TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

─────── \* ───────



**BÁO CÁO HỌC PHẦN**

**PROJECT I**

**Đề tài: Xây dựng phần mềm mã hóa**

**và giải mã tệp tin**

Giảng viên hướng dẫn : *ThS. Bùi Trọng Tùng*

Sinh viên thực hiện: *Nguyễn Huy Hoàng - 20180084*

Lớp: 709159 – Kỳ: 2021.1

***Hà Nội, tháng 1 năm 2022***

# MỤC LỤC

[MỤC LỤC 2](#_Toc93584886)

[DANH SÁCH TỪ VIẾT TẮT VÀ THUẬT NGỮ 3](#_Toc93584887)

[DANH SÁCH BẢNG VÀ HÌNH ẢNH 4](#_Toc93584888)

[1. Lời mở đầu 5](#_Toc93584889)

[1.1. Yêu cầu 5](#_Toc93584890)

[1.2. Mục tiêu 5](#_Toc93584891)

[2. Cơ sở lý thuyết 6](#_Toc93584892)

[2.1. Mã hóa 6](#_Toc93584893)

[2.1.1. Mã hóa bất đối xứng 6](#_Toc93584894)

[2.1.2. Mã xác thực 7](#_Toc93584895)

[2.2. Ngôn ngữ lập trình Java 8](#_Toc93584896)

[2.3. Thư viện Bouncy Castle 10](#_Toc93584897)

[3. Thiết kế chương trình 11](#_Toc93584898)

[3.1. Giới thiệu các Mô-đun chính của chương trình 11](#_Toc93584899)

[3.1.1. Sinh khóa 11](#_Toc93584900)

[3.1.2. Chọn tệp tin cần sử lý 11](#_Toc93584901)

[3.1.3. Mã hóa và giải mã 11](#_Toc93584902)

[3.2. Các biểu đồ chính 12](#_Toc93584903)

[3.2.1. Biểu đồ Use-Case 12](#_Toc93584904)

[3.2.2. Biểu đồ Sequence 13](#_Toc93584905)

[3.2.3. Biểu đồ lớp 14](#_Toc93584906)

[3.3. Chi tiết các chức năng 15](#_Toc93584907)

[3.3.1. Nhập chuỗi sinh khóa 16](#_Toc93584908)

[3.3.2. Chọn File cần xử lý. 17](#_Toc93584909)

[3.3.3. Chọn đường dẫn lưu File 19](#_Toc93584910)

[3.3.4. Mã hóa 19](#_Toc93584911)

[3.3.5. Giải mã 20](#_Toc93584912)

[4. Thử nghiệm và kết quả. 21](#_Toc93584913)

[4.1. Kiểm tra lỗi phát sinh. 21](#_Toc93584914)

[4.2. Thử nghiệm với số lượng kích thước file 22](#_Toc93584915)

[5. Kết luận 23](#_Toc93584916)

[DANH MỤC TÀI LIỆU THAM KHẢO 24](#_Toc93584917)

# DANH SÁCH TỪ VIẾT TẮT VÀ THUẬT NGỮ

|  |  |
| --- | --- |
| BTL | Bài tập lớn |
| KĐX | Khóa đối xứng |
| CSK | Chuỗi sinh khóa |
| MAC | Messasge Authentication Code |
| HMAC | Keyd-Hash Messasge Authentication Code |
| AES | Advanced Encryption Standard |
| CBC | Cipher Block Chaining |
| JRE | Java Runtime Environment |

# DANH SÁCH BẢNG VÀ HÌNH ẢNH

[Hình 1. Sơ đồ mã hóa 7](#_Toc93588816)

[Hình 2. Giải thích HMAC 8](#_Toc93588817)

[Hình 3. Các phiên bản HMAC 8](#_Toc93588818)

[Hình 4. Logo Java 8](#_Toc93588819)

[Hình 5. Ảnh nền Bouncy Castle 10](#_Toc93588820)

[Hình 6. Sinh khóa 11](#_Toc93588821)

[Hình 7. Mã hóa và giải mã 11](#_Toc93588822)

[Hình 8. Biểu đồ Use-case 12](#_Toc93588823)

[Hình 9. Biểu đồ Sequence 13](#_Toc93588824)

[Hình 10. Biểu đồ lớp 14](#_Toc93588825)

[Hình 11. Giao diện tổng quan 15](#_Toc93588826)

[Hình 12. Nhập chuỗi sinh khóa (ẩn) 16](#_Toc93588827)

[Hình 13. Nhập chuỗi sinh khóa (hiển thị) 16](#_Toc93588828)

[Hình 14. Lưu khóa vào file 16](#_Toc93588829)

[Hình 15. Dùng khóa bằng kéo thả 17](#_Toc93588830)

[Hình 16. Chọn file bằng JFileChoser 17](#_Toc93588831)

[Hình 17. Chọn File bằng kéo thả 18](#_Toc93588832)

[Hình 18. Hiển thị file đã chọn 18](#_Toc93588833)

[Hình 19. Chọn đường dẫn lưu File 19](#_Toc93588834)

[Hình 20. Đường dẫn đã chọn 19](#_Toc93588835)

[Hình 21. File config.txt 19](#_Toc93588836)

[Hình 22. Thông báo mã hóa thành công 20](#_Toc93588837)

[Hình 23. Thông báo giải mã thành công 20](#_Toc93588838)

[Hình 24. Lỗi chưa chọn File 21](#_Toc93588839)

[Hình 25. Lỗi chưa chọn đường dẫn 21](#_Toc93588840)

[Hình 26. Lỗi File mã hóa bị lỗi 21](#_Toc93588841)

[Hình 27. Lỗi file lưu khóa 21](#_Toc93588842)

[Hình 28. Lỗi key quá dài 22](#_Toc93588843)

[Hình 29. Bảng thử nghiệm với kích thước File 22](#_Toc93588844)

# Lời mở đầu

## Yêu cầu

Trong thời đại công nghệ thông tin phát triển, có muôn vàn các tác động kéo theo. Nhu cầu về sử dụng Internet tốc độ cao dẫn đến việc hình thành công nghệ cáp quang. Nhu cầu về tốc độ xử lý của máy tính thúc đẩy công nghệ về phần cứng máy tính. Nhu cầu lưu trữ dẫn đến việc hình hành công nghệ phát triển chip nhớ, ổ cứng thể rắn(SSD) và lưu trữ đám mây. Đi kèm với phát triển cũng có không ít những rủi ro mà các nhà phát triển và người dùng phải quan tâm và mối quan tâm lớn nhất chính là về vấn đề bảo mật.

Việc lưu giữ và bảo mật thông tin trừ trước đến nay luôn là vấn đề được nhiều người dùng để ý đến và gặp không ít vấn đề. Trong đó có bảo mật thông tin người dùng trên các trang điện tử và bảo mật lưu trữ dữ liệu, từ đó đặt ra nhu cầu về việc mã hóa dữ luêụ. Em đã tìm hiểu và quyết định xây dựng chương trình mã hóa và giải mã tệp tin sử dụng trên máy tính cá nhân.

## Mục tiêu

Chương trình xây dựng phải đáp ứng được nhu cầu về mã hóa và giải mã của người sử dụng và đáp ứng một số chức năng cần thiết. Công nghệ mã hóa phải sử dụng các công nghệ mới, có độ an toàn cao và được nhiều nhà phát triển tin dùng. Đồng thời chương trình phải nhỏ gọn, dễ cài đặt, dễ sử dụng và thân thiện với người dùng. Để phát triển ứng dụng, em quyết định dùng ngôn ngữ lập trình Java và thư viện mã hóa sử dụng thư viện Bouncy Castle.

# Cơ sở lý thuyết

## Mã hóa

### Mã hóa bất đối xứng

Mã hóa dữ liệu là quá trình chuyển dữ liệu từ dạng này sang dạng khác hoặc sang dạng mà chỉ có người có quyền truy cập vào khóa (key) giải mã hoặc có mật khẩu mới có thể đọc được dữ liệu đó. Trong mã hóa đối xứng sẽ sử dụng duy nhất một khóa, khóa giống nhau sẽ vừa được dùng để mã hóa, vừa được dùng để giải mã các tệp tin. Phương pháp mã hóa thông tin này đã được sử dụng khá phổ biến từ nhiều thập kỷ với mục đích tạo ra cách thức liên lạc bí mật giữa chính phủ với quân đội. Ngày nay, các thuật toán khóa đối xứng được ứng dụng rộng rãi trên nhiều hệ thống máy tính khác nhau nhằm tăng cường bảo mật cho dữ liệu. Em quyết định sử dụng mã hóa đối xứng vì nó đơn giản và thuật tiện cho người xử dụng.

Đây được cho là kỹ thuật mã hóa đơn giản, được sử dụng phổ biến nhất và đã có từ rất lâu với một số đặc điểm nổi bật như:

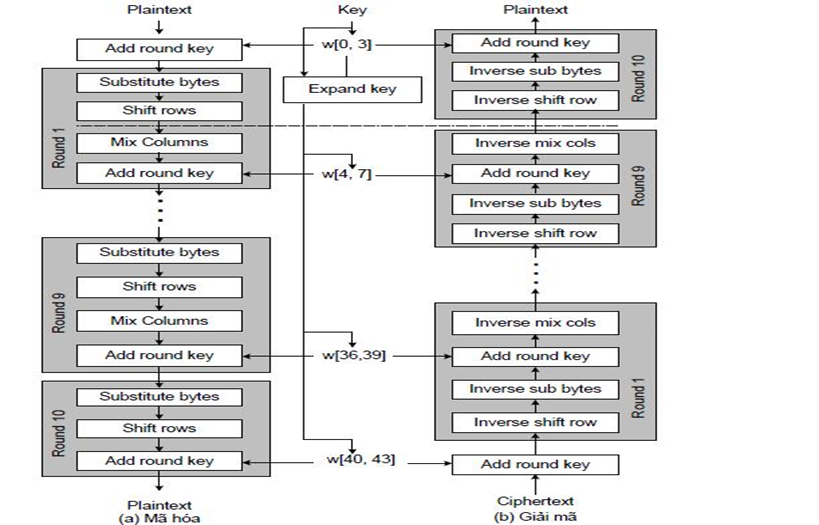
* Do thuật toán mã hóa đối xứng ít phức tạp hơn và có thể thực thi nhanh hơn, đây là kỹ thuật được đặc biệt ưa thích trong các hoạt động truyền tải dữ liệu hàng loạt.
* Văn bản gốc được mã hóa bằng một key trước khi gửi đi, và chính key này cũng sẽ được người nhận sử dụng để giải mã dữ liệu.
* Một số thuật toán mã hóa đối xứng được sử dụng phổ biến nhất bao gồm AES-128, AES-192 và AES-256.

Quá trình mã hóa bao gồm việc chạy văn bản thô (đầu vào) thông qua một thuật toán mã hóa còn gọi là mật mã (cipher) sẽ lần luợt tạo ra các bản mã - ciphertext (đầu ra). Khi sơ đồ mã hóa đủ mạnh thì cách duy nhất để đọc và truy cập được các thông tin chứa trong các bản mã là sử dụng khóa tương ứng để giải mã. Quá trình giải mã về cơ bản sẽ chuyển đổi các bản mã trở về dạng văn bản thô ban đầu.

Mức độ bảo mật của các hệ thống 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 brute force. Ví dụ, để dò ra mã hóa của 1 khóa 128-bit thì sẽ mất tới vài tỷ năm nếu sử dụng các phần cứng máy tính thông thường.

Trong chương trình em quyết định sử dụng AES-128 với kích thước khối là 16 bytes vì đây là thuật toán được sử dụng nhiều nhất nên sẽ tham khảo và tìm mã nguồn mở. AES (viết tắt của từ tiếng anh: Advanced Encryption Standard, hay Tiêu chuẩn mã hóa nâng cao) là một thuật toán mã hóa khối được chính phủ Mỹ áp dụng làm tiêu chuẩn mã hóa. Quá trình chuẩn bị gồm xây dựng bảng S-box (thuận và nghịch), sinh khóa phụ. Sơ đồ mã hóa được cho như hình.

* Hàm **AddRoundKey**: một khóa vòng được cộng với state bằng một phép XOR theo từng bit đơn giản.
* Hàm **SubBytes**: thay thế mỗi byte riêng rẽ của state trong khối bằng một giá trị mới sử dụng bảng thay thế (S - box).
* Hàm **ShiftRow**: các byte trong ba hàng cuối cùng của trạng thái được dịch vòng đi các số byte khác nhau (độ lệch).
* Hàm **MixColumn**: tính toán trên từng cột của state như phép nhân một ma trận.



Hình . Sơ đồ mã hóa

Thuật toán giải mã khá giống với thuật toán mã hóa về mặt cấu trúc nhưng 4 hàm sử dụng là 4 hàm ngược của quá trình mã hóa. Thiết kế và độ dài khóa của thuật toán AES là đủ an toàn để bảo vệ các thông tin.

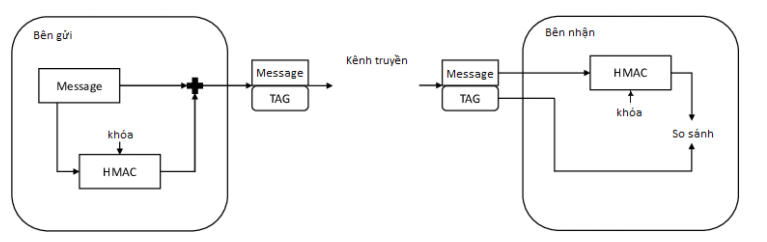
### Mã xác thực

Mã xác thực thông báo Message Authentication Code (MAC) là một đoạn mã được tạo ra nhằm mục đích xác thực tính toàn vẹn và nguồn gốc của thông báo. Đây là một cơ chế quan trọng để duy trì tính toàn vẹn và không thể chối bỏ dữ liệu. Với các giao thức trực tuyến, mã xác thực thông báo được sử dụng phổ biến và với việc kết hợp với các cơ chế khác (như chữ ký số) để đảm bảo tính xác thực giữa các bên tham gia giao dịch.

Mục đích của hàm MAC là đảm bảo để hai (hay nhiều) bên tham gia giao dịch khi có chung khóa bí mật có thể giao dịch với nhau, kèm theo khả năng có thể phát hiện được thay đổi của thông báo trong quá trình vận chuyển, nhằm tránh các tấn công làm thay đổi thông báo. Thuật toán MAC dựa trên thông báo đầu vào và khóa mật để tạo ra thẻ MAC (MAC tag). Thông báo và thẻ MAC được gửi tới người nhận, người nhận tính lại giá trị MAC và so sánh nó với giá trị thẻ MAC nhận được. Nếu hai giá trị thẻ MAC trùng nhau thì coi như thông báo chính xác, ngược lại thông báo coi là đã bị thay đổi.

Đối với kẻ tấn công giả mạo thông báo, mục tiêu đặt ra là phải phá được hàm MAC, việc này khó tương đương với việc phá khóa bảo vệ thông báo. Trên thực tế, các giao thức thường chia thông báo dài làm nhiều đoạn nhỏ và chúng được xác thực độc lập với nhau, dẫn đến phát sinh tấn công lặp lại (Replay Attack). Do vậy khi thiết kế giao thức có sử dụng MAC cần rất thận trọng.

Để giúp các nhà phát triển ứng dụng tích hợp MAC vào sản phẩm khác nhau, viện tiêu chuẩn và công nghệ quốc gia của Mỹ (NIST) đưa ra tiêu chuẩn mã xác thực thông báo sử dụng hàm một chiều có khóa HMAC (Keyd-Hash Messasge Authentication Code).



Hình . Giải thích HMAC

Đây là mã xác thực sử dụng hàm băm, khi đã có các hàm hash tốt, chúng ta muốn có các mã xác thực thông báo MAC dựa trên các hàm hash đó. Vì hàm hash thông thường nhanh hơn và mã nguồn của hàm hash được phổ biến rộng rãi hơn, nên việc sử dụng chúng tạo nên MAC sẽ hiệu quả. HMAC được thiết kế theo chuẩn Internet RFC2104, sử dụng hàm hash trên mẩu tin:

HMACK = Hash [(K+ XOR opad) || Hash [(K+ XOR ipad) || M)]]

*(K+ là khóa đệm mở rộng của K, opad và ipad là các hằng bộ đệm đặc biệt, M là thông báo)*

Các hàm xác thực thông điệp HMAC được xây dựng dựa trên mô hình mật mã khóa đối xứng. Do đó cách sinh khóa để sử dụng cho hàm này tương tự như cách thức sinh khóa của hệ mật mã AES. Sau đây là một số hàm HMAC điển hình:

|  |  |  |  |
| --- | --- | --- | --- |
| Hàm HMAC | Hàm băm | Kích thước khối(bit) | Kích thước mã HMAC(bit) |
| HMAC-MD5 | MD5 | 512 | 128 |
| HMAC-SHA1 | SHA-1 | 512 | 160 |
| HMAC-SHA224 | SHA-224 | 512 | 224 |
| HMAC-SHA256 | SHA-256 | 512 | 256 |
| HMAC-SHA384 | SHA-384 | 1024 | 384 |
| HMAC-SHA512 | SHA-512 | 1024 | 512 |

Hình . Các phiên bản HMAC

Độ an toàn của các hàm tăng theo thứ tự đã liệt kê, tương ứng với tốc độ tính toán giảm dần. Hiện nay hàm HMAC-MD5 không còn an toàn. Trong chương trình em sử dụng hàm HMAC-SHA256 vì đây là hàm được sử dụng phổ biến hiện nay.

## Ngôn ngữ lập trình Java

Java là một ngôn ngữ lập trình hướng đối tượng (OOP) và dựa trên các lớp (class), ban đầu được phát triển bởi Sun Microsystems do James Gosling khởi xướng và phát hành vào năm 1995. Khác với phần lớn ngôn ngữ lập trình thông thường, thay vì biên dịch mã nguồn thành mã máy hoặc thông dịch mã nguồn khi chạy, Java được thiết kế để biên dịch mã nguồn thành bytecode, bytecode sau đó sẽ được môi trường thực thi (runtime environment) chạy.



Hình . Logo Java

Là một ngôn ngữ lập trình hướng đối tượng vì vậy Java cũng có 4 đặc điểm chung sau đây:

* Tính trừu tượng (Abstraction): là tiến trình xác định và nhóm các thuộc tính, các hành động liên quan đến một thực thể đặc thù, xét trong mối tương quan với ứng dụng đang phát triển.
* Tính đa hình (Polymorphism): cho phép một phương thức có các tác động khác nhau trên nhiều loại đối tượng khác nhau. Với tính đa hình, nếu cùng một phương thức ứng dụng cho các đối tượng thuộc các lớp khác nhau thì nó đưa đến những kết quả khác nhau. Bản chất của sự việc chính là phương thức này bao gồm cùng một số lượng các tham số.
* Tính kế thừa (Inheritance): điều này cho phép các đối tượng chia sẻ hay mở rộng các đặc tính sẵn có mà không phải tiến hành định nghĩa lại.
* Tính đóng gói (Encapsulation): là tiến trình che giấu việc thực thi những chi tiết của một đối tượng đối với người sử dụng đối tượng ấy.

Ngoài ra Java còn có một số đặc điểm sau:

* Độc lập nền (Write Once, Run Anywhere): Không giống như nhiều ngôn ngữ lập trình khác như C và C ++, khi Java được biên dịch, nó không được biên dịch sang mã máy cụ thể, mà thay vào đó là mã bytecode chạy trên máy ảo Java (JVM). Điều này đồng nghĩa với việc bất cứ thiết bị nào có cài đặt JVM sẽ có thể thực thi được các chương trình Java.
* Đơn giản: Học Java thật sự dễ hơn nhiều so với C/C++, nếu bạn đã quen với các ngôn ngữ lập trình hướng đối tượng thì việc học Java sẽ dễ dàng hơn. Java trở nên đơn giản hơn so với C/C++ do đã loại bỏ tính đa kế thừa và phép toán con trỏ từ C/C++.
* Bảo mật: Java hỗ trợ bảo mật rất tốt bởi các thuật toán mã hóa như mã hóa một chiều (one way hashing) hoặc mã hóa công cộng (public key)…
* Thông dịch: Java là một ngôn ngữ lập trình vừa biên dịch vừa thông dịch. Chương trình nguồn viết bằng ngôn ngữ lập trình Java có đuôi \*.java và được biên dịch thành tập tin có đuôi \*.class sau đó được trình thông dịch thông dịch thành mã máy.
* Đa luồng: Với tính năng đa luồng Java có thể viết chương trình có thể thực thi nhiều task cùng một lúc. Tính năng này thường được sử dụng rất nhiều trong lập trình game.
* Hướng đối tượng: Hướng đối tượng trong Java tương tự như C++ nhưng Java là một ngôn ngữ lập trình hướng đối tượng hoàn toàn. Tất cả mọi thứ đề cập đến trong Java đều liên quan đến các đối tượng được định nghĩa trước, thậm chí hàm chính của một chương trình viết bằng Java (đó là hàm main) cũng phải đặt bên trong một lớp. Hướng đối tượng trong Java không có tính đa kế thừa (multi inheritance) như trong C++ mà thay vào đó Java đưa ra khái niệm interface để hỗ trợ tính đa kế thừa.
* Hiệu suất cao: Nhờ vào trình thu gom rác (garbage collection), giải phóng bộ nhớ đối với các đối tượng không được dùng đến.
* Linh hoạt: Java được xem là linh hoạt hơn C/C ++ vì nó được thiết kế để thích ứng với nhiều môi trường phát triển.

Java được sử dụng nhiều trong các ứng dụng và phần mềm sau:

* Các ứng dụng yêu cầu tính bảo mật cao
* Các ứng dụng cho hệ điều hành Android
* Điện toán đám mây
* Không gian nhúng
* Trong lĩnh vực giao dịch chứng khoán

## Thư viện Bouncy Castle

Thư viện Bouncy Castle cung cấp các hàm lập trình sử dụng mật mã học cho nhiều mục đích khác nhau. Vào những năm cuối thập niên 1990, hai lập trình viên, những người đã quá mệt mỏi bởi việc phải liên tục thay đổi các thư viện mật mã phức tạp khi phát triển các dự án trên nền tảng J2SE, bắt tay vào xây dựng thư viện mới có thể tương thích với nhiều hệ thống khác nhau. Tháng 05-2000, phiên bản đầu tiên của Bouncy Castle viết bằng Java được giới thiệu, cung cấp Provider cho thư viện mật mã JCA/JCE, hỗ trợ tất cả các nền tảng của Java. Vào năm 2004, nhóm giới thiệu một phiên bản khác cho ngôn ngữ lập trình C#. Với sự đơn giản khi sử dụng, Bouncy Castle nhanh chóng được các lập trình viên ưu chuộng, đặc biệt là các lập trình viên J2ME do vào thời điểm đó, rất ít thư viện mật mã hoạt động tốt trên nền tảng này. Hiện nay, Bouncy Castle là một trong những thư viện mật mã phổ biến nhất cho các ứng dụng di động trên nền tảng Android. Theo thống kê của website AppBrain, có trên 7000 ứng dụng Android hiện tại đang sử dụng Bouncy Castler, trong đó có những ứng dụng có lượt download lên đến con số 1.000.000.000.



Hình . Ảnh nền Bouncy Castle

Bouncy Castle hỗ trợ hầu hết các hàm mật mã học tiêu chuẩn, gồm hai thành phần chính:

* JCA/JCE Provider tương thích với thư viện JCA/JCE có sẵn của Java
* Lightweight API cung cấp số lượng hàm ít hơn JCE Provider, nhưng đơn giản để có thể chạy trên các thiết bị có bộ nhớ giới hạn.

Bên cạnh đó, Bouncy Castle cũng cung cấp các thư viện cho nhiều ngữ cảnh sử dụng mật mã học như các chuẩn PGP, S/MIME trong email, hay Java Secure Socket Extension, DNSSec… Mã nguồn và tài liệu đầy đủ để tìm hiểu về Bouncy Castle tại địa chi sau:

*https://downloads.bouncycastle.org/java/lcrypto-jdk15on-166.zip*

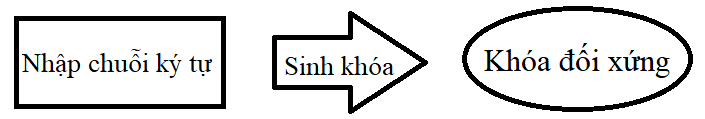
Trong chương trình, em sử dụng thư viện trên để chạy các hàm sinh khóa mã hóa/giải mã, khởi tạo bộ mật mã, mã hóa dữ liệu và giải mã dữ liệu.

# Thiết kế chương trình

## Giới thiệu các Mô-đun chính của chương trình

### Sinh khóa

Người sử dụng nhập vào một chuỗi ký tự, chuỗi ký tự này được dùng để sinh khóa đối xứng. Chuỗi ký tự nhập vào khác nhau sẽ sinh được các khóa đối xứng khác nhau, khóa này phục vụ múc đích mã hóa và giải mã sau này.



Hình . Sinh khóa

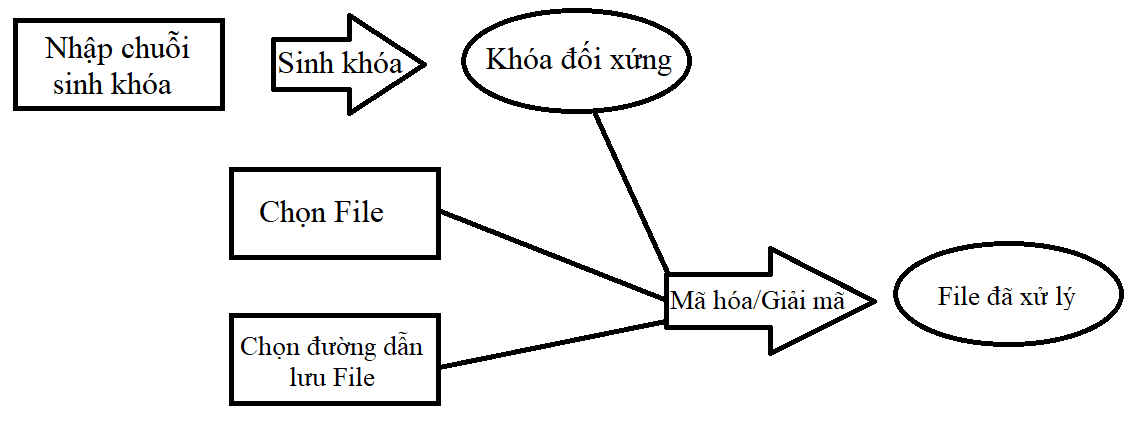
### Chọn tệp tin cần sử lý

Người sử dụng có thể thoải mái chọn tệp tin để xử lý, có thể cùng lúc chọn nhiều tiệp tin, có thể chọn thư mục chứa tệp tin hoặc chứa thư mục con, có thể chọn nhiều tệp tin tại nhiều vị trí khác nhau trên bộ nhớ. Những tệp tin được được chọn này sẽ được xử lý mã hóa hoặc giải mã. Có hai cách để chọn tệp tin:

* File Chosser: Người sử dụng chọn đường dẫn đến tệp tin cần xử lý.
* Drap and Drop: Người sử dụng kéo thả tệp tin cần xử lý vào cửa sổ chương trình.

### Mã hóa và giải mã

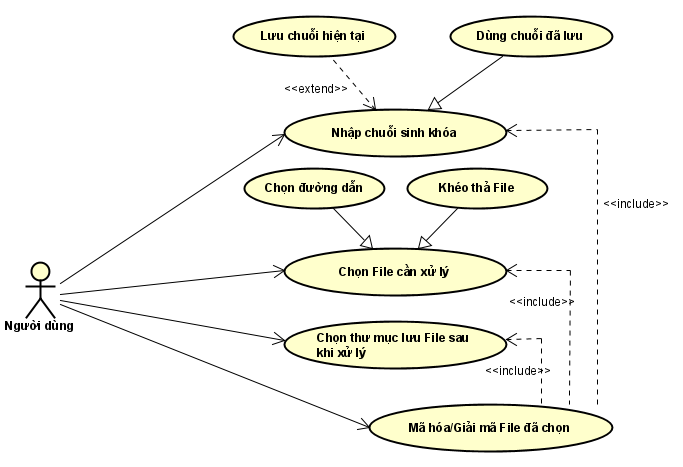
Sau khi đã chọn tệp tin và nhập chuỗi sinh khóa, người sử dụng sau đó sẽ phải chọn đường dẫn đến thư mục lưu tệp tin sau khi xử lý. Sau đó tùy vào mục đích sẽ chọn mã hóa hay giải mã tệp tin đã chọn.



Hình . Mã hóa và giải mã

## Các biểu đồ chính

### Biểu đồ Use-Case



Hình . Biểu đồ Use-case

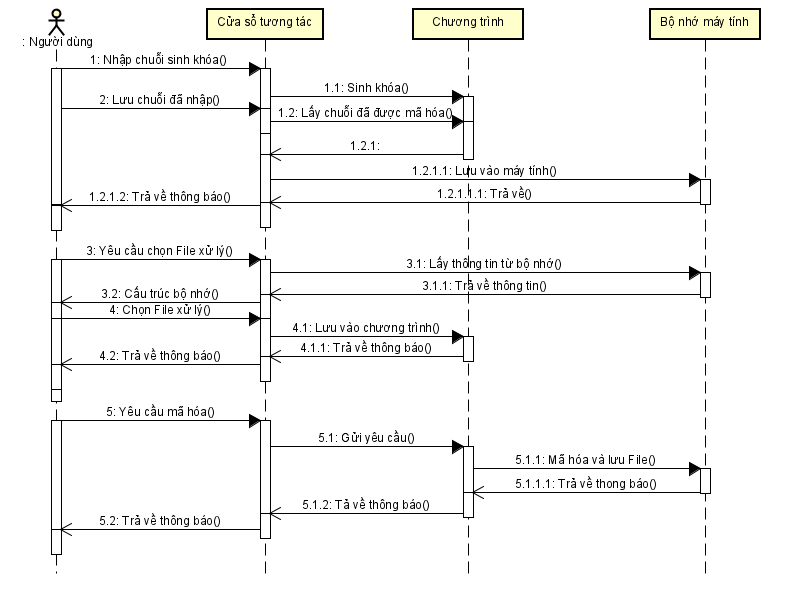
Người dùng sẽ có các chức năng:

* **Nhập chuỗi sinh khóa**: Người dùng nhập vào một chuỗi phục vụ cho mục đích sinh khóa đối xứng, sau đó có thể lưu chuỗi đã nhập vào file để dùng cho sau này. Khi đã có file khóa, người dùng có thể dùng file đó thay cho chuỗi sinh khóa để tạo khóa đối xứng.
* **Chọn File cần xử lý**: Người dùng chọn File cần xử lý bằng cách kéo thả File vào cửa sổ chương trình hoặc chọn đường dẫn đến File.
* **Chọn thư mục lưu File sau khi xử lý**: Chọn đường dẫn đến thư mục lưu file sau khi mã hóa hoặc giải mã.
* **Mã hóa/Giải mã File đã chọn**: Cho phép chương tình tiến hành Mã hóa/ Giải mã File đã chọn. Thao tác này yêu cầu người dùng phải thực hiện xong 3 thao tác ở trên.

### Biểu đồ Sequence

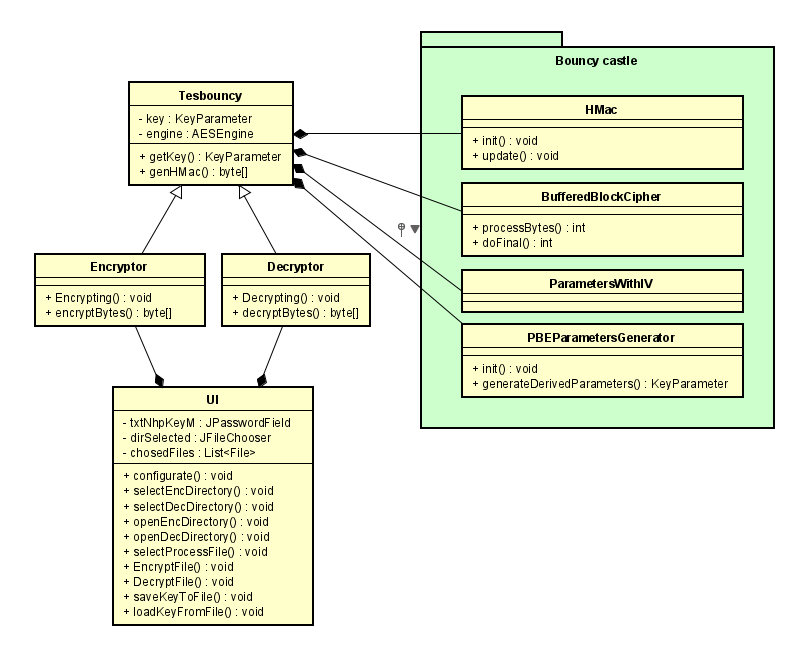
Khi người dùng sử dụng chức năng chọn File theo đường dẫn từ cửa sổ tương tác, cửa sổ sẽ lấy thông tin cấu trúc các thư mục từ bộ nhớ và trả về cho người sử dụng để lựa chọn. Sau khi người sử dụng đã chọn xong File, cửa sổ tương báo báo về cho chương trình lưu File đó lại.

Khi người dùng sử dụng chức năng Mã hóa trên cửa sổ tương tác, cửa sổ tương tác sẽ gửi đến chương trình yêu cầu mã hóa File đã chọn, sau khi mã hóa xong chương trình lưu file đó vào bộ nhớ máy tính rồi trả về kết quả cho cửa sổ, cửa sổ hiển thị kết quả mã hóa cho người dùng.



Hình . Biểu đồ Sequence

### Biểu đồ lớp



Hình . Biểu đồ lớp

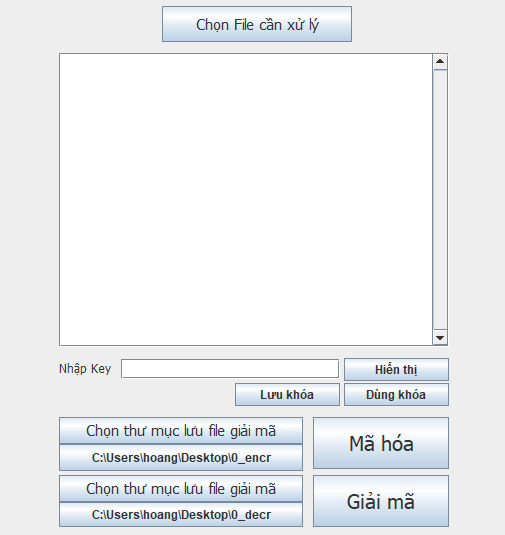
Các lớp xây dựng lại từ chương trình:

* **UI**: lớp UI sẽ lưu các chức năng liên quan đến thao tác của người dử dụng
  + **txtNhpKeyM**: Lưu chuỗi sinh khóa người dùng nhập vào.
  + **dirSelected**: Lưu đường dẫn lưu File sau khi xử lý.
  + **chosedFiles**: Lưu đường dẫn đến các File cần xử lý.
  + **configurate()**: Cấu hình lại chương trình theo config đã lưu trước đó.
  + **selectDirectory()**: Chọn đường dẫn lưu File sau khi xử lý.
  + **openDirectory()**: Mở mở mực theo đường dẫn.
  + **selectProcessFile()**: Chọn File cần xử lý.
  + **EncryptFile()**: Nhận đầu vào là chosedFiles và txtNhpKeyM sau đó tiến hành mã hóa File.
  + **DecryptFile()**: Nhận đầu vào là chosedFiles và txtNhapKeyM sau đó tiến hành mã hóa File.
* **Tesbouncy**: Lớp này lưu các attribute và các phương phức liên quan đến mã hóa và giải mã.
  + **getKey()**: Nhận đầu vào là chuỗi sinh khóa, đầu ra là khóa đối xứng theo kiểu KeyParameter.
  + **genHMac()**: Nhận đầu vào là File dữ liệu và khóa đối xứng, đầu ra là thông điệp HMAC tương ứng.
* **Encryptor** và **Decryptor**: Đây là hai lớp dùng để mã hóa và giải mã, được kế thừa từ lớp Tesbouncy và tùy chức năng được chia làm hai lớp mã hóa và giải mã.
  + **Encrpyting() / Decrypting():** Nhận đầu vào là danh sách File cần xử lý, khóa đối xứng và thư mục lưu file, sau đó tiến hànhMã hóa / Giải mã file theo khóa rồi lưu vào thư mục lưu file.
  + **encrpytBytes() / decryptBytes():** Nhận đầu vào là một dãy byte và chuỗi sinh khóa, đầu ra là dãy byte đã được mã hóa/giải mã (dùng cho việc mã hóa chuỗi sinh khóa để lưu vào file).

Các lớp trong packet Bouncy castle là của thư viện Bouncy Castle.

* **HMac**: tạo thông điệp HMAC kiểm tra tính toàn vẹn.
* **BufferedBlockCipher**: tạo ra Bộ mã mật từ AESEngine dùng cho mã hóa hoặc giải mã.
* **ParametersWithIV**: do thuật toán AES ở chế độ CBC sử dụng vec-tơ khởi tạo (Initial Vector) nên khóa dưới kiểu KeyParameter cần được đóng gói vào đối tượng kiểu ParametersWithIV.
* **PBEParametersGenerator**: tạo ra generator để sinh khóa đối xứng từ chuỗi sinh khóa.

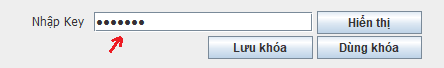
## Chi tiết các chức năng



Hình . Giao diện tổng quan

### Nhập chuỗi sinh khóa

Người dùng nhập chuỗi sinh khóa vào ô như hình dưới, chuỗi sẽ được lưu vào biến txtNhpKeyM. Các ký tự nhập vào có thể bất kỳ kí tự gì trong bộ UTF-8.



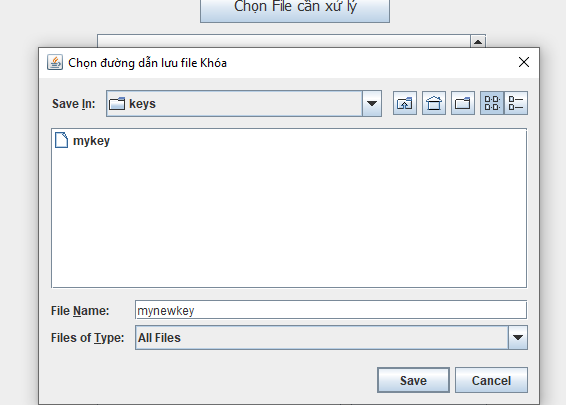
Hình . Nhập chuỗi sinh khóa (ẩn)

Để cho người dùng dễ hiểu, tiêu đề của của ô đó sẽ là “Nhập Key” giống như nhập mật khẩu. Nút “Hiển thị” dùng để hiện rõ ra chuỗi nhập vào là gì. Đây là chức năng của JtoggleButton thay đổi cấu hình của txtNhpKeyM trong UI.



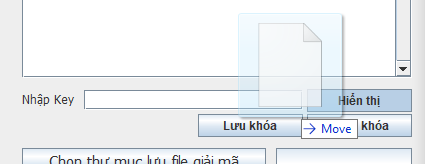
Hình . Nhập chuỗi sinh khóa (hiển thị)

Người dùng có thể lưu chuỗi đã nhập vào file bằng cách chọn “Lưu khóa”, một cửa sổ JfileChoser sẽ được hiện lên, người dùng chọn đường dẫn lưu file khóa và tên file, sau đó chọn “Save”. Thông tin trong ô nhập chuỗi (txtNhpKeyM) sẽ được mã hóa bằng phương thức encryptBytes() của lớp Encryptor sau đó được lưu vào file đã chọn.



Hình . Lưu khóa vào file

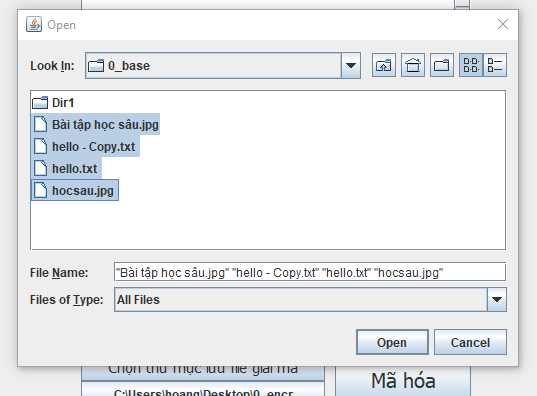
Ngoài cách nhập chuỗi sinh khóa từ bàn phím, người dùng có thể dùng File khóa đã lưu trước đó để nhập chuỗi bằng cách ấn phím “Dùng khóa” hoặc kéo thả file khóa vào ô nhập chuỗi. Sau khi xác định được file khóa, chương trình sẽ đọc và giải mã file khóa, thông tin sau khi giải mã sẽ được truyền vào ô nhập chuỗi (txtNhpKeyM).



Hình . Dùng khóa bằng kéo thả

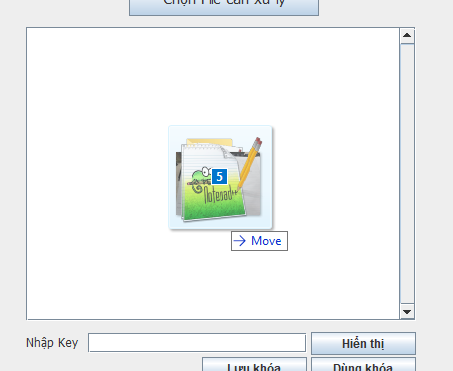
### Chọn File cần xử lý.

Có hai cách để người dùng chọn File, đầu tiên là cách sử dụng JFileChoser bằng cách ấn vào “Chọn File cần xử lý” ở phía trên cùng của cửa sổ. Sau đó cửa sổ JfileChoser hiện ra rồi chọn đường dẫn đến file cần xử lý, cùng lúc có thể chọn được nhiều File và thư mục.



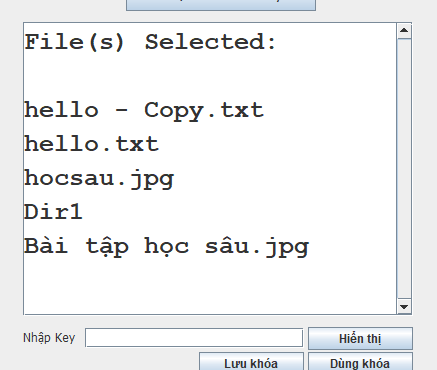
Hình . Chọn file bằng JFileChoser

Cách thứ hai là kéo thả một hoặc nhiều File vào cửa sổ của chương trình. Đây là chức năng của TransferHandler với cài đặt DataFlavor.javaFileListFlavor là List<File>.



Hình . Chọn File bằng kéo thả

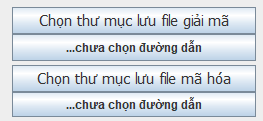
Các đường dẫn sẽ được lưu vào danh sách chosedFile. Thông tin hiển thị trên JtextArea sẽ là danh sách các file đã được chọn.



Hình . Hiển thị file đã chọn

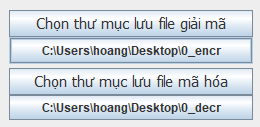
### Chọn đường dẫn lưu File

Người dùng chọn đường dẫn lưu file mã hóa riêng và đường dẫn lưu file giải mã riêng bằng hai phím “Chọn thư mục lưu file giải mã” và “Chọn thư mục lưu file mã hóa”.



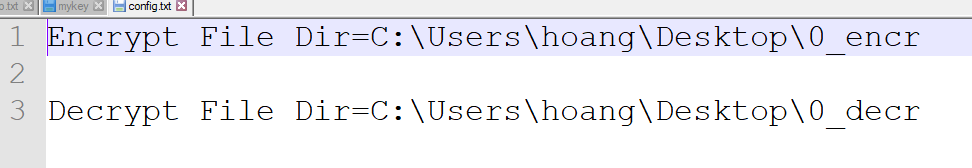
Hình . Chọn đường dẫn lưu File

Sau khi chọn xong, đường dẫn được hiện thị ở hai ô tương ứng phía dưới. Để mở nhanh thư mục, ta ấn vào hai đường dẫn đó. Các đường dẫn sẽ được lưu vào hai biến dirEncSelected(mã hóa) và dirDecSelected(giải mã) tương ứng.



Hình . Đường dẫn đã chọn

Các đường dẫn đến thư mục lưu file mã hóa, thư mục lưu file giải mã sẽ được lưu vào file config.txt, file được tạo ở thư mục lưu chương trình. Sau mỗi lần mở chương trình, chương trình sẽ tìm đến file config.txt và kiểm tra xem các đường dẫn trước đó là gì và đặt các đường dẫn hiện tại theo đường dẫn đó.



Hình . File config.txt

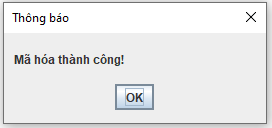
### Mã hóa

Sau khi đã chọn được File mã hóa và thư mục lưu File, người dùng mới có thể chọn “Mã hóa” để mã hóa file. Cửa sổ UI sẽ gọi đến phương thức Encrypting() của lớp Encryptor, đầu vào là danh sách đường dẫn các file, chuỗi sinh khóa và thư mục lưu file dirEncSelected đã chọn. Ta dùng đệ quy để đi vào các thư mục và mã hoá tất cả các file trong thư mục đó và thư mục con nếu có.

Đầu tiên, chuỗi sinh khóa sẽ được sử dụng để sinh khóa tương ứng (khóa đối xứng) bằng phương thức getKey() của lớp Tesbouncy. Do thuật toán AES ở chế độ CBC sử dụng vec-tơ khởi tạo (Initial Vector) nên khóa dưới kiểu KeyParameter cần được đóng gói vào đối tượng kiểu ParametersWithIV. Như vậy với mỗi file sẽ dùng khóa để sinh ra vectơ IV với kích thước 16 byte. Sau đó tạo một bộ mật mã encryptCipher từ vectơ IV. Đầu tiên tạo file ghi kết quả (file mã hóa) tại vị trí lưu file rồi ghi ra vectơ IV vào file đó. Dữ liệu từ file sẽ đọc lần lượt thành các đoạn kích thước (16\*1024) bytes và mã khóa theo từng khối bằng phương thức processBytes() của encryptCipher, mã hóa xong được khối nào ghi ra file kết quả khối đó. Khối cuối dùng được xử lý và padding bằng phương thức doFinal() sau đó cũng được ghi ra file kết quả. Vì encryptCipher là bộ PaddedBufferedBlockCipher với block size là 128 bit = 16 byte nên kích thước số byte của file kết quả luôn chia kết cho 16. Sau đó thông điệp Hmac được sinh ra bằng phương thức genHMac() của lớp Tesbouncy đầu vào là khóa và file kết quả, kích thước của Hmac là 256bit tương ứng 32 byte. Hmac sẽ được update trên toàn bộ dữ liệu trong file kết quả. Sau cùng file kết quả ghi thêm thông đệp Hmac ở cuối. Gọi kích thước của file gốc là X byte, kích thước của file kết quả (file mã hóa) là Y byte:

Y =

Nếu mã hóa thành công sẽ hiện thông báo:

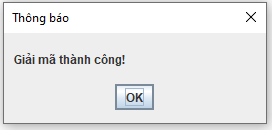


Hình . Thông báo mã hóa thành công

### Giải mã

Sau khi đã chọn được File mã hóa và thư mục lưu File, người dùng mới có thể chọn “Giải mã” để giải mã file. Cửa sổ UI sẽ gọi đến phương thức Decrypting() của lớp Encryptor, đầu vào là danh sách đường dẫn các file, chuỗi sinh khóa và thư mục lưu file dirDecSelected đã chọn. Ta dùng đệ quy để đi vào các thư mục và giải mã tất cả các file trong thư mục đó và thư mục con nếu có.

Đầu tiên, chuỗi sinh khóa sẽ được sử dụng để sinh khóa tương ứng (khóa đối xứng) bằng phương thức getKey() của lớp Tesbouncy. Đầu tiên là kiểm tra tính toàn vẹn của file. Giá trị Hmac cũ sẽ được lấy từ 32 byte cuối của file đầu vào. Giá tị Hmac mới được lấy từ khóa và file đầu vào bỏ đi 32 byte cuối bằng phương thức genHMac() của lớp Tesbouncy. Nếu Hmac cũ và mới khác nhau thì ra sẽ trả về thông báo có lỗi, nếu không thì ta tiến hành giải mã. Vectơ IV được lấy từ 16 byte đầu của file đầu vào, ta khởi tạo decryptCipher từ vectơ IV sau đó tiến hành giải mã từ sau file thứ 16 và bỏ qua 32 byte cuối. Tương tự như mã hóa, ta tạo file kết quả theo như đường dẫn và cũng sẽ đọc vào từng đoạn (16\*1028) byte và giải mã theo từng khối bằng phương thức processBytes(), giải mã được đến đâu ghi kết quả đến đó. Khối cuối cùng được xử lý bằng phương thức doFinal(). File sau khi mã hóa sẽ giống y như file gốc.

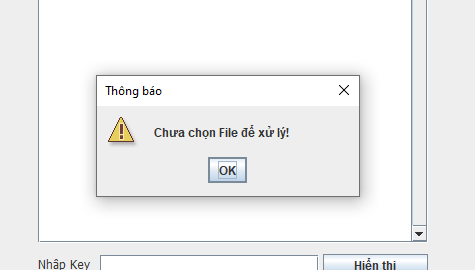


Hình . Thông báo giải mã thành công

# Thử nghiệm và kết quả.

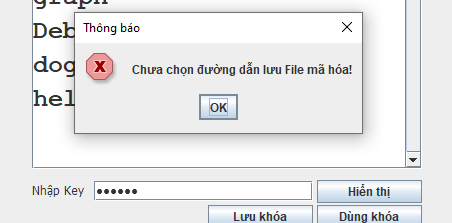
## Kiểm tra lỗi phát sinh.

* Người dùng chưa chọn File để xử lý:



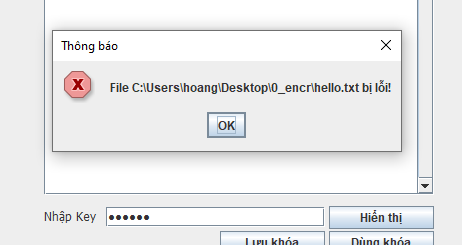
Hình . Lỗi chưa chọn File

* Người dùng chưa chọn thư mục lưu file sau khi xử lý:



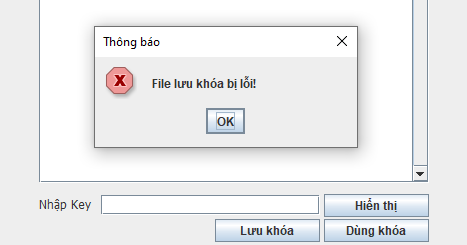
Hình . Lỗi chưa chọn đường dẫn

* File giải mã bị lỗi:



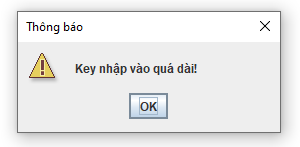
Hình . Lỗi File mã hóa bị lỗi

* File lưu khóa bị lỗi:



Hình . Lỗi file lưu khóa

* Key nhập vào quá dài



Hình . Lỗi key quá dài

## Thử nghiệm với số lượng kích thước file

Chương trình sử dụng thư viện JavaSE – 1.8 và Bouncy Castle bcprov-jdk15on-170. Hệ điều hành window 10.

Cấu hình máy thử nghiệm:

* CPU: Intel-3 8100
* RAM: 8GB
* Ổ cứng: SSD

|  |  |  |
| --- | --- | --- |
| Kích thước File (Mb) | Thời gian mã hóa (giây) | Thời gian giải mã (giây) |
| < 1MB | ~0s | ~0s |
| 5MB | 1s | 1s |
| 50MB | 9s | 8s |
| 200MB | 38s | 35s |
| 1GB | 195s | 181s |
| 5GB | ~17p | ~16h |

Hình . Bảng thử nghiệm với kích thước File

# Kết luận

Trong bài báo cáo này, em đã sử dụng ngôn ngữ lập trình Java và thư viện Bouncy Castle để xây dựng chương trình mã hóa và giải mã tệp tin trên hệ điều hành window 10 cho máy tính cá nhân. Trong quá trình xây dựng chương trình, em đã tìm hiểu được về thư viện Bouncy Castle, hiểu thêm về mã hóa và giải mã trên thuật toán AES với khóa đối xứng cũng như việc xác thực thông tin bằng HMAC. Đồng thời em nâng cao được kỹ năng sử dụng ngôn ngữ lập trình Java, kỹ năng xây dựng và phát triển chương trình. Khó khăn lớn nhất là phải tìm hiểu về thư viện Bouncy Castle và chức năng mã khóa theo từng đoạn lớn để xử lý tệp tin lớn.

Chương trình về cơ bản đã hoàn thành và chạy khá tốt, chưa thấy lỗi mới nào phát sinh trong quá trình thử nghiệm, em cũng chưa có thời gian để thử nghiệm phá khóa của chương trình. Tuy nhiên em tự nhận xét chương trình cần có những cải thiện sau. Thứ nhất là về thời gian chạy, mã nguồn chưa được trong sáng và rõ ràng dẫn đến chương trình chưa được tối ưu. Thứ hai là chức năng sử dụng khóa bất đối xứng để tăng tính bảo mật. Ngoài ra còn có thể cải biến mã nguồn để chương trình chạy trên các hệ điều hành khác như Linux, Android.

# DANH MỤC TÀI LIỆU THAM KHẢO

[1] SU, D. F., CHEN, Z. Q., & HUO, L. (2005). Communication between Java and other advance language based on JNI [J]. *Journal of Guangxi University (Natural Science Edition)*, *1*.

[2] Zadok, E., Badulescu, I., & Shender, A. (1998). *Cryptfs: A stackable vnode level encryption file system* (Vol. 184). Technical Report CUCS-021-98, Computer Science Department, Columbia University.

[3] Makwana, V., & Parmar, N. (2014). Encrypt an audio file using combine approach of transformation and cryptography. IJCSIT) International Journal of Computer Science and Information Technologies, 5(3), 4473-4476.

[4] Krawczyk, H., Bellare, M., & Canetti, R. (1997). RFC2104: HMAC: Keyed-hashing for message authentication.

[5] Krawczyk, H., Bellare, M., & Canetti, R. (1997). HMAC: Keyed-hashing for message authentication.

[6] Panda, M., & Nag, A. (2015, May). Plain text encryption using AES, DES and SALSA20 by java based bouncy castle API on Windows and Linux. In 2015 Second International Conference on Advances in Computing and Communication Engineering (pp. 541-548). IEEE.