**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**Môn: Mật Mã Và An Ninh Mạng (CO3069)**

**----------\*---------**

****

**BÁO CÁO BÀI TẬP LỚN SỐ 1**

**GVHD: Nguyễn Hữu Hiếu**

**SV:**

**Phan Thị Ngọc Ánh - 1610129**

**Nguyễn Giáp Phương Duy - 1610473**

**Nguyễn Thị Thanh Nhật - 1612390**

**MỤC LỤC**

[**1.** **Giới thiệu.** 3](#_Toc4392414)

[**2.** **Tổng quan chương trình.** 3](#_Toc4392415)

[**3.** **Tính năng cơ bản.** 3](#_Toc4392416)

[**3.1. Giải thuật bất đối xứng RSA.** 3](#_Toc4392417)

[**3.2. Giải thuật đối xứng: AES-128, AES-192, AES-256, DES, Triple DES** 4](#_Toc4392418)

[**3.2. Hash** 5](#_Toc4392419)

[**4.** **Tính năng nâng cao.** 6](#_Toc4392420)

[**5.** **Các bước thực hiện mã hóa** 6](#_Toc4392421)

[**5.1. Bất đối xứng:** 6](#_Toc4392422)

[**5.2. Đối xứng:** 8](#_Toc4392423)

[**5.3. Kiểm tra hash:** 8](#_Toc4392424)

[**6.** **Ưu điểm và nhược điểm.** 8](#_Toc4392425)

[**6.1. Ưu điểm:** 8](#_Toc4392426)

[**6.2. Nhược điểm:** 9](#_Toc4392427)

1. **Giới thiệu.**

Mã hoá là phương pháp giúp bảo vệ dữ liệu cá nhân nhạy cảm trên máy tính của bạn, cho dù bạn có gởi dữ liệu cho cá nhân, tổ chức nào đó qua mạng Internet, hay sao lưu dữ liệu cá nhân trên các máy chủ, Cloud,..., thì việc mã hoá sẽ ngăn chặn bất cứ ai có thể đọc được dữ liệu trước khi được sự cho phép của bạn. Trong assignment này chúng ta sẽ thực hiện một chương trình mã hóa để giữ cho các tập tin và thư mục trên máy tính của mình thật sự an toàn. Cụ thể là xây dựng chương trình mã hóa và giải mã các tập tin và thư mục sử dụng các giải thuật mã hóa được sử dụng phổ biến trong thực tế như DES, AES-128, AES-192, AES-256, RSA...

1. **Tổng quan chương trình.**

Chương trình mã hóa được xây dựng bằng ngôn ngữ Java, sử dụng phần mềm tích hợp NetBeans để hỗ trợ quá trình lập trình và phát triển dưới dạng một ứng dụng Desktop.

Yêu cầu cài đặt: máy tính có hỗ trợ môi trường Java.

Các định dạng file hỗ trợ: Tất cả các dạng file (khuyến khích mã hóa trên File có kích thước vừa phải, không quá lớn để quá trình mã hóa được thực hiện nhanh chóng).

Ứng dụng được tích hợp các giải thuật sau: RSA, DES, AES-128, AES-192, AES-256, Triple DES.

1. **Tính năng cơ bản.**

Ứng dụng mã hóa và giải mã file có định dạng bất kỳ.

Người dùng sử dụng ứng dụng có thể lựa chọn giải thuật theo từng mục đích khác nhau ở màn hình chính của ứng dụng. Màn hình chính của ứng dụng như sau:

Các giải thuật được tích hợp trong chương trình:

## **3.1. Giải thuật bất đối xứng RSA.**

*Quy trình thực hiện được tiến hành như sau:*

* Tạo ra một bộ key gồm 3 thông số n, e, d. Lưu lại dưới các định dạng: \*.rsk (cả 2 key), \*.publickey (khóa công khai), \*.privatekey (khóa bí mật).
* Với Public Key – khóa công khai bao gồm (n, e), Private Key – khóa bí mật bao gồm (n, d).
* Người dùng có thể import RSA key, nếu đã có sẵn vào ứng dụng để sử dụng, hoặc tạo mới một cặp key cho riêng mình. Key được tạo mới có thể lưu lại bằng cách export ra file có định dạng: \*.rsk, \*.publickey và \*.privatekey.
* Mã hóa bất kỳ định dạng file nào tạo ra ra file được mã hóa với định dạng \*.rse, sử dụng public key.
* Giải mã file mã hóa định dạng \*.rse ngược lại thành file gốc, thêm mã thời gian vào tên file để tránh trùng lặp với file cũ.

*Hiện thực RSA:*

* Tạo giá trị Key:

+ Bước 1: Khởi tạo kích thước ban đầu của N là 512 bits.

+ Bước 2: Sinh ngẫu nhiên 2 số nguyên tố lớn p và q.

+ Bước 3: Tính giá trị n = p\*q, m = (q-1)(p-1).

+ Bước 4: Chọn số e là số nguyên tố cùng nhau với m.

+ Bước 5: Từ số e được chọn ở trên ta tính được giá trị d tương ứng.

* Mã hóa file:
* Input: Khóa công khai và file input cần mã hóa.
* Output: File đã được mã hóa lưu lại dưới định dạng \*.rse.

+ Bước 1: Tính giá trị MD5 của file input ghi vào file output.

+ Bước 2: Đọc file input lần lượt từng khối dưới dạng byte có kích thước 64 bits. Mã hóa các block dữ liệu đọc được.

+ Bước 3: Ghi dữ liệu khối đã được mã hóa vào file input sau đó quay lại bước 2, đến khi file không còn dữ liệu nữa thì dừng lại.

* Giải mã file:
* Input: File input bị mã hóa \*.rse và khóa bí mật của người nhận.
* Output: File dữ liệu với nội dung gốc.

+ Bước 1: Đọc dữ liệu dạng Integer của file input để lấy kích thước của khối dữ liệu mã hóa.

+ Bước 2: Đọc tiếp dữ liệu của file input với kích thước ở trên, giải mã từng khối và ghi kết quả vào output.

+ Bước 3: Lặp lại đến khi đọc hết toàn bộ file.

+ Bước 4: Tính giá trị MD5 của file output để kiểm tra tính toàn vẹn của dữ liệu.

## **3.2. Giải thuật đối xứng: AES-128, AES-192, AES-256, DES, Triple DES**

*Quy trình thực hiện được tiến hành như sau:*

* Người dùng chọn 1 file bất kỳ trong thư viện máy tính.
* Người dùng chọn giải thuật: AES-128, AES-192, AES-256, DES, Triple DES
* Người dùng nhập chuỗi khóa bất kỳ. Nếu độ dài chuỗi khóa đúng với số byte key yêu cầu thì đưa trực tiếp key vào hàm encrypt. Ngược lại, nếu độ dài chuỗi khóa không đúng với số byte key yêu cầu thì đưa qua hàm hash SHA-256 để hash thành chuỗi có độ dài mong muốn.
* AES-128: 128 bits
* AES-192: 192 bits
* AES-256: 256 bits
* DES: 64 bits (sử dụng thực tế 56 bits)
* Triple DES: 192 bits (sử dụng thực tế 168 bits)
* Người dùng có thể nhập Init Vector (cần thiết đúng size). Nếu người dùng không nhập, thì chương trình tự sinh random init vector đúng với size theo giải thuật.
* AES-128: 128 bits
* AES-192: 128 bits
* AES-256: 128 bits
* DES: 64 bits
* Triple DES: 8 64 bits
* Mã hóa bất kỳ định dạng file nào tạo ra ra file được mã hóa với định dạng \*.encrypted.
* Giải mã file mã hóa định dạng \*.encrypted ngược lại thành file gốc, thêm mã thời gian vào tên file để tránh trùng lặp với file cũ, thêm phần đuôi theo file mã hóa trước, nhờ đó có thể open ngay tại chỗ.

*Hiện thực RSA:*

* Mã hóa file:
* Input: Khóa công khai và file input cần mã hóa.
* Output: File đã được mã hóa lưu lại dưới định dạng \*.encrypted

+ Bước 1: Đọc file input lần lượt từng khối dưới dạng byte có kích thước như sau:

* AES: 128 bits
* DES: 64 bits
* Triple DES: 64 bits

Mã hóa các block dữ liệu đọc được.

+ Bước 2: Ghi dữ liệu khối đã được mã hóa vào file input sau đó quay lại bước 1, đến khi file không còn dữ liệu nữa thì dừng lại.

* Giải mã file:
* Input: File input bị mã hóa \*.encrypted.
* Output: File dữ liệu với nội dung gốc.

+ Bước 1: Đọc dữ liệu của file input với kích thước như trên, giải mã từng khối và ghi kết quả vào output.

+ Bước 2: Lặp lại đến khi đọc hết toàn bộ file.

## **3.2. Hash**

*Quy trình thực hiện được tiến hành như sau:*

* Người dùng chọn 2 file trong thư viện máy tính, 1 file gốc và 1 file đã decrypted.
* Người dùng chọn giải thuật hash, chương trình hỗ trợ các giải thuật: MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-512
* Hash từng file và xuất ra output là một chuỗi Hex
* So sánh hai chuỗi để kiểm tra file sau khi mã hóa và giải mã có còn đảm bảo tính toàn vẹn hay không.

1. **Tính năng nâng cao.**

* Xem thời gian thực hiện các giải thuật.
* Hiện thực quá trình sinh khóa và phân phối khóa.
* Open file decrypted để việc kiểm tra thuận tiện hơn.

1. **Các bước thực hiện mã hóa**

## **5.1. Bất đối xứng:**

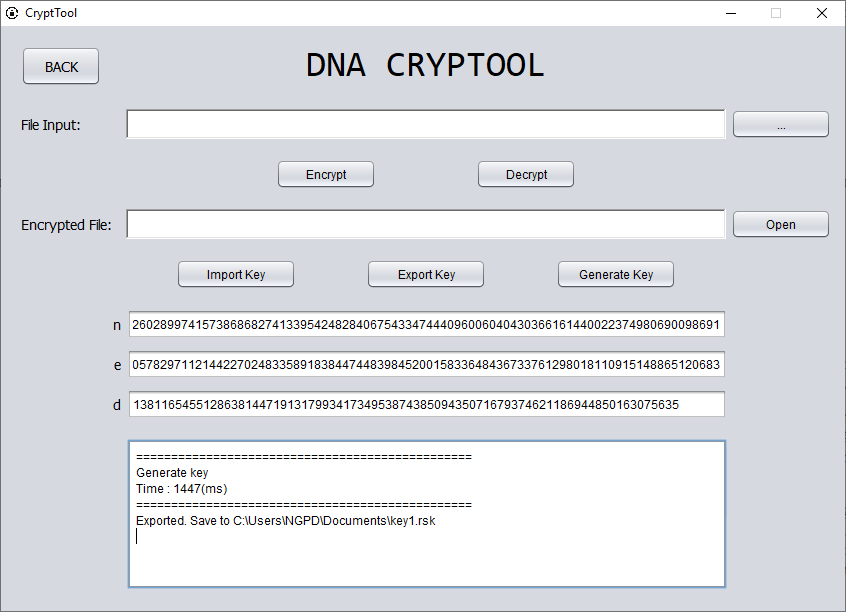


Figure 1 Giao diện của mã hóa bất đối xứng

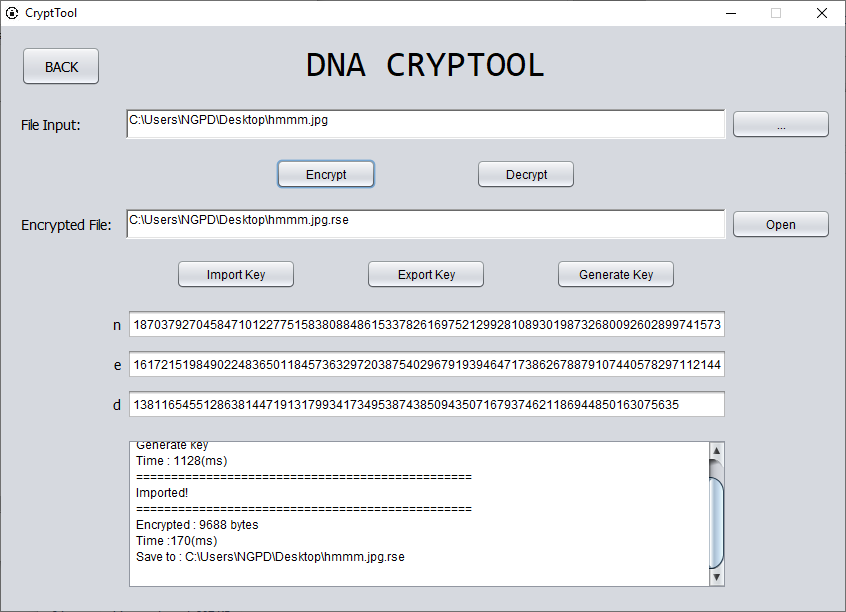


Figure 2 Mã hóa

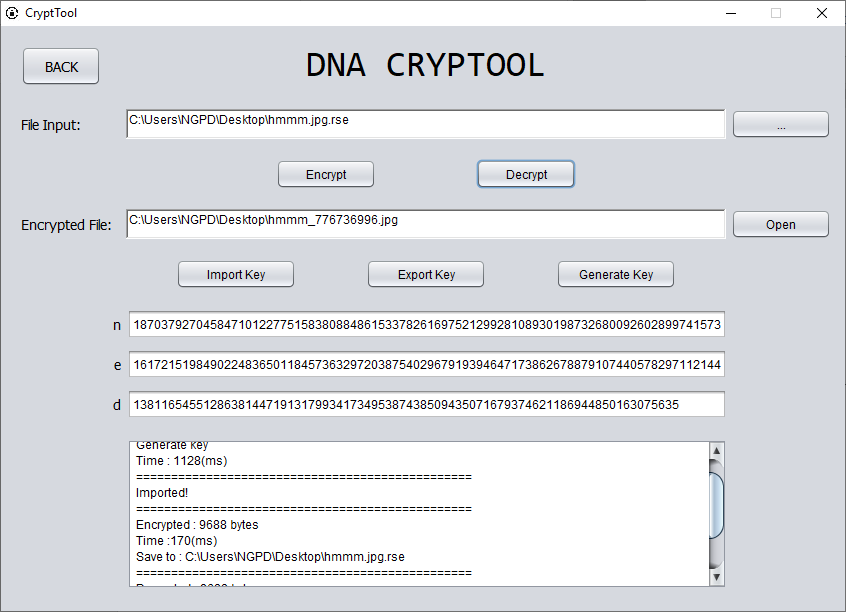
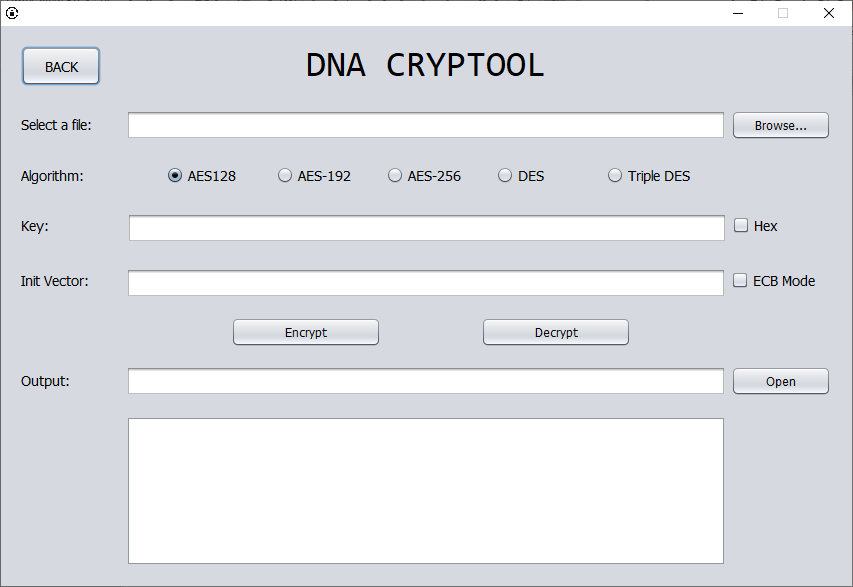


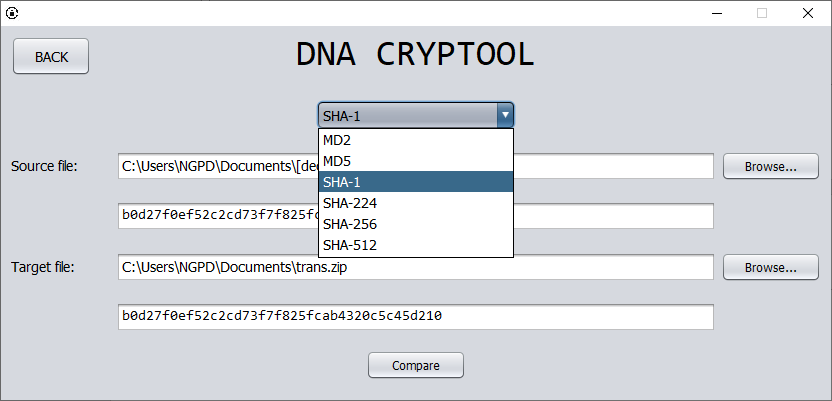
Figure 3 Giải mã

## **5.2. Đối xứng:**

Các bước thực hiện hoàn toàn tương tự



## **5.3. Kiểm tra hash:**



1. **Ưu điểm và nhược điểm.**

## **6.1. Ưu điểm:**

- Mã hóa được tất cả các file.

- Giao diện đơn giản, dễ dùng.

- Mã hóa nhanh chóng.

## **6.2. Nhược điểm:**

**-** Đối với giải thuật RSA, chỉ nên mã hóa các file có kích thước 10MB, nếu lớn hơn thì thời gian mã hóa khá lâu.

**-** Màn hình chưa được scale với nhiều kích thước.