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

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

****

**MẬT MÃ VÀ AN NINH MẠNG (CO3069)**

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

**BÁO CÁO BÀI TẬP LỚN SỐ 1**

**Nhóm DNA**

GVHD: Nguyễn Hữu Hiếu

|  |  |  |
| --- | --- | --- |
| Thành viên: | 1610129 | Phan Thị Ngọc Ánh |
|  | 1610473 | Nguyễn Giáp Phương Duy |
|  | 1612390 | Nguyễn Thị Thanh Nhật |
|  | 1720075 | Nguyễn Hoàng Hải Triều |
|  | 1820059 | Lê Thanh Sơn |

TPHCM, tháng 03/2019

**MỤC LỤC**

[1. Tóm tắt 3](#_Toc4502566)

[2. Giới thiệu và Tổng quan chương trình 3](#_Toc4502567)

[2.1 Giới thiệu mã hóa 3](#_Toc4502568)

[2.2 Tổng quan chương trình 3](#_Toc4502569)

[3. Nội dung 4](#_Toc4502570)

[3.1. Giải thuật bất đối xứng RSA. 4](#_Toc4502571)

[3.2. Giải thuật đối xứng (AES, DES, Triple DES) 5](#_Toc4502572)

[3.2. So trùng Hash 7](#_Toc4502573)

[3.3. Tính năng nâng cao. 7](#_Toc4502574)

[4. Phân tích và kết luận. 8](#_Toc4502575)

[4.1.Tổng kết kết quả: 8](#_Toc4502576)

[4.2. Ưu điểm: 8](#_Toc4502577)

[4.3. Nhược điểm: 8](#_Toc4502578)

[5. Hướng phát triển 8](#_Toc4502579)

[6. Tham khảo: 9](#_Toc4502580)

[Phụ lục 1: Bảng phân chia công việc và tiến độ hoàn thành 10](#_Toc4502581)

[Phụ lục 2: Hướng dẫn sử dụng chương trình 10](#_Toc4502582)

1. **Tóm tắt**

Bài báo cáo này mô tả chi tiết về ứng dụng mã hóa “DNA Cryptool” được thực hiện bởi nhóm DNA thuộc lớp L01, trong quá trình làm bài tập lớn môn Mật mã và An ninh mạng. Chương trình có thể mã hóa được các file với nhiều định dạng bằng các giải thuật RSA, AES (128-192-256), DES, Triple DES trên các mode CBC và ECB.

Bài báo cáo bao gồm:

* Giới thiệu tổng quan về chương trình
* Cơ sở của các giải thuật mã hóa
* Cách hiện thực các giải thuật
* Phân tích và đánh giá kết quả thực hiện
* Hướng phát triển của ứng dụng
* Hướng dẫn sử dụng ứng dụng

1. **Giới thiệu và Tổng quan chương trình**

## **2.1 Giới thiệu mã hóa**

Mã hoá là phương pháp giúp bảo vệ dữ liệu cá nhân nhạy cảm trên máy tính và mạng máy tính. Cho dù là gửi 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 cũng sẽ giúp chúng ta ngăn chặn bất cứ ai có thể đọc được dữ liệu trước khi được cho phép. Trong assignment này chúng ta sẽ hiện thực 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 (128/192/256), RSA.

## **2.2 Tổng quan chương trình**

Chương trình mã hóa được xây dựng bằng ngôn ngữ Java với Java Cryptography Architecture, sử dụng NetBeansIDE 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 định dạng file, chương trình hoạt động tốt trên File có kích thước vừa phải, đối với file có kích cỡ quá lớn thì quá trình mã hóa được thực hiện chưa thực sự hiệu quả (giải thuật RSA vốn không phù hợp với dữ liệu kích thước lớn).

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

Ngoài ra, ứng dụng còn hỗ trợ các giải thuật Hash (MD2, MD5, SHA-1, SHA-256, SHA-384, SHA-512) để kiểm tra tính toàn vẹn của file sau khi mã hóa và giải mã.

1. **Nội dung**

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:



Hình 1. Màn hình chính của ứng dụng

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, DES, Triple DES)**

*Quy trình mã hóa/giải mã được tiến hành như sau:*

* Người dùng chọn 1 file bất kỳ trong thư viện máy tính.
* Người dùng chọn 1 trong các giải thuật chương trình hỗ trợ: AES-128, AES-192, AES-256, DES, Triple DES
* Nhập khóa: để truyền chính xác khóa cho chương trình, người dùng cần nhập khóa ở dạng Hex và có độ dài x đúng với độ dài khóa yêu cầu của giải thuật tương ứng, nếu ngắn hơn hoặc dài hơn, thì chương trình sẽ hash lại chuỗi hex này theo giải thuật SHA-256 rồi lấy ra x bit đầu để làm khóa (như vậy khóa thực sự dùng để mã hóa/giải mã sẽ là giá trị hash chứ không phải là chuỗi đã nhập vào). Khi người dùng nhập khóa dạng văn bản, chương trình sẽ dùng hàm getBytes() của java với giá trị chuỗi nhập vào rồi hash tương tự. Giá trị của x ứng với từng giải thuật như sau:
* AES-128: 128 bits (16 bytes hay 32 ký tự Hex)
* AES-192: 192 bits (24 bytes hay 48 ký tự Hex)
* AES-256: 256 bits (32 bytes hay 64 ký tự Hex)
* DES: 64 bits (sử dụng thực tế 56 bits – 8 bytes hay 16 ký tự Hex)
* Triple DES: 192 bits (sử dụng thực tế 168 bits – 24 bytes hay 48 ký tự Hex)
* Nếu sử dụng mode ECB thì không cần đến Init Vector, tuy nhiên mặc định chương trình sử dụng CBC mode cho các giải thuật (do ECB có độ bảo mật kém so với CBC) và yêu cầu phải có một giá trị Init Vector. Nếu muốn nhập Init Vector theo ý muốn, người dùng cần nhập đúng kích thước ở dạng Hex của nó. Nếu người dùng không nhập thì chương trình sẽ tự sinh ngẫu nhiên nó với hàm random lấy từ class SecureRandom để tạo Init Vector đúng với size theo giải thuật. Tuy nhiên để giải mã thành công thì người dùng phải có cả khóa lẫn Init Vector, nên cần phải lưu lại giá trị được sinh ra này. Độ dài của Init Vector đối với từng giải thuật như sau:
* AES: 128 bits
* DES, Triple DES: 64 bits
* Vì hiện thực trên kiểu dữ liệu byte nên chương trình có thể mã hóa bất kỳ định dạng file nào, và kết quả tạo ra là file được mã hóa với định dạng [tên đầy đủ của file ban đầu].encrypted.
* Giải mã file đã mã hóa thành file gốc chỉ chấp nhận file có tên \*.encrypted, xóa bỏ đuôi này và thêm prefix để tránh ghi đè lên file ban đầu, mặc dù thực tế file ban đầu thường được xóa đi sau khi mã hóa. Hỗ trợ Open ngay khi giải mã xong.

*Hiện thực:*

* Mã hóa file:

INPUT: File input cần mã hóa, khóa và Init Vector (nếu dùng CBC).

OUTPUT: File đã được mã hóa lưu lại dưới định dạng \*.encrypted

+ Bước 1: Sinh ngẫu nhiên Init Vector.

+ Bước 2: Truyền Key và Init Vector vào nếu nhập đúng.

+ Bước 3: Đọc file input lần lượt từng khối dưới dạng mảng byte. Sau khi thử với nhiều kích cỡ khác nhau, nhóm chọn giá trị blocksize là 1024 bytes để có thời gian thực thi tốt nhất. Trên thực tế CipherInputStream sẽ phân chia lại blocksize này và chỉ hoạt động trên những block có kích thước như sau:

* AES: 128 bits (16 bytes)
* DES, Triple DES: 64 bits (8 bytes)

+ Bước 4: Mã hóa trên từng block dữ liệu đọc được.

+ Bước 5: Ghi dữ liệu khối đã được mã hóa vào file input sau đó quay lại bước 3, 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: Sinh ngẫu nhiên Init Vector

+ Bước 2: Truyền Key và Init Vector vào nếu nhập đúng

+ Bước 3: Đọc dữ liệu của file input với kích thước như khi mã hóa

+ Bước 4: Giải mã từng khối và dùng CipherOutputStream ghi kết quả vào output file.

+ Bước 5: Lặp lại đến khi đọc hết toàn bộ file.

## **3.2. So trùng Hash**

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

* Người dùng chọn giải thuật hash, mặc định là MD5.
* Người dùng chọn file thứ nhất trong thư viện máy tính.
* Sau khi đọc xong file, chương trình tự động Hash theo giải thuật đã chọn
* Người dùng chọn file thứ hai trong thư viện máy tính.
* Chương trình xuất ra giá trị Hash theo giải thuật đã chọn.
* So sánh hai chuỗi để kiểm tra file sau khi mã hóa và giải mã có còn đảm bảo tính toàn vẹn hay không.

## **3.3. Tính năng nâng cao.**

* Kết luận 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. **Phân tích và kết luận.**

## **4.1.Tổng kết kết quả:**

* Ứng dụng đã mã hóa thành công file với nhiều định dạng khác nhau bằng các giải thuật: RSA, DES, AES-128, AES-192, AES-256, Triple DES.
* Với một file có kích thước vừa phải (khoảng dưới 1MB), các giải thuật mã hóa đối xứng chạy tương đối nhanh trong không quá 1 giây.
* Ứng dụng có bổ sung các tính năng nâng cao, các tính năng chạy tốt và ổn định.

## **4.2. Ưu điểm:**

* Mã hóa được tất cả các file.
* Giao diện đơn giản, dễ dùng.
* Mã hóa nhanh chóng.

## **4.3. Nhược điểm:**

* Chưa mã hóa được thư mục.
* Giao diện chưa bắt mắt.
* 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.
* Chưa tự động phát hiện file sau khi giải mã có sử dụng được hay không mà phải thực hiện thủ công.

1. **Hướng phát triển**

Trong tương lai, ứng dụng sẽ có những hướng phát triển sau:

* Tích hợp lớp mã hóa vào các ứng dụng có sẵn như ứng dụng chat, file sharing để bảo mật dữ liệu gởi nhận.
* Có khả năng mã hóa toàn bộ tập tin trong thư mục được chọn.
* Hiển thị trạng thái mã hóa cụ thể hơn.
* Tự động dò và xác nhận file.

1. **Tham khảo:**
2. Cryptography Stack Exchange
3. StackOverflow
4. Javadoc
5. Java AES Encryption Decryption Example (26/03/2019): <https://howtodoinjava.com/security/java-aes-encryption-example/>
6. RSA (mã hóa), Wikipedia (26/03/2019): [*https://vi.wikipedia.org/wiki/RSA\_(m%C3%A3\_h%C3%B3a)*](https://vi.wikipedia.org/wiki/RSA_(m%C3%A3_h%C3%B3a))
7. Triple DES, Wikipedia (26/03/2019): [*https://en.wikipedia.org/wiki/Triple\_DES*](https://en.wikipedia.org/wiki/Triple_DES)
8. Data Encryption Standard, Wikipedia (26/03/2019): [*https://en.wikipedia.org/wiki/Data\_Encryption\_Standard*](https://en.wikipedia.org/wiki/Data_Encryption_Standard)
9. Advanced Encryption Standard, Wikipedia (26/03/2019): [*https://en.wikipedia.org/wiki/Advanced\_Encryption\_Standard*](https://en.wikipedia.org/wiki/Advanced_Encryption_Standard)
10. Github Gist (26/03/2019): [*https://gist.github.com/dmydlarz/32c58f537bb7e0ab9ebf*](https://gist.github.com/dmydlarz/32c58f537bb7e0ab9ebf)
11. Java - Mã hóa và giải mã với thuật toán RSA (26/03/2019): [*https://viblo.asia/p/java-ma-hoa-va-giai-ma-voi-thuat-toan-rsa-bJzKmW3Xl9N*](https://viblo.asia/p/java-ma-hoa-va-giai-ma-voi-thuat-toan-rsa-bJzKmW3Xl9N)

**Phụ lục 1: Bảng phân chia công việc và tiến độ hoàn thành**

|  |  |  |  |
| --- | --- | --- | --- |
| **MSSV** | **Họ và tên** | **Nhiệm vụ** | **Hoàn thành** |
| 1610129 | Phan Thị Ngọc Ánh | - Hiện thực giải thuật RSA, sinh khóa, giao diện.  - Soạn thảo báo cáo. | 100% |
| 1610473 | Nguyễn Giáp Phương Duy | - So trùng hash.  - Hiệu chỉnh giải thuật mã hóa đối xứng để có thể mở rộng thêm giải thuật.  - Hoàn thiện báo cáo. | 100% |
| 1612390 | Nguyễn Thị Thanh Nhật | - Hiện thực giải thuật mã hóa đối xứng AES, DES. Giao diện  - Soạn thảo báo cáo. | 100% |

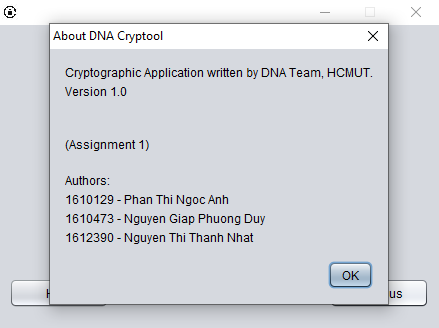
**Phụ lục 2: Hướng dẫn sử dụng chương trình**

Giao diện chính của chương trình



Hình 2. Giao diện chính của chương trình

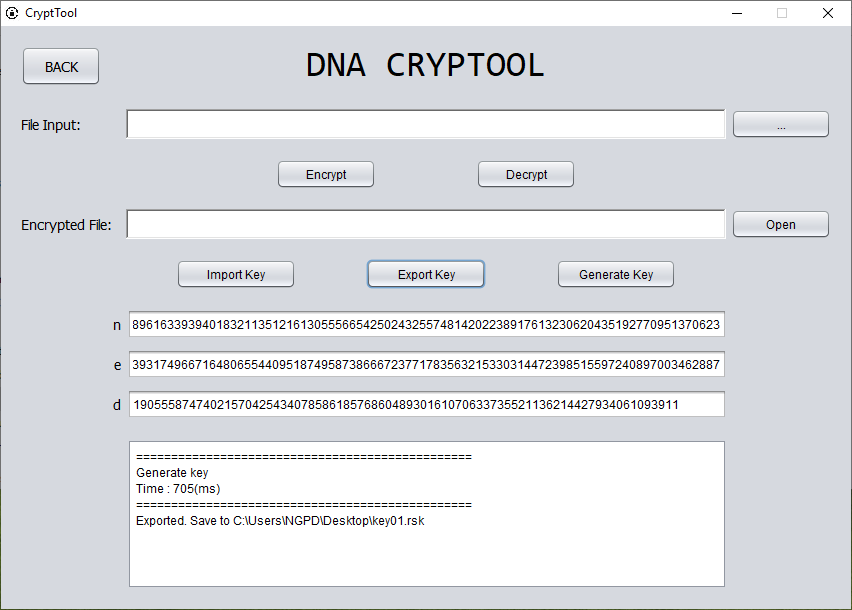
Thông tin về nhóm



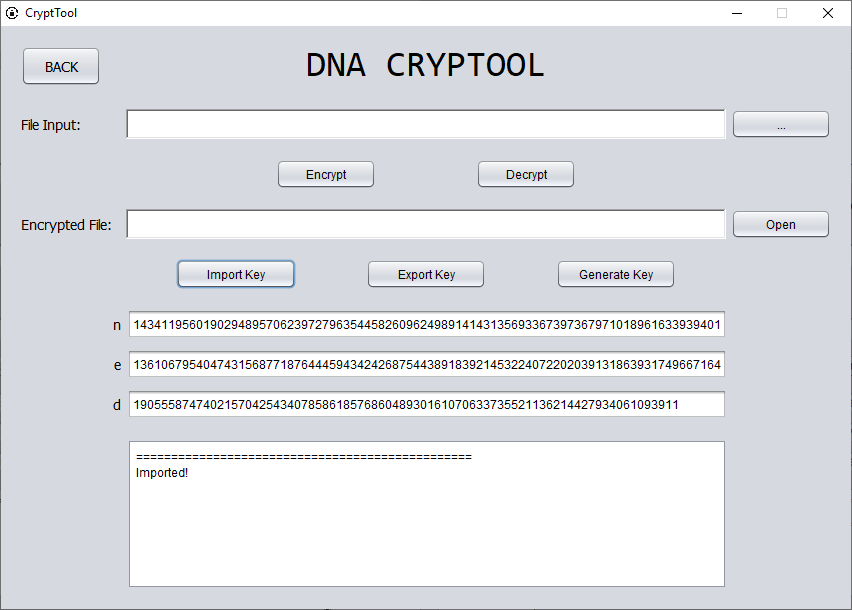
Hình 3 Thông tin về nhóm

***Để sử dụng giải thuật mã hóa bất đối xứng – RSA, chọn Asymmetric***

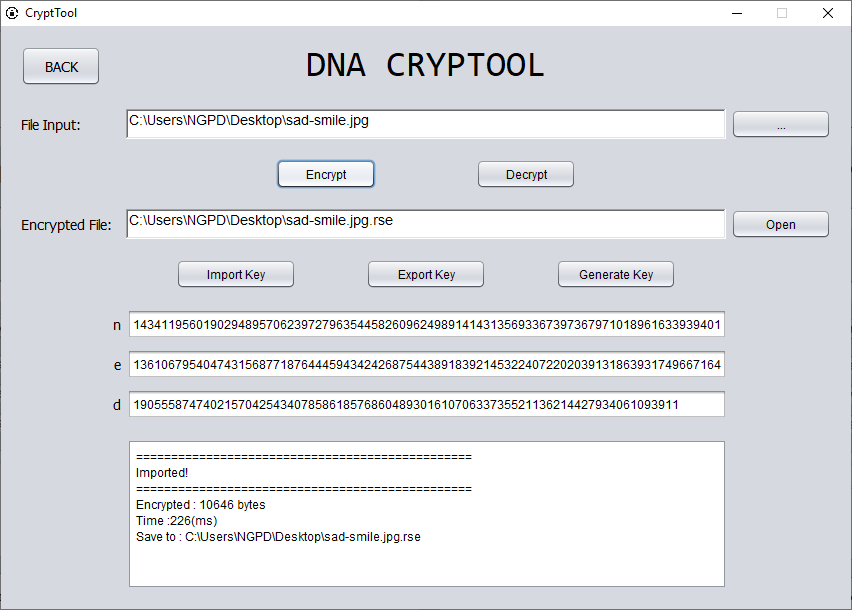
1. Khi chưa có khóa, người dùng cần tạo hoặc nhập khóa. Để tạo khóa chọn Generate Key, rồi chọn Export Key để lưu lại khóa, tiếp tục chọn đường dẫn, tên file lưu khóa và nhập mật khẩu để bảo mật khóa.



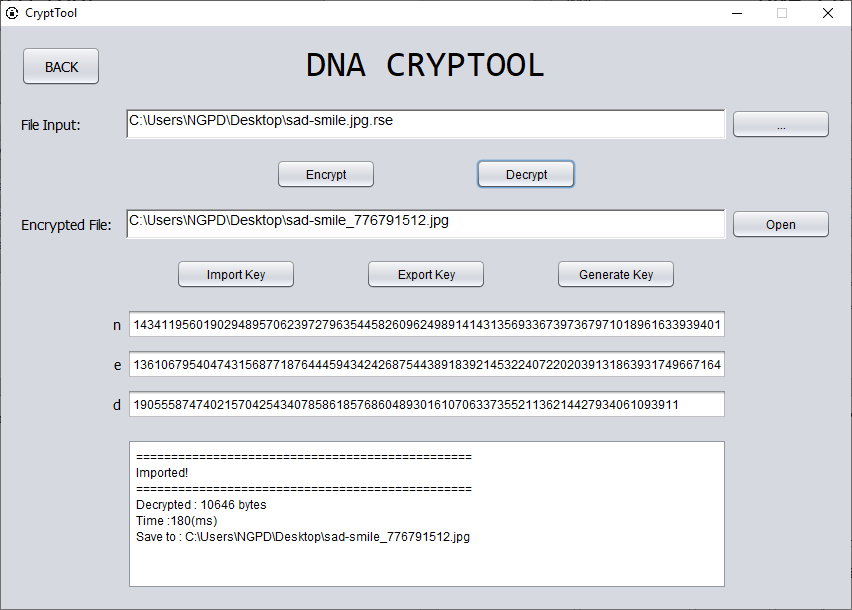
1. Khi đã có khóa từ trước đó thì chọn Import Key để nhập khóa



1. Chọn “…” để đưa đường dẫn đến file cần mã hóa/giải mã vào File input
2. Chọn Encrypt hoặc Decrypt



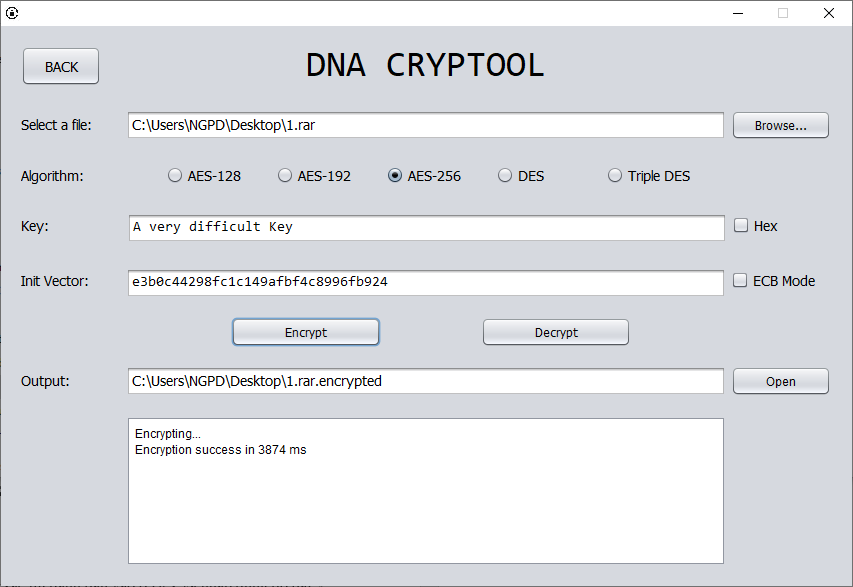
Minh họa cho quá trình Encrypt một tệp hình ảnh



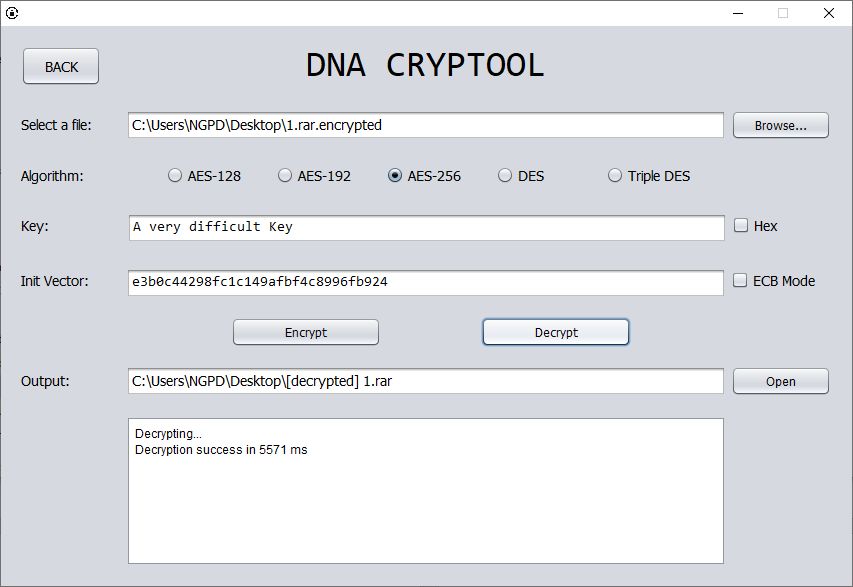
Quá trình giải mã cho chính tập tin trên. Sau khi giải mã có thể Open ngay

***Để sử dụng giải thuật mã hóa đối xứng, chọn Symmetric***

1. Chọn Browse…, tìm đến file cần mã hóa/giải mã, nhấn Open
2. Chọn giải thuật mã hóa/giải mã ở Algorithm
3. Nhập khóa ở Key. Nếu muốn nhập khóa chính xác thì đánh dấu vào ô Hex và nhập đúng độ dài.
4. Tick vào ECB nếu không muốn sử dụng Init Vector, tuy nhiên chế độ này được đánh giá là kém bảo mật hơn.
5. Nhập Init Vector. Bắt buộc phải nhập ở dạng Hex. Nếu không nhập phần mềm sẽ tự tạo ngẫu nhiên.
6. Nhấn Encrypt/Decrypt.
7. Sau khi thực hiện thành công, đường dẫn tới file output được hiển thị, có thể nhấn Open để xem nội dung (nếu đang giải mã).



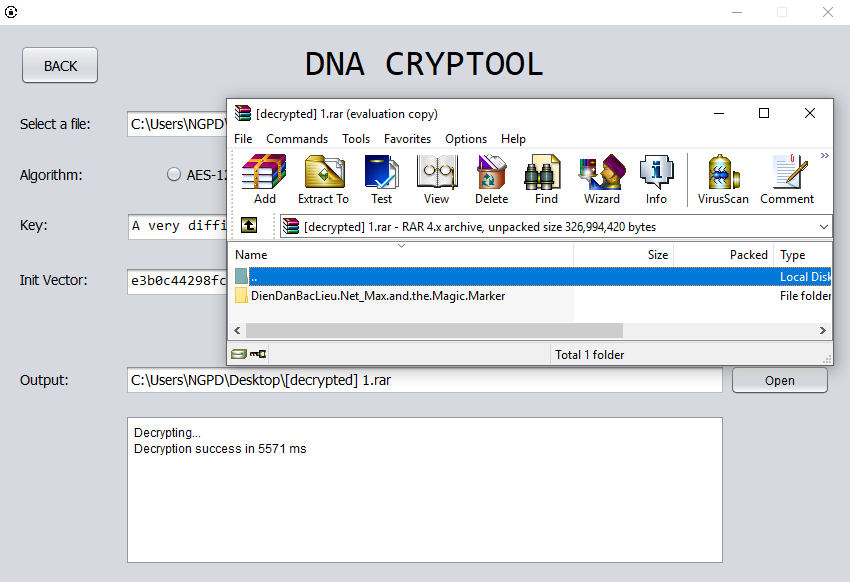
Minh họa cho quá trình mã hóa dùng giải thuật AES-256, tập tin 1.rar có kích thước là 137MB



Quá trình giải mã chính tập tin vừa mã hóa nêu trên.

File này có thể được mở thành công ngay sau khi giải mã

Lưu ý rằng nếu chọn sai thông tin lúc giải mã thì quá trình giải mã vẫn tiến hành nhưng tệp không thể mở vì nội dung của nó không còn đúng định dạng nữa.

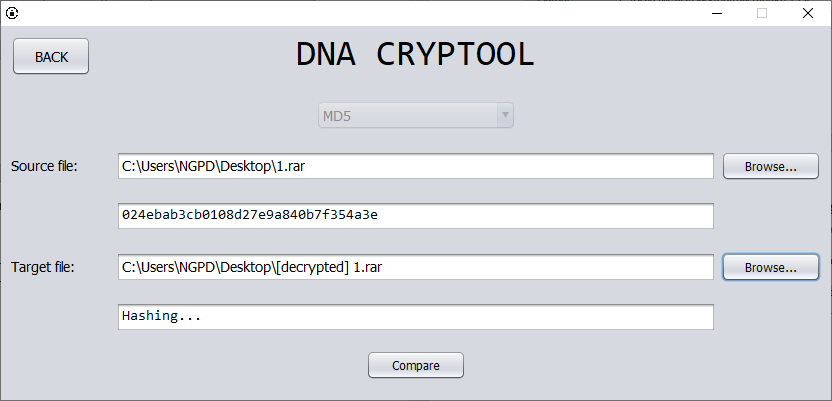


***Để kiểm tra tính toàn vẹn, chọn Hash***

Chọn Browse... để xác định đường dẫn đến file gốc ở ô Source file (chương trình tự động tính giá trị Hash và hiển thị ở ngay bên dưới đường dẫn)

Chọn Browse... để xác định đường dẫn đến file cần kiểm tra ở ô Source file

Lúc này các giá trị Hash đều đã hiển thị và có thể kiểm tra ngay bằng mắt, nhưng nếu nó quá dài có thể nhấn Compare để biết chính xác 2 file có giống nhau hay không.



Chương trình tự tính Hash ngay khi chọn file hoặc đổi giải thuật Hash, tuy nhiên với tập tin lớn thì để tránh xung đột người dùng cần đợi quá trình hash hoàn tất thì mới được chọn giải thuật khác.

Kết quả kiểm tra Hash:

