dùng với access token và refresh token mới Kết quả thực tế: <ul style="list-style-type: none"> • Mã: 200 • Phản hồi: <pre> { "id": 16, "name": "Demo", "email": "a@gmail.com", "key": { "publicKey": "MFYwEAYHKoZIzj0CAQYFK4EEAAoDQgAER_aGWYwBnT_yzmsrvIgG07wBWVEvEEQjYlmz_HKeL1ZJ\n bcpI3UDLFG1NjJ3GQ8EXsyr0HJ8-AK0IdMxczBVhLQ\n", "privateKey": "ApXROwTagKebt2- 1kLJxd9yi7ywUzJV89WY2u5zknB0fbLPcb1Rr2PQpeWogwatBHUetFR- FFpX0\nMuTiJUG_8qmQTQmECaqG6Jl04_TYqa-p3melfvMzVKl_3BbcNuevFi0POyr32- 88OumhLZSeULrU\nd2tmCxnQSjcnkioiww92nv5zyiwQIEV5GWcbxqmO\n" }, } </pre>

<pre> "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MTYsImhhdCI6MTY1OTQzNjM0OSwiZXhwIjoxNjYwMDQxMTQ5fQ.EG1Q5LFq6eVG9iqbtdXj1sdKqQSq5mBxFUc0eV2Zv3U", "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MTYsImhhdCI6MTY1OTQzNjM0OSwiZXhwIjoxNjU5Njk1NTQ5fQ.7hQuJAH1IUdaavNldBrBjiNjMKCsTdi-lsKNLQyhjE" } </pre>
Pass hoặc Fail: Pass
TC16 – Kết nối với socket bằng token không hợp lệ Thứ tự thực hiện: <ol style="list-style-type: none"> 1. Nhập URL của ứng dụng vào Postman, chế độ SocketIO 2. Chuyển sang tab Params, thêm Query Params với Key là token, Value là access token của người dùng 3. Ấn Connect để kết nối với socket Dữ liệu kiểm thử: <ul style="list-style-type: none"> • URL: http://localhost:3000/ • Token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MTYsImhhdCI6MTY1OTQzNTcwMywiZXhwIjoxNjU5Njk0OTAzfQ.Fo0tX8WF1HurlJTPOfG1RoVzsk-79n8QhPPIsdC6m5M Kết quả mong đợi: Không connect được với Socket Kết quả thực tế: Không connect được với Socket Pass hoặc Fail: Pass
TC17 – Kết nối với socket bằng token hợp lệ Thứ tự thực hiện: <ol style="list-style-type: none"> 1. Nhập URL của ứng dụng vào Postman, chế độ SocketIO 2. Chuyển sang tab Params, thêm Query Params với Key là token, Value là access token của người dùng 3. Ấn Connect để kết nối với socket Dữ liệu kiểm thử: <ul style="list-style-type: none"> • URL: http://localhost:3000/ • Token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MTYsImhhdCI6MTY1OTQ0NjI3MiwiZXhwIjoxNjU5NzA1NDcyfQ.rjZkwWccxmiVbKpBqcIODC5_REMY4zGsXAi6sGRiTsU Kết quả mong đợi: Connect được với Socket, và thông tin SocketID của người dùng kết nối tới được lưu vào bảng <i>informations</i> Kết quả thực tế: Connect được với Socket và trong CSDL có thông tin SocketID kết nối tới Pass hoặc Fail: Pass
TC18 – Ngắt kết nối với Socket Điều kiện tiên quyết: Người dùng đã kết nối tới socket Thứ tự thực hiện: <ol style="list-style-type: none"> 1. Ấn nút Disconnect Kết quả mong đợi: Ngắt kết nối với Socke thành công, và xóa thông tin SocketID tương ứng trong bảng <i>informations</i> Kết quả thực tế: Ngắt kết nối thành công và thông tin SocketID tương ứng bị xóa Pass hoặc Fail: Pass
TC19 – Gửi tin nhắn thông qua socket Điều kiện tiên quyết: Những người dùng nhắn tin với nhau đều kết nối thành công

tới socket

Thứ tự thực hiện:

1. Tại tab Events, thêm sự kiện `receive_message`
2. Tại mục New message, thêm sự kiện `send_message`, và đổi loại dữ liệu sang JSON
3. Nhập tin nhắn theo đúng cú pháp và ấn Send

Dữ liệu kiểm thử:

- Message tuân theo dạng dưới

```
{
  "to": 17,
  "content": "Hi"
}
```

Kết quả mong đợi: Người nhận nhận được tin nhắn và người gửi nhận được tin ack trả về, bảng *messages* cập nhật tin nhắn vừa được gửi và bảng *conversations* cập nhật tin nhắn mới nhất ứng với 2 userid gửi và nhận

Kết quả thực tế: Người nhận nhận được tin nhắn và người gửi nhận được tin ack trả về, bảng *messages* cập nhật tin nhắn vừa được gửi và bảng *conversations* cập nhật tin nhắn mới nhất ứng với 2 userid gửi và nhận

Pass hoặc Fail: Pass

TC20 – Lấy danh sách những hội thoại của người dùng đăng nhập

Điều kiện tiên quyết: Token hợp lệ

Thứ tự thực hiện:

1. Nhập URL lấy đoạn hội thoại vào Postman, chế độ GET
2. Chuyển sang tab Authorization, mục Type, chọn Bearer Token, nhập refresh token vào mục Token
3. Ấn send để gửi yêu cầu lên máy chủ

Dữ liệu kiểm thử:

- URL: <http://localhost:3000/v1/conversation>
- Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MTYsIm1hdCI6MTY1OTQ0NjI3MiwiZmxhZSI6bnV5ZnZlNDcyfQ.rjZkwWccxmiVbKpBqcIODC5_REMY4zGsXAI6sGRiTsu

Kết quả mong đợi: Phản hồi gửi về mã 200 và thông tin những đoạn hội thoại của người dùng hiện tại

Kết quả thực tế:

- Mã: 200
- Phản hồi:

```
[
  {
    "id": 1,
    "userId": 16,
    "toUserId": 17,
    "content": "Hello",
    "type": "text",
    "timestamp": "2022-08-02T14:16:22.771Z"
  }
]
```

Pass hoặc Fail: Pass

TC21 – Lấy danh sách những danh sách tin nhắn với người dùng khác

Điều kiện tiên quyết: Token hợp lệ

Thứ tự thực hiện:

1. Nhập URL lấy danh sách tin nhắn vào Postman, chế độ GET

2. Chuyển sang tab Authorization, mục Type, chọn Bearer Token, nhập refresh token vào mục Token
3. Ấn send để gửi yêu cầu lên máy chủ

Dữ liệu kiểm thử:

- URL: `http://localhost:3000/v1/message/17`
- Token:
`eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MTYsIm1hdCI6MTY1OTQ0NjI3MiwiZmxhZSI6bnV5NDcyfQ.rjZkwWccxmiVbKpBqcIODC5_REMY4zGsXAI6sGRiTsU`

Kết quả mong đợi: Phản hồi gửi về mã 200 và danh sách tin nhắn với user có id 17

Kết quả thực tế:

- Mã: 200
- Phản hồi:

```
[
  {
    "id": 11,
    "from": 16,
    "to": 17,
    "content": "Hi",
    "type": "text",
    "timestamp": "2022-08-02T14:15:58.340Z"
  },
  {
    "id": 12,
    "from": 17,
    "to": 16,
    "content": "Hello",
    "type": "text",
    "timestamp": "2022-08-02T14:16:22.771Z"
  }
]
```

Pass hoặc Fail: Pass

4.2.2 Kiểm thử khi hệ thống triển khai trên Heroku Server

Các test case như khi kiểm thử trên local, nhưng URL của máy chủ, thay thế bằng URL của máy chủ trên Heroku. Các kết quả của test case khi kiểm thử trên Heroku Server đều pass

4.3 Kiểm thử các chức năng trên ứng dụng Android

4.3.1 Kiểm thử chức năng trao đổi khoá

Để thực hiện kiểm thử chức năng trao đổi khoá, tôi đã tách riêng module trao đổi khoá ra thành một project riêng. Kết quả **Error! Reference source not found.** là kết quả của 2 lần kiểm thử trong số nhiều lần kiểm tra.

Bảng 4.4 Bảng kiểm thử chức năng trao đổi khoá

Lần test	Alice	Bob
1	PublicKey: 30 56 30 10 06 07 2a 86 48 ce 3d 02 01 06 05 2b 81 04 00 0a 03 42 00 04 46 50 53 8c b7 c5 cb 14 76 06 07 49 86 3d 37 4d 35 7d 82 96	PublicKey: 30 56 30 10 06 07 2a 86 48 ce 3d 02 01 06 05 2b 81 04 00 0a 03 42 00 04 27 f8 ed 22 65 dd d3 e1 90 9c 67 aa e8 71 5d 7b 8a e4 6b a5

	<p>4c 36 ab e4 89 73 1b 70 3d 17 a4 cc 25 03 b4 dc da d0 af 59 a2 9e af 96 02 d8 2a d8 17 36 35 69 92 52 c8 c6 1f 25 e3 13 28 5c 6b 3e</p> <p>Private Key: 30 81 8d 02 01 00 30 10 06 07 2a 86 48 ce 3d 02 01 06 05 2b 81 04 00 0a 04 76 30 74 02 01 01 04 20 0f 26 95 92 ab cd bc 32 6b 45 02 93 c4 df 5e 2a 78 73 69 ce 7d 5e 5f e8 fb 30 00 aa 57 43 a5 97 a0 07 06 05 2b 81 04 00 0a a1 44 03 42 00 04 46 50 53 8c b7 c5 cb 14 76 06 07 49 86 3d 37 4d 35 7d 82 96 4c 36 ab e4 89 73 1b 70 3d 17 a4 cc 25 03 b4 dc da d0 af 59 a2 9e af 96 02 d8 2a d8 17 36 35 69 92 52 c8 c6 1f 25 e3 13 28 5c 6b 3e</p> <p>Shared Key: 9b 3d ba 22 c4 d6 ef db 3c 4e 96 ac e0 69 2d 58 32 7d 24 98 f9 1b 2e 48 33 fc 09 82 7e 13 ad 66</p> <p>Final Key: 6f 37 ea a8 58 05 f7 e6 c1 ce 60 96 3f 83 de f9 1f fa ea f9 34 0b 30 7e a7 65 bd 46 65 8e 38 ec</p>	<p>d8 a5 e6 65 d4 60 83 25 81 18 98 5c 14 5b e2 b2 7a 31 81 1b aa 74 51 ac 03 5b 82 82 d7 9c a9 33 fa 70 3e 1d 45 6b c7 c4 e0 31 ee 06</p> <p>Private Key: 30 81 8d 02 01 00 30 10 06 07 2a 86 48 ce 3d 02 01 06 05 2b 81 04 00 0a 04 76 30 74 02 01 01 04 20 95 40 23 bd 9b 34 d9 5a 9f 2d 94 e3 5c 23 fb 9b c0 bd e1 61 23 9b 57 4b d9 c5 b8 b7 86 25 e9 9f a0 07 06 05 2b 81 04 00 0a a1 44 03 42 00 04 27 f8 ed 22 65 dd d3 e1 90 9c 67 aa e8 71 5d 7b 8a e4 6b a5 d8 a5 e6 65 d4 60 83 25 81 18 98 5c 14 5b e2 b2 7a 31 81 1b aa 74 51 ac 03 5b 82 82 d7 9c a9 33 fa 70 3e 1d 45 6b c7 c4 e0 31 ee 06</p> <p>Shared Key: 9b 3d ba 22 c4 d6 ef db 3c 4e 96 ac e0 69 2d 58 32 7d 24 98 f9 1b 2e 48 33 fc 09 82 7e 13 ad 66</p> <p>Final Key: 6f 37 ea a8 58 05 f7 e6 c1 ce 60 96 3f 83 de f9 1f fa ea f9 34 0b 30 7e a7 65 bd 46 65 8e 38 ec</p>
2	<p>PublicKey: 30 56 30 10 06 07 2a 86 48 ce 3d 02 01 06 05 2b 81 04 00 0a 03 42 00 04 69 0b f0 08 06 10 7d a7 e7 ba 54 c8 a2 0a 70 68 b8 e5 4a 6b 61 86 5a b6 d5 fa a5 cb fc 78 d1 d4 19 63 29 ef 21 d5 63 8f d3 14 de e8 09 9d 2d 0d 0b 05 f1 73 fe 4f 04 6e 2c 20 3e bd 6e 9a ca 1e</p> <p>Private Key: 30 81 8d 02 01 00 30 10 06 07 2a 86 48 ce 3d 02 01 06 05 2b 81 04 00 0a 04 76 30 74 02 01 01 04 20 76 00 4e be 7a ad 44 05 ab 7e a0 34 e7 e7 78 17 fc f6 e0 59 a6 45 87 10 33 b7 75 28 f3 d9 da 22 a0 07 06 05 2b 81 04 00 0a a1 44 03 42 00 04 69 0b f0 08 06 10 7d a7 e7 ba 54 c8 a2 0a 70 68 b8 e5 4a 6b 61 86 5a b6 d5 fa a5 cb fc 78 d1 d4 19 63 29 ef 21 d5 63 8f d3 14 de e8 09 9d 2d 0d 0b 05 f1 73 fe 4f 04 6e 2c 20 3e bd 6e 9a ca 1e</p> <p>Shared Key: 99 5f be 27 ef f0 bf ea 79 f5 3b dc 4a 5f 1d bf 28 62 13 8a c2 83 1e 32 e8 c9 4a 01 1d 10 d2 c0</p> <p>Final Key: 2d 9e b0 90 ec c8 75 19 58 8c 08 3b 3f 78 6d 94 17 75 43 2f b9 81 84 79 c7 10 ef 75 9e 98 0a 04</p>	<p>PublicKey: 30 56 30 10 06 07 2a 86 48 ce 3d 02 01 06 05 2b 81 04 00 0a 03 42 00 04 35 24 4a c9 b0 d0 55 82 7f 7e 00 2a 09 49 c3 2d 7f 6d 92 c2 0d fb 0e e9 c3 24 d8 dd 50 77 a2 3f a1 8e 9e 2a cc 3f 53 93 3d 9d d0 e9 ea 6c f7 bc 04 0a f7 a3 92 b6 42 44 ba e5 26 39 a2 ed 79 44</p> <p>Private Key: 30 81 8d 02 01 00 30 10 06 07 2a 86 48 ce 3d 02 01 06 05 2b 81 04 00 0a 04 76 30 74 02 01 01 04 20 c3 01 0e 22 01 20 c0 51 9a f8 95 6f bc 4c 62 e7 93 30 f2 93 14 7c c3 e2 d1 aa 22 79 c9 f6 a6 23 a0 07 06 05 2b 81 04 00 0a a1 44 03 42 00 04 35 24 4a c9 b0 d0 55 82 7f 7e 00 2a 09 49 c3 2d 7f 6d 92 c2 0d fb 0e e9 c3 24 d8 dd 50 77 a2 3f a1 8e 9e 2a cc 3f 53 93 3d 9d d0 e9 ea 6c f7 bc 04 0a f7 a3 92 b6 42 44 ba e5 26 39 a2 ed 79 44</p> <p>Shared Key: 99 5f be 27 ef f0 bf ea 79 f5 3b dc 4a 5f 1d bf 28 62 13 8a c2 83 1e 32 e8 c9 4a 01 1d 10 d2 c0</p> <p>Final Key: 2d 9e b0 90 ec c8 75 19 58 8c 08 3b 3f 78 6d 94 17 75 43 2f b9 81 84 79 c7 10 ef 75 9e 98 0a 04</p>

4.3.2 Kiểm thử các chức năng liên quan đến đăng nhập, đăng ký

Bảng 4.5 là các test case để kiểm thử chức năng đăng nhập, đăng kí

Bảng 4.5 Bảng kiểm thử chức năng đăng nhập, đăng kí

ID	Mô tả test case	Thứ tự thực hiện	Test data	Kết quả mong đợi	Kết quả thực tế	Pass hoặc Fail
TC22	Đăng ký mà không nhập đầy đủ form	1. Không nhập đầy đủ các mục trong Name, Email, Password, Cornfirm Password 2. Ấn Sign In để gửi đăng kí		Hiện thông báo nhập đầy đủ các mục	Hiện thông báo nhập đầy đủ các mục	Pass
TC23	Đăng ký mà không	1. Nhập Name 2. Nhập Email không hợp lệ	Name: Demo Email: demo Password:	Thông báo email không hợp lệ	Thông báo email không hợp lệ	Pass

	nhập email không hợp lệ	3. Nhập Password 4. Nhập Confirm Password 5. Ấn Sign In để gửi đăng kí	Demo_hd2k Confirm Password: Demo_hd2k			
TC24	Đăng ký mà không nhập password và confirm password không giống nhau	1. Nhập Name 2. Nhập Email 3. Nhập Password 4. Nhập Confirm Password không giống Password 5. Ấn Sign In để gửi đăng kí	Name: Demo Email: demo@gmail.com Password: Demo_hd2k Confirm Password: Demo_hd2	Thông báo confirm password không đúng	Thông báo confirm password không đúng	Pass
TC25	Đăng ký mà không nhập password yếu giống nhau	1. Nhập Name 2. Nhập Email 3. Nhập Password yếu 4. Nhập Confirm Password giống Password 5. Ấn Sign In để gửi đăng kí	Name: Demo Email: demo@gmail.com Password: Demo12345 Confirm Password: Demo12345	Thông báo mật khẩu yếu	Thông báo mật khẩu yếu	Pass
TC26	Đăng ký với các thông tin hợp lệ	1. Nhập Name 2. Nhập Email hợp lệ 3. Nhập Password 4. Nhập Confirm Password giống Password 5. Ấn Sign In để gửi đăng kí	Name: Demo Email: demo@gmail.com Password: Demo_hd2k Confirm Password: Demo_hd2k	- Thông báo đăng kí thành công nếu trên máy chủ chưa có gmail này - Thông báo đăng kí không thành công nếu tồn tại người dùng trên máy chủ hoặc không có mạng	- Thông báo đăng kí thành công nếu trên máy chủ chưa có gmail này - Thông báo đăng kí không thành công nếu tồn tại người dùng trên máy chủ hoặc không có mạng	Pass
TC27	Đăng nhập với người dùng không có trên hệ thống hoặc mật khẩu sai	1. Nhập Email 2. Nhập Password 3. Ấn Login để gửi thông tin đăng nhập	TH1: Email: demo@gmail.com Password: Demo_hd2 TH2: Email: demo2@gmail.com Password: Demo2	Thông báo email hoặc mật khẩu sai	Thông báo email hoặc mật khẩu sai	Pass
TC28	Đăng nhập với các thông tin hợp lệ	1. Nhập Email 2. Nhập Password 3. Ấn Login để gửi thông tin đăng nhập	Email: demo@gmail.com Password: Demo_hd2k	- Thông báo đăng nhập thành công và chuyển màn hình sang màn hình home	- Thông báo đăng nhập thành công và chuyển màn hình sang màn hình home	Pass

4.3.3 Kiểm thử chức năng nhấn tin

Các test case để kiểm thử chức năng nhấn tin như

Bảng 4.6

Bảng 4.6 Kiểm thử chức năng nhắn tin

ID	Mô tả test case	Thứ tự thực hiện	Test data	Kết quả mong đợi	Kết quả thực tế	Pass hoặc Fail
TC29	A thực hiện nhắn tin cho B	1. A nhập nội dung tin nhắn dạng text 2. Ấn Gửi để gửi tin nhắn	Nội dung tin nhắn: Demo Key: Secret	B nhận được tin nhắn như A gửi và trên máy chủ lưu tin nhắn đã được mật mã hoá	B nhận được tin nhắn như A gửi và trên máy chủ lưu tin nhắn đã được mật mã hoá	Pass

4.4 Đánh giá

Từ các kết quả kiểm thử được nêu ở những mục trên, ứng dụng đã đạt được những mục tiêu ban đầu đề ra trong đồ án.

4.5 Kết luận chương

Trong Chương 4, đồ án đã thực hiện kiểm thử và đánh giá hệ mật và ứng dụng có sử dụng mã hoá đầu cuối. Ở chương này, đồ án đã kiểm tra tất cả những trường hợp có thể xảy ra khi người dùng sử dụng. Sau quá trình kiểm thử, hệ mật và ứng dụng đều đạt được những yêu cầu đã đề ra khi làm đồ án.

KẾT LUẬN

Kết luận chung

Dưới sự hướng dẫn của thầy Đỗ Trọng Tuấn và anh Vũ Văn Mạnh, em đã hoàn thành được đồ án tốt nghiệp. Em đã có thêm hiểu biết về các kiến thức mật mã hoá, đặc biệt là mật mã hoá hỗn loạn, một trong những vấn đề liên quan đến bảo mật được nghiên cứu nhiều năm gần đây. Ngoài ra, em đã có thể thiết kế được một hệ mật hỗn loạn, ứng dụng hệ mật trong mã hoá đầu cuối và triển khai và một ứng dụng sử dụng mã hoá đầu cuối. Sau quá trình tìm hiểu lý thuyết hỗn loạn, thiết kế hệ mật hỗn loạn, hệ mật hỗn loạn sau thiết kế có thể mật mã hoá và giải mật được ra đúng dữ liệu ban đầu khi khoá đúng, thời gian mật mã và giải mật nhanh. Ứng dụng mã hoá đầu cuối có sử dụng hệ mật hỗn loạn được thiết kế hoạt động đúng chức năng, tin nhắn gửi đi được mật mã hoá và tại nơi nhận được giải mật thành công.

Hướng phát triển

Tuy đạt được những mục tiêu ban đầu đề ra, nhưng do hạn chế về mặt thời gian, đồ án vẫn chưa triển khai được hết những ứng dụng của mã hoá đầu cuối, ví dụ như trong nhắn tin với một nhóm nhiều người, gọi điện, video call... Và thuật toán mật mã hoá hỗn loạn code chưa được tối ưu, chưa đánh giá được hết hiệu quả của thuật toán mật mã hoá. Trong tương lai, nếu có cơ hội, em sẽ nghiên cứu, phát triển tiếp mật mã hoá hỗn loạn trong mã hoá đầu cuối, nghiên cứu những thuật toán khác tối ưu hơn để triển khai vào hệ thống.

TÀI LIỆU THAM KHẢO

- [1] R. L. Rivest, "Cryptography," in *Handbook of Theoretical Computer Science*, vol. 1, 1990.
- [2] C. E. Shannon, "Communication theory of secrecy systems," in *The Bell System Technical Journal*, vol. 28, 1949, pp. 656-715.
- [3] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd ed., 1995.
- [4] T. T. Tung, *Giáo trình Mật mã học và An toàn thông tin*, 2003.
- [5] R. Gilmore and M. Lefranc, *The Topology of Chaos: Alice in Stretch and Squeezeland*, 2nd ed., 2002.
- [6] M. Curtin, *Brute Force: Cracking the Data Encryption Standard*, 2005th ed., 2005.
- [7] S. Halevi and H. Krawczyk, "Strengthening Digital Signatures Via Randomized Hashing," *Advances in Cryptology - CRYPTO 2006*, pp. 41-49, 2006.
- [8] "IBM," [Online]. Available: <https://www.ibm.com/>. [Accessed 25 July 2022].
- [9] S. Nakov, *Practical Cryptography for Developer*, 2018.
- [10] N. Sullivan, "A (Relatively Easy To Understand) Primer on Elliptic Curve Cryptography," CloudFlare, [Online]. Available: <https://blog.cloudflare.com/>. [Accessed 4 July 2022].
- [11] D. Hankerson, S. Vanstone and A. Menezes, *Guide to Elliptic Curve Cryptography*, 2004.
- [12] "Mvp Workshop," [Online]. Available: <https://mvpworkshop.co/>. [Accessed 25 July 2022].
- [13] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. S. Stein, "Section 31.9: Integer factorization," in *Introduction to Algorithms*, 3rd ed., 2009.
- [14] "Andrea Corbellini," [Online]. Available: <https://andrea.corbellini.name/>. [Accessed 25 July 2022].
- [15] L. Kocarev and S. Lian, *Chaos-based Cryptography*, 2011.
- [16] L. Kocarev and S. Lian, *Chaos-Based Cryptography Theory, Algorithms and Applications*, 2011.
- [17] R. Gilmore and M. Lefranc, *The topology of chaos: Alice in stretch*, Chichester: Wiley, 2002.

- [18] K. A. K. Patro and B. Acharya, "An efficient colour image encryption," *Journal of Information Security and Application*, vol. 46, pp. 23-41, 2019.
- [19] Alvarez, "Some basic cryptographic requirements for chaos based cryptosystems," *International Journal of Bifurcation and Chaos* 16, pp. 2129-2151, 2006.
- [20] Q. Lu, C. Zhu and G. Wang, "Novel S-Box Design Algorithm Based on a New Compound Chaotic System.," 2019.
- [21] "Viblo," [Online]. Available: <https://viblo.asia/>. [Accessed 25 July 2022].
- [22] "Tutorialspoint," [Online]. Available: <https://www.tutorialspoint.com>. [Accessed 4 August 2022].
- [23] "Nodejs," [Online]. Available: <https://nodejs.org>. [Accessed 25 July 2022].
- [24] "Nest JS," [Online]. Available: <https://docs.nestjs.com/>. [Accessed 25 July 2022].
- [25] "RedHat," [Online]. Available: <https://www.redhat.com/>. [Accessed 25 July 2022].
- [26] "FPT Cloud," [Online]. Available: <https://fptcloud.com/>. [Accessed 25 July 2022].
- [27] "Bizfly Cloud," [Online]. Available: <https://bizflycloud.vn/>. [Accessed 2 April 2022].
- [28] "Tino Group," [Online]. Available: <https://wiki.tino.org/>. [Accessed 25 July 2022].
- [29] "Android Source," [Online]. Available: <https://source.android.com/>. [Accessed 25 July 2022].
- [30] "Developer Android," [Online]. Available: <https://developer.android.com/>. [Accessed 5 August 2022].

PHỤ LỤC

Phụ lục 1. Các phần mềm, framework sử dụng trong đồ án

1.1 *Phía máy chủ*

Có nhiều framework để lập trình phía server: Java Spring, Ktor, Ruby,... Trong đồ án này, tôi sử dụng framework Nestjs trên nền tảng Nodejs – một nền tảng được xây dựng trên “V8 Javascript engine”.

1.1.1 *Giới thiệu về Nodejs*

Nodejs là một nền tảng (Platform) phát triển độc lập được xây dựng ở trên Javascript Runtime của Chrome mà chúng ta có thể xây dựng được các ứng dụng mạng một cách nhanh chóng và dễ dàng mở rộng. Phần Core bên dưới của Nodejs được viết hầu hết bằng C++ nên cho tốc độ xử lý và hiệu năng khá cao. Nodejs tạo ra được các ứng dụng có tốc độ xử lý nhanh, thời gian thực. Nodejs áp dụng cho các sản phẩm có lượng truy cập lớn, cần mở rộng nhanh, cần đổi mới công nghệ, hoặc tạo ra các dự án khởi nghiệp nhanh nhất có thể [23].

1.1.2 *Giới thiệu về Nestjs*

Nest (NestJS) là một framework để xây dựng các ứng dụng phía máy chủ Node.js hiệu quả, có thể mở rộng. Nestjs sử dụng JavaScript, được xây dựng và hỗ trợ đầy đủ TypeScript (nhưng vẫn cho phép các nhà phát triển viết mã bằng JavaScript thuần túy) và kết hợp các yếu tố của OOP (Lập trình hướng đối tượng), FP (Lập trình chức năng) và FRP (Lập trình phản ứng chức năng).

Nest sử dụng các khung Máy chủ HTTP mạnh mẽ như Express và tùy chọn có thể được định cấu hình để sử dụng cả Fastify. Nest cung cấp mức độ trừu tượng cao hơn các khung Node.js phổ biến này (Express / Fastify), nhưng cũng hiển thị trực tiếp các API của họ cho nhà phát triển. Điều này cho phép các nhà phát triển tự do sử dụng vô số mô-đun của bên thứ ba có sẵn cho nền tảng cơ bản. [24]

1.2 *Giới thiệu về REST API*

REST API (còn được gọi là RESTful API) là một giao diện lập trình ứng dụng (API hoặc web API) tuân theo các ràng buộc của nguyên tắc thiết kế REST và cho phép tương

tác với các dịch vụ web RESTful. REST là viết tắt của chuyển trạng thái đại diện (representational state transfer) và được tạo ra bởi nhà khoa học máy tính Roy Fielding [25].

Các REST API giao tiếp thông qua các yêu cầu HTTP để thực hiện các chức năng cơ sở dữ liệu tiêu chuẩn như tạo, đọc, cập nhật và xóa các bản ghi (còn được gọi là CRUD) trong một tài nguyên. Ví dụ: REST API sẽ sử dụng yêu cầu GET để truy xuất một bản ghi, một yêu cầu POST để tạo một bản ghi, một yêu cầu PUT để cập nhật một bản ghi và một yêu cầu DELETE để xóa một bản ghi. Tất cả các phương thức HTTP đều có thể được sử dụng trong các lệnh gọi API. API REST được thiết kế tốt tương tự như một trang web chạy trong trình duyệt web với chức năng HTTP được tích hợp sẵn. Trạng thái của tài nguyên tại bất kỳ thời điểm cụ thể hoặc dấu thời gian nào, được gọi là biểu diễn tài nguyên. Thông tin này có thể được gửi tới máy khách ở hầu hết mọi định dạng bao gồm JSON, HTML, XLT, Python, PHP hoặc văn bản thuần túy. JSON phổ biến vì nó có thể đọc được bởi cả con người và máy móc. Tiêu đề yêu cầu và tham số cũng rất quan trọng trong lệnh gọi API REST vì chúng bao gồm thông tin định danh quan trọng như siêu dữ liệu, ủy quyền, mã định danh tài nguyên thống nhất (URI), bộ nhớ đệm, cookie và hơn thế nữa. Header của yêu cầu và header của phản hồi, cùng với mã trạng thái HTTP thông thường (HTTP status code), được sử dụng trong các REST API được thiết kế tốt [8].

1.3 Giới thiệu về Socket IO

Socket IO đã ra đời tạo thuận lợi cho quá trình giao tiếp giữa server và client. Công cụ này chính thức được phát hành từ năm 2010. Đây có thể xem như công cụ hỗ trợ tạo môi trường giao tiếp thuận lợi hơn trong hệ thống mạng internet. Từ đó trả về kết quả hữu ích ngay tại thời điểm các bên đang giao tiếp với nhau. Quá trình tương tác giữa server và client được duy trì bởi Socket IO với điều kiện client cần module tại trình duyệt. Đồng thời, server phải tích hợp sẵn dịch vụ Socket IO. Phần lớn ứng dụng tích hợp Socket IO đều yêu cầu tốc độ phản hồi ngay. Chẳng hạn như ứng dụng xem trực tiếp kết quả xổ số, ứng dụng chat,.. [26]

1.4 Giới thiệu về Postges SQL

1.4.1 Tổng quan về SQL

SQL là viết tắt của Structured Query Language, là ngôn ngữ truy vấn có cấu trúc,

cho phép bạn truy cập và thao tác với các cơ sở dữ liệu để tạo, xóa, sửa đổi, trích xuất dữ liệu. Các hệ thống quản trị cơ sở dữ liệu như MySQL, MS Access, Oracle, Sybase, Informix, Postgres hay SQL Server đều lấy SQL làm ngôn ngữ cơ sở dữ liệu tiêu chuẩn [21].

Trong đề tài này, tôi sử dụng Postgres SQL để làm nơi lưu trữ các dữ liệu của người dùng

1.4.2 Tổng quan về Postgres SQL

PostgreSQL là một hệ thống quản trị cơ sở dữ liệu quan hệ-đối tượng (object-relational database management system) có mục đích chung, hệ thống cơ sở dữ liệu mã nguồn mở tiên tiến nhất hiện nay. PostgreSQL là một phần mềm mã nguồn mở miễn phí. Mã nguồn của phần mềm khả dụng theo license của PostgreSQL, một license nguồn mở tự do. Theo đó, bạn sẽ được tự do sử dụng, sửa đổi và phân phối PostgreSQL dưới mọi hình thức. PostgreSQL không yêu cầu quá nhiều công tác bảo trì bởi có tính ổn định cao. Ngoài ra, PostgreSQL có hiệu năng cao và có nhiều tính năng tùy chỉnh thêm vào. Do đó, khi phát triển các ứng dụng dựa trên PostgreSQL, chi phí sở hữu sẽ thấp hơn so với các hệ thống quản trị dữ liệu khác [27].

1.5 Giới thiệu về Heroku Server

Heroku là một nền tảng đám mây cho phép các cá nhân, doanh nghiệp xây dựng, triển khai, quản lý và mở rộng ứng dụng. Lợi thế chính của nền tảng này nằm ở tính linh hoạt và dễ sử dụng, đây là giải pháp giúp các nhà phát triển đưa ứng dụng của họ ra thị trường đơn giản và nhanh chóng nhất. Được quản lý bởi một công ty cùng tên, Heroku mang lại một trải nghiệm tuyệt vời cho nhà phát triển ứng dụng, khi họ chỉ cần tập trung vào phát triển, nâng cấp sản phẩm cốt lõi của mình mà không bị phân tâm trong việc duy trì hoạt động máy chủ, phần cứng hoặc các cơ sở hạ tầng. Heroku hỗ trợ nhiều ngôn ngữ lập trình: NodeJS, Ruby, Python, PHP, Java, Scala, Clojure, Go, Kotlin. Người dùng có thể sử dụng dịch vụ hoàn toàn miễn phí với các ứng dụng web không yêu cầu phải có tốc độ truy cập cao hay dung lượng lớn [28].

Heroku Server cho phép trở domain về, hỗ trợ up code dùng Github với tốc độ rất nhanh. Thêm nữa, Heroku rất ổn định và chạy tốt với ứng dụng nodejs. Bên cạnh những điểm lợi, Heroku có những hạn chế nhất định: lưu trữ file trực tiếp trên ứng dụng sẽ bị mất, database hỗ trợ dạng plugin nên sử dụng phức tạp và đặc biệt HTTP timeout.

1.6 Phía máy khách

Hiện nay, phía máy khách có rất nhiều nền tảng. Ví dụ: Web, Windows, Linux, iOS,... Nền tảng Android là nền tảng mà đồ án sử dụng để triển khai thử nghiệm sản phẩm.

1.6.1 Tổng quan về hệ điều hành Android

Android là hệ điều hành mã nguồn mở, dựa trên Linux Kernel, dành cho các thiết bị di động nói chung (điện thoại, máy tính bảng, đồng hồ thông minh, máy nghe nhạc,...). Có nghĩa là Android không chỉ giới hạn trong phạm vi một hệ điều hành cho điện thoại! Nó có thể được nhà sản xuất cài đặt lên đồng hồ, máy nghe nhạc, thiết bị định vị GPS, thậm chí là ô tô (các thiết bị Android Auto). Android cũng không phải là một thiết bị hay sản phẩm cụ thể, nó là một hệ điều hành dựa trên Linux, nguồn mở, linh hoạt. Hiện Android là một thương hiệu của Google. Có khả năng tùy biến rất cao và có thể chạy trên nhiều thiết bị, nhiều kiến trúc vi xử lý (ARM / x86) [21].

1.4.2 Kiến trúc hệ điều hành Android

Kiến trúc hệ điều hành Android như sơ đồ Hình 4.1 [29]. Trong đó:

- **Application framework:** Khung ứng dụng được các nhà phát triển ứng dụng sử dụng thường xuyên nhất. Là một nhà phát triển phần cứng, bạn nên biết các API của nhà phát triển vì nhiều API ánh xạ trực tiếp đến các giao diện HAL bên dưới và có thể cung cấp thông tin hữu ích về việc triển khai các trình điều khiển.
- **Binder IPC:** Cơ chế Binder Inter-Process Communication (IPC) cho phép khung ứng dụng vượt qua ranh giới tiến trình và gọi vào mã dịch vụ hệ thống Android. Điều này cho phép các khung API cấp cao tương tác với các dịch vụ hệ thống Android. Ở cấp khung ứng dụng (application framework), giao tiếp này bị ẩn khỏi nhà phát triển và mọi thứ dường như "chỉ hoạt động" ("just work").
- **System services:** Dịch vụ hệ thống là các thành phần tập trung, mô-đun như Trình quản lý cửa sổ, Dịch vụ tìm kiếm hoặc Trình quản lý thông báo. Chức năng được thể hiện bởi các khung ứng dụng API giao tiếp với các dịch vụ hệ thống để truy cập vào phần cứng bên dưới. Android bao gồm hai nhóm dịch vụ: hệ thống (chẳng hạn như Trình quản lý cửa sổ và Trình quản lý thông báo)

và phương tiện (các dịch vụ liên quan đến phát và ghi phương tiện).



Hình 4.1 Kiến trúc hệ điều hành Android

- **Lớp trừu tượng phần cứng (HAL):** HAL xác định một giao diện tiêu chuẩn để các nhà cung cấp phần cứng triển khai, điều này cho phép Android không có khả năng triển khai trình điều khiển cấp thấp hơn. Sử dụng HAL cho phép bạn triển khai chức năng mà không ảnh hưởng hoặc sửa đổi hệ thống cấp cao hơn. Các triển khai HAL được đóng gói thành các mô-đun và được tải bởi hệ thống Android vào thời điểm thích hợp.
- **Nhân Linux (Linux Kernel):** Android sử dụng phiên bản của nhân Linux

với một số bổ sung đặc biệt như Low Memory Killer (hệ thống quản lý bộ nhớ tích cực hơn trong việc quản lý bộ nhớ), wake locks (một dịch vụ hệ thống PowerManager), trình điều khiển IPC Binder và các tính năng quan trọng khác cho một nền tảng nhúng di động. Những bổ sung này chủ yếu dành cho chức năng hệ thống và không ảnh hưởng đến việc phát triển trình điều khiển.

1.4.3 Giới thiệu về Android Studio

Android Studio là môi trường phát triển tích hợp (IDE) chính thức để phát triển ứng dụng Android, dựa trên IntelliJ IDEA. Dựa trên các trình soạn thảo mã và công cụ phát triển mạnh mẽ của IntelliJ, Android Studio còn cung cấp thêm nhiều tính năng giúp lập trình viên nâng cao năng suất khi xây dựng ứng dụng Android [30].

1.4.4 Giới thiệu về NDK

Native Development Kit (NDK) là một bộ công cụ giúp lập trình viên sử dụng code C/C++ trong Android, cung cấp các thư viện giúp chúng ta có thể sử dụng để quản lý các hoạt động của thiết bị, truy nhập vào các bộ phận vật lý của máy như các cảm biến sensors, màn hình cảm ứng,...

Việc sử dụng NDK giúp cải thiện được hiệu năng của ứng dụng do NDK được viết bằng C/C++ có tốc độ chạy rất cao, nhờ đó giúp cho tốc độ của ứng dụng được cải thiện đáng kể. Ngoài ra native code cho phép lập trình viên có thể sử dụng một số các tính năng của bộ xử lý trong khi Android SDK thì không thể, và có thể cải thiện các đoạn mã ở mức assembly [21].