ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA KHOA HỌC & KĨ THUẬT MÁY TÍNH

MẬT MÃ & AN NINH MẠNG

BÀI TẬP LỚN 1

GVHD: NGUYỄN ĐỨC THÁI

NGUYỄN NHẬT NAM

Danh sách nhóm:

ĐẶNG XUÂN ÁNH 51200135

ĐĂNG MINH VIỆT 51204488

LÊ TUẤN VŨ 51204609

I. MỤC LỤC

PHẦN 1: TÓM TẮT

Mã hóa với mục đích tạo một lớp bảo vệ cho dữ liệu để người khác đọc được, ngoại trừ những ai được cho chép đọc. Mã hóa sử dụng thuật toán và khóa để biến đổi dữ liệu từ hình thức đơn giản rõ ràng (plain hay cleartext), làm biến dữ liệu sang hình thức mật mã (code hay ciphertext). Chỉ có những ai có thông tin giải mã thì mới giải mã được và đọc được dữ liệu. Chúng ta biết rằng các dịch vụ lưu trữ đám mây phổ biến hiện nay (như DropBox hay SugarSync đã mã hóa dữ liệu của chúng ta, bảo vệ dữ liệu khi truyền và trong khi được lưu trên các máy chủ của họ. Tuy nhiên, cũng chính những dịch vụ đó nắm giữ chìa khóa giải mã, có nghĩa là họ có thể giải mã các tập tin của bạn mà bạn không hề hay biết. Nếu bạn có bất kỳ tập tin thực sự nhạy cảm nào trong dịch vụ lưu trữ đám mây của mình, hãy sử dụng lớp mã hóa thứ hai để giữ cho chúng an toàn khỏi những con mắt tò mò.

Trong assignment này chúng tôi sẽ thực hiện lớp mã hóa thứ hai để giữ cho các tập tin và thư mục trong quá trình truyền gửi hay lưu trữ được 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 như DES, RSA, AES, MD5.

PHẦN 2: GIỚI THIỆU

- Chương trình tích hợp ba giải thuật mã hóa, trong đó bao gồm giải thuật mã hóa đối xứng DES, giải thuật mã hóa bất đối xứng RSA và giải thuật mã hóa AES.
- Chương trình hỗ trợ mã hóa một tập tin với định dạng bất kì như: hình ảnh, âm thanh, doc, pdf...
- Quá trình mã hóa: nhận input là tập tin bất kì và tập tin text chứa chìa khóa mã hóa (encryption key) và một số option khác (nếu cần), output là tập tin hay thư mục chứa dữ liệu đã được mã hóa.
- Quá trình giải mã: nhận input là tập tin hay thư mục chứa dữ liệu đã được mã hóa và tập tin text chứa chìa khóa giải mã (decryption key) và một số option khác (nếu cần), output chương trình là tập tin được giãi mã thành công.
- Sử dụng hàm hash MD5 để chứng minh tính toàn vẹn giữa tập tin gốc ban đầu được chọn và tập tin output của quá trình giải mã.
- Chương trình sử dụng ngôn ngữ: **C#**.
- Thư viện mã hóa : **System.Security.Cryptography**;
- Chương trình chưa thực hiện được các tính năng nâng cao như file BTL yêu cầu.

PHẦN 3: THÂN BÀI

3.1 CƠ SỞ LÝ THUYẾT:

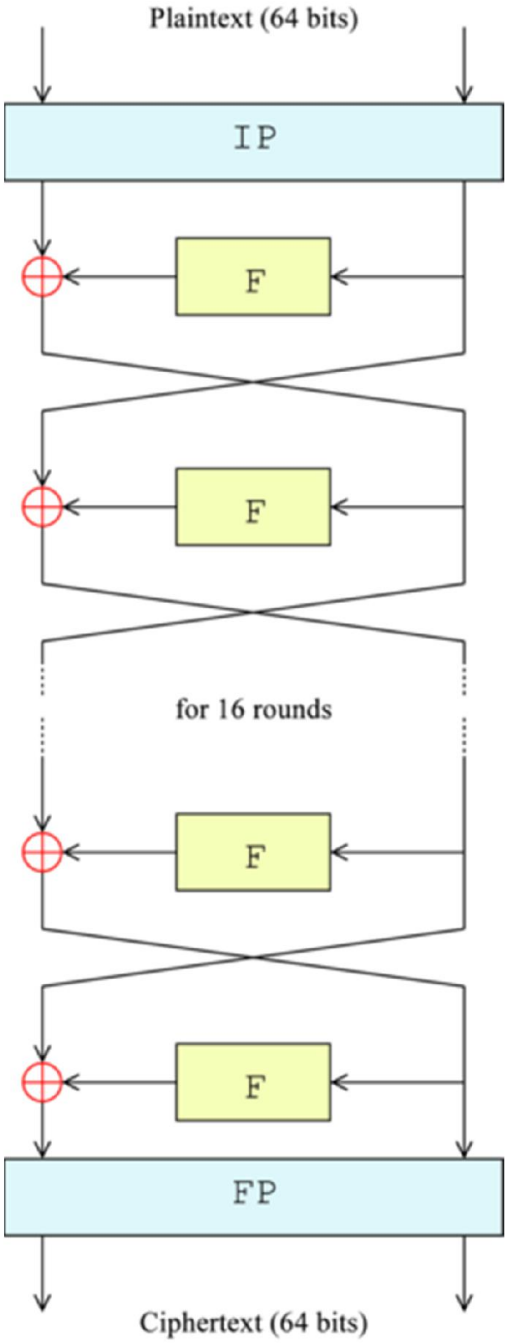
3.1.1 GIẢI THUẬT MÃ HÓA ĐỐI XỨNG DES:

3.1.1.1 Giới thiệu về DES :

DES (viết tắt của Data Encryption Standard, hay Tiêu chuẩn Mã hóa Dữ liệu) là một phương pháp mật mã hóa được FIPS (Tiêu chuẩn Xử lý Thông tin Liên bang Hoa Kỳ) chọn làm chuẩn chính thức vào năm 1976. Sau đó chuẩn này được sử dụng rộng rãi trên phạm vi thế giới. Ngay từ đầu, thuật toán

của nó đã gây ra rất nhiều tranh cãi, do nó bao gồm các thành phần thiết kế mật, độ dài khóa tương đối ngắn, và các nghi ngờ về cửa sau để Cơ quan An ninh quốc gia Hoa Kỳ (NSA) có thể bẻ khóa. Do đó, DES đã được giới nghiên cứu xem xét rất kỹ lưỡng, việc này đã thúc đẩy hiểu biết hiện đại về mật mã khối (*block cipher*) và các phương pháp thám mã tương ứng.

3.1.1.2 Mô tả thuật toán:



Hình 1 – Cấu trúc thuật toán Feistel dùng trong DES

DES là thuật toán mã hóa khối: nó xử lý từng khối thông tin của bản rõ có độ dài xác định và biến đổi theo những quá trình phức tạp để trở thành khối thông tin của bản mã có độ dài không thay đổi. Trong trường hợp của DES, độ dài mỗi khối là 64 bit. DES cũng sử dụng khóa để cá biệt hóa quá trình chuyển đổi. Nhờ vậy, chỉ khi biết khóa mới có thể giải mã được văn bản mã. Khóa dùng trong DES có độ dài toàn bộ là 64 bit. Tuy nhiên chỉ có 56 bit thực sự được sử dụng; 8 bit còn lại chỉ dùng cho việc kiểm tra. Vì thế, độ dài thực tế của khóa chỉ là 56 bit.

Giống như các thuật toán mã hóa khối khác, khi áp dụng cho các văn bản dài hơn 64 bit, DES phải được dùng theo một phương pháp nào đó. Trong tài liệu FIPS-81 đã chỉ ra một số phương pháp, trong đó có một phương pháp dùng cho quá trình nhận thực. Một số thông tin thêm về những cách sử dụng DES được miêu tả trong tài liệu FIPS-74 .

❖ Tổng thể :

Cấu trúc tổng thể của thuật toán được thể hiện ở Hình 1: có 16 chu trình giống nhau trong quá trình xử lý. Ngoài ra còn có hai lần hoán vị đầu và cuối (*Initial and final permutation* - **IP** & **EP**). Hai quá trình này có tính chất đối nhau (Trong quá trình mã hóa thì IP trước EP, khi giải mã thì ngược lại). IP và EP không có vai trò xét về mật mã học và việc sử dụng chúng chỉ có ý nghĩa đáp ứng cho quá trình đưa thông tin vào và lấy thông tin ra từ các khối phần cứng có từ thập niên 1970. Trước khi đi vào 16 chu trình chính, khối thông tin 64 bit được tách làm hai phần 32 bit và mỗi phần sẽ được xử lý tuần tự (quá trình này còn được gọi là mạng Feistel).

Cấu trúc của thuật toán (mạng Feistel) đảm bảo rằng quá trình mã hóa và giải mã diễn ra tương tự. Điểm khác nhau chỉ ở chỗ các khóa con được sử dụng theo trình tự ngược nhau. Điều này giúp cho việc thực hiện thuật toán trở nên đơn giản, đặc biệt là khi thực hiện bằng phần cứng.

Ký hiệu sau: $\oplus$ thể hiện phép toán XOR. Hàm F làm biến đổi một nửa của khối đang xử lý với một khóa con. Đầu ra sau hàm F được kết hợp với nửa còn lại của khối và hai phần được tráo đổi để xử lý trong chu trình kế tiếp. Sau chu trình cuối cùng thì 2 nửa không bị tráo đổi; đây là đặc điểm của cấu trúc Feistel khiến cho quá trình mã hóa và giải mã trở nên giống nhau.

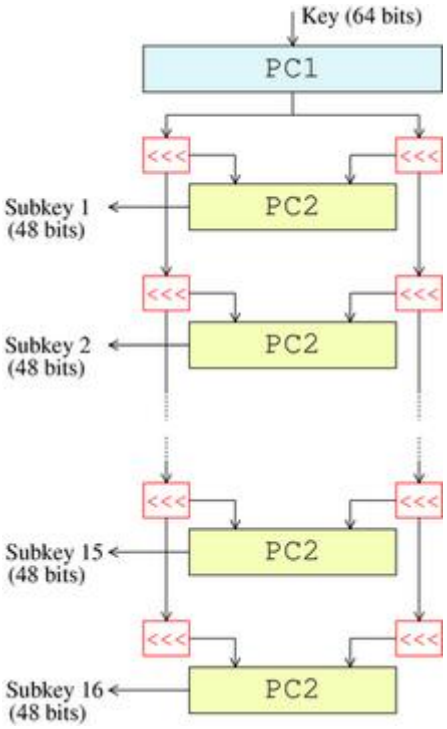
❖ Hàm Feistel (F) :

Hàm F, hoạt động trên khối 32 bit và bao gồm bốn giai đoạn:

- *Mở rộng*: 32 bit đầu vào được mở rộng thành 48 bit sử dụng thuật toán hoán vị mở rộng (*expansion permutation*) với việc nhân đôi một số bit. Giai đoạn này được ký hiệu là E trong sơ đồ.
- *Trộn khóa*: 48 bit thu được sau quá trình mở rộng được XOR với khóa con. Mười sáu khóa con 48 bit được tạo ra từ khóa chính 56 bit theo một chu trình tạo khóa con (*key schedule*) miêu tả ở phần sau.
- *Thay thế*: 48 bit sau khi trộn được chia làm 8 khối con 6 bit và được xử lý qua hộp thay thế S-box. Đầu ra của mỗi khối 6 bit là một khối 4 bit theo một chuyển đổi phi tuyến được thực hiện bằng một bảng tra. Khối S-box đảm bảo phần quan trọng cho độ an toàn của DES. Nếu không có S-box thì quá trình sẽ là tuyến tính và việc thám mã sẽ rất đơn giản.
- *Hoán vị*: Cuối cùng, 32 bit thu được sau S-box sẽ được sắp xếp lại theo một thứ tự cho trước (còn gọi là P-box).

Quá trình luân phiên sử dụng S-box và sự hoán vị các bít cũng như quá trình mở rộng đã thực hiện được tính chất gọi là sự xáo trộn và khuyếch tán (*confusion and diffusion*). Đây là yêu cầu cần có của một thuật toán mã hoá được Claude Shannon phát hiện trong những năm 1940.

❖ Quá trình tạo khóa con :



Hình 2 – Quá trình tạo khóa con trong DES.

Hình 2 mô tả thuật toán tạo khóa con cho các chu trình. Đầu tiên, từ 64 bit ban đầu của khóa, 56 bit được chọn (*Permuted Choice 1*, hay PC-1); 8 bit còn lại bị loại bỏ. 56 bit thu được được chia làm hai phần bằng nhau, mỗi phần được xử lý độc lập. Sau mỗi chu trình, mỗi phần được dịch đi 1 hoặc 2 bit (tùy thuộc từng chu trình). Các khóa con 48 bit được tạo thành bởi thuật toán lựa chọn 2 (*Permuted Choice 2*, hay PC-2) gồm 24 bit từ mỗi phần. Quá trình dịch bit (được ký hiệu là "<<<" trong sơ đồ) khiến cho các khóa con sử dụng các bit khác nhau của khóa chính; mỗi bit được sử dụng trung bình ở 14 trong tổng số 16 khóa con.

Quá trình tạo khóa con khi thực hiện giải mã cũng diễn ra tương tự nhưng các khóa con được tạo theo thứ tự ngược lại. Ngoài ra sau mỗi chu trình, khóa sẽ được dịch phải thay vì dịch trái như khi mã hóa

3.1.2 GIẢI THUẬT MÃ HÓA BẤT ĐỐI XỨNG RSA:

Trong mật mã học , RSA là một thuật toán mã hóa khóa công khai .Đây là thuật toán đầu tiên phù hợp với việc tạo ra chữ ký điện tử đồng thời với việc mã hóa .Nó đánh dấu một sự tiến bộ vượt bậc của lĩnh vực mật mã học trong việc sử dụng khóa công cộng .RSA đang được sử dụng phổ biến trong thương mại điện tử và được cho là đảm bảo an toàn với diiều kiện độ dài khóa đủ lớn .

3.1.2.1 Giới thiệu về RSA :

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

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

Thuật toán RSA được MIT đăng ký bằng sáng chế tại Hoa Kỳ vào năm 1983.Tuy nhiên , do thuật toán đã được công bố trước khi có đăng ký bảo hộ nên sự bảo hộ hầu như không có giá trị bên ngoài Hoa Kỳ .Ngoài ra , nếu như công trình của Clifford Cocks đã công bố trước đó thì bằng sáng chế RSA đã không thể được đăng ký .

3.1.2.2 Nguyên tắc hoạt động :

❖ **Mô tả sơ lược :**

Thuật toán RSA có hai khóa :khóa công khai (hau khóa công cộng) và khóa bí mật (hay khóa cá nhân) . Mỗi khóa là những khoá cố định sử dụng trong quá trình mã hóa và giải mã . Khóa công khai được công bố rộng rãi cho mọi người và được dùng để mã hóa. Những thông tin được mã hóa bằng khóa công khai chỉ có thể được giải mã bằng khóa bí mật tương ứng. Nói cách khác, mọi người đều có thể mã hóa nhưng chỉ có người biết khóa cá nhân (bí mật) mới có thể giải mã được.

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

❖ **Tạo khóa :**

Giả sử Alice và Bob cần trao đổi thông tin bí mật thông qua một kênh không an toàn (ví dụ như Internet). Với thuật toán RSA, Alice đầu tiên cần tạo ra cho mình cặp khóa gồm khóa công khai và khóa bí mật theo các bước sau:

- Chọn 2 số nguyên tố lớn p và q với $p \neq q$, lựa chọn ngẫu nhiên và độc lập.
- Tính: $n = pq$.
- Tính: giá trị hàm số Ơle $\phi(n) = (p-1)(q-1)$.
- Chọn một số tự nhiên *e* sao cho $1 < e < \phi(n)$ và là số nguyên tố cùng nhau với $\phi(n)$.
- Tính: *d* sao cho $de \equiv 1 \pmod{\phi(n)}$.

- **Một số lưu ý**:

-Các số nguyên tố thường được chọn bằng phương pháp thử xác suất.

-Các bước 4 và 5 có thể được thực hiện bằng giải thuật Euclid mở rộng

-Bước 5 có thể viết cách khác: Tìm số tự nhiên x sao cho $d = \frac{x(p-1)(q-1)+1}{e}$ cũng là số tự nhiên.

Khi đó sử dụng giá trị $d \mod (p-1)(q-1)$.

-Từ bước 3, sử dụng $\lambda = LCM(p-1, q-1)$ thay cho $\phi = (p-1)(q-1)$.

- **Khóa công khai bao gồm**:

- n, môđun.
- e, số mũ công khai (cũng gọi là số mã hóa*)*.

- **Khóa bí mật bao gồm:**

- n, môđun, xuất hiện cả trong khóa công khai và khóa bí mật,.
- d, số mũ bí mật (cũng gọi là số mũ giải mã).

❖ **Mã hóa :**

Giả sử Bob muốn gửi đoạn thông tin *M* cho Alice. Đầu tiên Bob chuyển *M* thành một số $m < n$ theo một hàm có thể đảo ngược (từ *m* có thể xác định lại *M*) được thỏa thuận trước.

Lúc này Bob có *m* và biết *n* cũng như *e* do Alice gửi. Bob sẽ tính *c* là bản mã hóa của *m* theo công thức:

$$c = m^e \mod n$$

Hàm trên có thể tính dễ dàng sử dụng phương pháp tính hàm mũ (theo môđun) bằng (thuật toán bình phương và nhân) Cuối cùng Bob gửi *c* cho Alice.

❖ **Giải mã :**

Alice nhận *c* từ Bob và biết khóa bí mật *d*. Alice có thể tìm được *m* từ *c* theo công thức sau:

$$m = c^d \mod n$$

Biết *m*, Alice tìm lại *M* theo phương pháp đã thỏa thuận trước. Quá trình giải mã hoạt động vì ta có:

$$c^d \equiv (m^e)^d \equiv m^{ed} \pmod{n}$$

Do $ed \equiv 1 \pmod{p-1}$ và $ed \equiv 1 \pmod{q-1}$, (theo Định lý Fermat nhỏ) nên:

$$m^{ed} \equiv m \pmod{p}$$

và

$$m^{ed} \equiv m \pmod{q}$$

Do *p* và *q* là hai số nguyên tố cùng nhau, áp dụng định lý số dư Trung Quốc, ta có:

$$m^{ed} \equiv m \pmod{pq}$$

hay:

$$c^d \equiv m \pmod{n}$$

❖ **Ví dụ :**

Sau đây là một ví dụ với những số cụ thể. Ở đây chúng ta sử dụng những số nhỏ để tiện tính toán còn trong thực tế phải dùng các số có giá trị đủ lớn.
Lấy :

p = 61	— số nguyên tố thứ nhất (giữ bí mật hoặc hủy sau khi tạo khóa)
q = 53	— số nguyên tố thứ hai (giữ bí mật hoặc hủy sau khi tạo khóa)
n = pq = 3233	— môđun (công bố công khai)
e = 17	— số mũ công khai
d = 2753	— số mũ bí mật

- Khóa công khai là cặp (e, n). Khóa bí mật là (d,e).
- Hàm mã hóa là: encrypt(m) = m^e mod n = m^{17} mod 3233 với m là văn bản rõ .
- Hàm giải mã là: decrypt(c) = c^d mod n = c^{2753} mod 3233 với c là văn bản mã .
- Để mã hóa văn bản có giá trị 123, ta thực hiện phép tính:

encrypt(123) = 123^{17} mod 3233 = 855

- Để giải mã văn bản có giá trị 855, ta thực hiện phép tính:

decrypt(855) = 855^{2753} mod 3233 = 123

Cả hai phép tính trên đều có thể được thực hiện hiệu quả nhờ giải thuật bình phương và nhân .

3.1.3 GIẢI THUẬT MÃ HÓA AES:

3.1.3.1 Giới thiệu về AES:

Tiêu chuẩn Advanced Encryption Standard(AES)-tiêu chuẩn mã hóa tiên tiến- là một thuật toán tiêu chuẩn của chính phủ Hoa kỳ nhằm mã hóa và giải mã dữ liệu do Viện Tiêu chuẩn và Công nghệ quốc gia Hoa Kỳ phát hành ngày 26/11/2001.

AES là một thuật toán " mã hóa khối "(block cipher) ban đầu được tạo ra bởi hai nhà mật mã học người Bỉ là Joan Daemen và Vincent Rijmen. Kể từ khi được công bố là một tiêu chuẩn, AES trở thành một trong những thuật toán phổ biến nhất sử dụng khóa mã đối xứng để mã hóa và giải mã.

3.1.3.1 Mô tả thuật toán:

AES là một thuật toán mã hóa khối đối xứng với độ dài là 128 bít, 192 bít, 256 bít tương ứng gọi là AES-128, AES-192, AES-256. AES-128 sử dụng 10 vòng, AES-192 sử dụng 12 vòng và AES-256 sử dụng 14 vòng.

- **Phép mã hóa :**

Tại thời điểm bắt đầu phép mã hóa, đầu vào được sao chép vào mảng trạng thái sử dụng các quy ước. Sau phép cộng khóa vòng khởi đầu, mảng trạng thái được biến đổi bằng cách thực hiện một hàm vòng liên tiếp với số vòng lặp là 10,12 hoặc 14 (tương ứng với độ dài khóa) vòng cuối cùng khác biệt không đáng kể với các vòng đầu tiên. Trạng thái cuối cùng được chuyển thành đầu ra.Hàm vòng được tham số hóa bằng cách sử dụng một lược đồ khóa- mảng một chiều chứa các từ 4 byte nhận từ phép mở rộng khóa.

Phép biến đổi cụ thể gồm SubBytes(), ShiftRows(), MixColums() và AddRoundkey() dùng để xử lý trạng thái.

- **SubBytes()**:

Phép biến đổi dùng trong phép mã hóa áp dụng lên trạng thái (kết quả mã hóa trung gian, được mô tả dưới dạng một mảng chữ nhật của các byte) sử dụng một bảng thay thế byte phi tuyến (Hộp S- bảng thay thế phi tuyến được sử dụng trong một phép thay thế byte và trong quy trình mở rộng khóa, nhằm thực hiện một phép thay thê 1-1 đối với giá trị mỗi byte) trên mỗi byte trạng thái một cách độc lập.

- **ShiftRows():**

Phép biến đổi dùng trong phép mã hóa áp dụng thực hiện bằng cách chuyển dịch các vòng ba hàng cuối của trạng thái theo số lượng byte các ofset khác nhau.

- **MixColumns():**

Phép biến đổi trong phép mã hóa thực hiện bằng cách lấy tất cả các cột trạng thái trộn với dữ liệu của chúng (một cách độc lập nhau) để tạo ra các cột mới.

- **AddRoundKey():**

Phép biến đổi dùng trong phép mã hóa và giải mã. Trong đó, một khóa vòng (các giá trị sinh ra từ khóa mã bằng quy trình mở rộng khóa) được thêm vào trạng thái bằng phép toán XOR. Độ dài của khóa vòng bằng độ dài của trạng thái.

❖ **Mở rộng khóa :**

Thuật toán AES nhận vào một mã K và thực hiện phép mở rộng khóa để tạo ra một lược đồ khóa. Phép mở rộng khóa tạo ra tổng số Nb(Nr+1) từ(với Nb độ dài khối và Nr số vòng). Thuật toán yêu cầu một tập khởi tạo gồm Nb từ và mỗi trong số nr vòng đòi hỏi Nb từ làm dữ liệu khóa đầu vào. Lược đồ khóa kết quả là một mảng tuyến tính các từ 4 byte.

❖ **Phép giải mã :**

Các phép biến đổi trong phép mã hóa có thể được đảo ngược và sau đó thực hiện theo chiều ngược lại nhằm tạo ra phép giải mã trực tiếp của thuật toán AES. Các phép biến đổi sử dụng trong phép giải mã gồm: InvSubButytes(), InvShiftRows(), InvMixColumns() và AddRoundKey().

- **InvSubBytes():** Là nghịch đảo của phép SubBytes, trong đó sử dụng một hộp-S nghịch đảo áp dụng cho mỗi byte của trạng thái.

- **InvShiftRows():** Là phép biến đổi ngược của ShiftRows(). Các byte trong ba từ cuối của trạng thái được dịch vòng theo số byte khác nhau. Ở hàng đầu tiên (r=0) không thực hiện phép dịch chuyển, ba hàng dưới cùng được dịch vòng Nb-shift(r,Nb) byte.

- **InvMixColumns():** Là phép ngược của MixColumns(). Nó thap tác theo từng cột của trạng thái, xem mỗi cột như một đa thức bốn hạng tử.

- **AddRoundkey():** Như phép giải mã.

3.1.4 Hàm hash MD5:

3.1.4.1 Giới thiệu về MD5 :

MD5 (Message-Digest algorithm 5) là một hàm băm để mã hóa với giá trị băm là 128bit. Từng được xem là một chuẩn trên Internet, MD5 đã được sữ dụng rông rải trong các chương trình an ninh mạng, và cũng thường được dùng để kiểm tra tính nguyên vẹn của tập tin.

MD5 được thiết kế bởi Ronald Rivest vào năm 1991 để thay thế cho hàm băm trước đó, MD4 (cũng do ông thiết kế, trước đó nữa là MD2).

3.1.4.2 Nguyên tắc hoạt động :

- **Thuật giải :**

MD5 biến đổi một thông điệp có chiều dài bất kì thành một khối có kích thước cố định 128 bits. Thông điệp đưa vào sẻ được cắt thành các khối 512 bits. Thông điệp được đưa vào bộ đệm để chiều dài của nó sẻ chia hết cho 512.

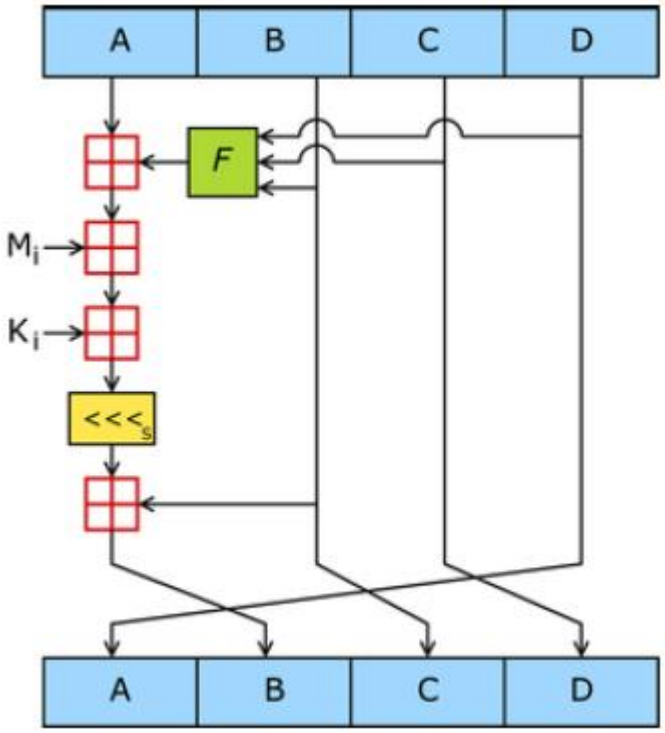
Bộ đệm hoạt động như sau:

- Trước tiên nó sẻ chèn bit 1 vào cuối thông điệp.
- Tiếp đó là hàng loạt bit Zero cho tới khi chiều dài của nó nhỏ hơn bội số của 512 một khoảng 64 bit .
- Phần còn lại sẻ được lấp đầy bởi một số nguyên 64 bit biểu diễn chiều dài ban đầu của thông điệp.

Thuật toán chính của MD5 hoạt động trên một bộ 128 bit. Chia nhỏ nó ra thành 4 từ 32 bit, kí hiệu là A,B,C và D. Các giá trị này là các hằng số cố định.

Sau đó thuật toán chính sẻ luân phiên hoạt động trên các khối 512 bit. Mỗi khối sẻ phối hợp với một bộ. Quá trình xữ lý một khối thông điệp bao gồm 4 bước tương tự nhau, gọi là vòng ("round"). Mỗi vòng lại gồm 16 quá trình tương tự nhau dựa trên hàm một chiều F, phép cộng module và phép xoay trái…

Hình bên dưới mô tả một quá trình trong một vòng. Có 4 hàm một chiều F có thể sử dụng. Mỗi vòng sử dụng một hàm khác nhau.



Hình 3: Một vòng của MD5.

Hàm băm MD5 (còn được gọi là hàm tóm tắt thông điệp - message degests) sẻ trả về một chuỗi số thập lục phân gồm 32 số liên tiếp.

Dưới đây là các ví dụ mô tả các kết quả thu được sau khi băm :

MD5 (“ The quick brown for jumps over the lazy dog ”)

= 9e107d372bb6826bd81d3542a419d6

Thậm chỉ chỉ cần một thay đổi nhỏ cũng làm thay đổi hoàn toàn kết quả trả về:

MD5 (“ The quick brown **fox** jumps over the lazy cog ”)=1055d3e698d289f2af8663725127bd4b

Ngay cả một chuỗi rỗng cũng cho ra một kết quả phức tạp:

MD5("") = d41d8cd98f00b204e9800998ecf8427e

3.2 HIỆN THỰC CHƯƠNG TRÌNH:

Thư viện dùng chung trong chương trình để mã hóa là : System.Security.Cryptography;

Chương trình gồm 3 giải thuật: DES, RSA, AES, và các hàm hash kiểm tra tính toàn vẹn dự liệu.

DES :

Cấu trúc chung của DES cũng như AES như sau:

```
public static void Encrypt_DES(string fileIn, string fileOut, string Password)
      {
          FileStream fsIn = new FileStream(fileIn, FileMode.Open, FileAccess.Read);
          FileStream fsOut = new FileStream(fileOut, FileMode.OpenOrCreate, FileAccess.Write);
```

```
        PasswordDeriveBytes pdb = new PasswordDeriveBytes(Password, new byte[] { 0x49, 0x76,
        0x61, 0x6e, 0x20, 0x4d, 0x65, 0x64, 0x76, 0x65, 0x64, 0x65, 0x76 });

        DES alg = DES.Create();
        alg.Key = pdb.GetBytes(8);
        alg.IV = pdb.GetBytes(8);


        CryptoStream cs = new CryptoStream(fsOut, alg.CreateEncryptor(), CryptoStreamMode.Write);

        int bufferLen = 4096;
        byte[] buffer = new byte[bufferLen];
        int bytesRead;

        do
        {
            bytesRead = fsIn.Read(buffer, 0, bufferLen);
            cs.Write(buffer, 0, bytesRead);
        } while (bytesRead != 0);

        cs.Close();
        fsIn.Close();

    }


public static void Decrypt_DES(string fileIn, string fileOut, string Password)
    {

        FileStream fsIn = new FileStream(fileIn,FileMode.Open, FileAccess.Read);
        FileStream fsOut = new FileStream(fileOut,FileMode.OpenOrCreate, FileAccess.Write);


        PasswordDeriveBytes pdb = new PasswordDeriveBytes(Password, new byte[] {0x49, 0x76, 0x61,
         0x6e, 0x20, 0x4d, 0x65, 0x64, 0x76, 0x65, 0x64, 0x65, 0x76});
        DES alg = DES.Create();

        alg.Key = pdb.GetBytes(8);
        alg.IV = pdb.GetBytes(8);

        CryptoStream cs = new CryptoStream(fsOut, alg.CreateDecryptor(), CryptoStreamMode.Write);

        int bufferLen = 4096;
        byte[] buffer = new byte[bufferLen];
        int bytesRead;

        do {
            bytesRead = fsIn.Read(buffer, 0, bufferLen);
            cs.Write(buffer, 0, bytesRead);
        } while(bytesRead != 0);

        cs.Close();
        fsIn.Close();
    }
```

Trong đó, quan trọng là hàm: `DES alg = DES.Create();` được sử dụng trong việc sử dụng thư viện mã hóa của C#.
Tiến hành đọc file theo kiểu CBC, tức là chia theo các block, mỗi block 4096 byte.
Hàm PasswordDeriveBytes pdb = new PasswordDeriveBytes(Password, new byte[] {0x49, 0x76, 0x61, 0x6e, 0x20, 0x4d, 0x65, 0x64, 0x76, 0x65, 0x64, 0x65, 0x76}); có nhiệm vụ lấy string password từ file và chuyển sang đúng định dạng của thư viện.

AES :

Tương tự như DES, thay `DES alg = DES.Create();` bởi `Aes alg = Aes.Create();`
Cũng tiến hành đọc theo từng block kiểu CBC.

RSA :

RSA là 1 kiểu mã hóa tốn kém và đắt đỏ, bởi thời gian lâu và giới hạn độ lớn dữ liệu thấp, ngược lại độ bảo mật cao như đã phân tích ở trên.
Bởi vậy, khi mã hóa RSA đối với lượng dữ liệu lớn, người ta thường sử dụng một giải thuật mã hóa đối xứng như DES, AES hoặc Rijndael (phiên bản trước của AES và được sử dụng trong bài tập lớn) , key sinh ra bởi mã hoá đối xứng sẽ tiếp tục sử dụng RSA để mã hóa, sau đó người dùng trao đổi khóa public key cũng như dùng private key để giải mã khóa đối xứng, sau đó dùng khóa đối xứng để giải mã ra file đã được mã hóa bằng thuật toán mã hóa đối xứng.

Đầu tiên, cần thực hiện hàm sinh key: publickey và privatekey cho RSA.
Key public:

```
            var csp = new RSACryptoServiceProvider(2048);
          //lets take a new CSP with a new 2048 bit rsa key pair

            var pubKey = csp.ExportParameters(false);

        //converting the public key into a string representation
            string pubKeyString;
            {
                //we need some buffer
                var sw = new System.IO.StringWriter();
                //we need a serializer
                var xs = new System.Xml.Serialization.XmlSerializer(typeof(RSAParameters));
                //serialize the key into the stream
                xs.Serialize(sw, pubKey);
                //get the string from the stream
                pubKeyString = sw.ToString();
            }
```

Key private:

```
          var privKey = csp.ExportParameters(true);

            string privateKeyString;
            {
                //we need some buffer
                var sw = new System.IO.StringWriter();
                //we need a serializer
                var xs = new System.Xml.Serialization.XmlSerializer(typeof(RSAParameters));
                //serialize the key into the stream
                xs.Serialize(sw, privKey);
                //get the string from the stream
                privateKeyString = sw.ToString();
            }
```

Đầu tiên, người dùng nhập key.txt để sử dụng cho mã hóa đối xứng, thuật toán nhóm lựa chọn là Rijndael.
Tương tự như DES cũng như Aes, ta cần hàm `Rijndael alg = Rijndael.Create();`
Hỗ trợ bởi thư viện System.Security.Cryptography để mã hóa dự liệu.

Sau đó tiến hành sinh key public và private, 2 key có độ dài 2048 bits này sẽ được lưu lại, người dùng sử dụng key public, đồng thời đưa vào key.txt (key sử dụng để mã hóa bất đối xứng) để mã hóa file key.txt của mã hóa đối xứng. Cuối cùng ta được 2 file: cirpher.txt (mã hóa RSA của key đối xứng) + file mã hóa .code của dữ liệu (mã hóa bằng thuật toán đối xứng). Ta sẽ gửi 2 file này cho người nhận.

Người nhận tiến hành nhận 2 file trên, dùng private key để giải mã file cirphertext.txt được file key.txt (key mã hóa đối xứng), sau đó dùng key này giải mã dữ liệu, cuối cùng được dữ liệu ban đầu.

PHẦN 4: PHÂN TÍCH, KẾT LUẬN

4.1 PHÂN TÍCH CÁC GIẢI THUẬT:

4.1.1 Giải thuật mã hóa đối xứng DES:

DES có tính chất bù:

$$E_K(P) = C \Leftrightarrow E_{\overline{K}}(\overline{P}) = \overline{C}$$

trong đó $\overline{x}$ là phần bù của x theo từng bít (1 thay bằng 0 và ngược lại). E_K là bản mã hóa của E với khóa K. P và C là văn bản rõ (trước khi mã hóa) và văn bản mã (sau khi mã hóa). Do tính bù, ta có thể giảm độ phức tạp của tấn công duyệt toàn bộ xuống 2 lần (tương ứng với 1 bít) với điều kiện là ta có thể lựa chọn bản rõ.

Ngoài ra DES còn có 4 khóa yếu (weak keys). Khi sử dụng khóa yếu thì mã hóa (E) và giải mã (D) sẽ cho ra cùng kết quả:

$E_K(E_K(P)) = P$ or equivalently , $E_K = D_K$

Bên cạnh đó, còn có 6 cặp *khóa nửa yếu* (semi-weak keys). Mã hóa với một khóa trong cặp, K_1, tương đương với giải mã với khóa còn lại, K_2:

$$E_{K_1}(E_{K_2}(P)) = P \text{ or equivalently, } E_{K_2} = D_{K_1}$$

Tuy nhiên có thể dễ dàng tránh được những khóa này khi thực hiện thuật toán, có thể bằng cách thử hoặc chọn khóa một cách ngẫu nhiên. Khi đó khả năng chọn phải khóa yếu là rất nhỏ.

DES đã được chứng minh là không tạo thành nhóm. Nói một cách khác, tập hợp $\{E_K\}$ (cho tất cả các khóa có thể) theo phép hợp thành không tạo thành một nhóm hay gần với một nhóm (Campbell and Wiener, 1992). Vấn đề này vẫn còn để mở và nếu như không có tính chất này thì DES có thể bị phá vỡ dễ dàng hơn và việc áp dụng DES nhiều lần (ví dụ như trong Triple DES) sẽ không làm tăng thêm độ an toàn của DES.

4.1.2 Giải thuật mã hóa bất đối xứng RSA:

Việc tìm ra 2 số nguyên tố đủ lớn p và q thường được thực hiện bằng cách thử xác suất các số ngẫu nhiên có độ lớn phù hợp (dùng phép kiểm tra nguyên tố cho phép loại bỏ hầu hết các hợp số).

P và Q còn cần được chọn không quá gần nhau để phòng trường hợp phân tích n bằng phương pháp phân tích Fermat . Ngoài ra, nếu p-1 hoặc q-1 có thừa số nguyên tố nhỏ thì n cũng có thể dễ dàng bị phân tích và vì thế p và q cũng cần được thử để tránh khả năng này.

Bên cạnh đó, cần tránh sử dụng các phương pháp tìm số ngẫu nhiên mà kẻ tấn công có thể lợi dụng để biết thêm thông tin về việc lựa chọn (cần dùng các bộ tạo số ngẫu nhiên tốt). Yêu cầu ở đây là các số được lựa chọn cần đồng thời ngẫu nhiên và không dự đoán được. Đây là các yêu cầu khác nhau: một số có thể được lựa chọn ngẫu nhiên (không có kiểu mẫu trong kết quả) nhưng nếu có thể dự đoán được dù chỉ một phần thì an ninh của thuật toán cũng không được đảm bảo. Một ví dụ là bảng các số ngẫu nhiên do tập đoàn Rand xuất bản vào những năm 1950 có thể rất thực sự ngẫu nhiên nhưng kẻ tấn công cũng có bảng này. Nếu kẻ tấn công đoán được một nửa chữ số của p hay q thì chúng có thể dễ dàng tìm ra nửa còn lại (theo nghiên cứu của Donald Coppersmith vào năm 1997)

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

Mặc dù e đã từng có giá trị là 3 nhưng hiện nay các số mũ nhỏ không còn được sử dụng do có thể tạo nên những lỗ hổng (đã đề cập ở phần chuyển đổi văn bản rõ). Giá trị thường dùng hiện nay là 65537 vì được xem là đủ lớn và cũng không quá lớn ảnh hưởng tới việc thực hiện hàm mũ.

- **Tốc độ :**

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

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

Phân phối hóa: Cũng giống như các thuật toán mã hóa khác, cách thức phân phối khóa công khai là một trong những yếu tố quyết định đối với độ an toàn của RSA. Quá trình phân phối khóa cần chống lại được tấn công đứng giữa (man-in-the-middle attack). Giả sử Eve có thể gửi cho Bob một khóa bất kỳ và khiến Bob tin rằng đó là khóa (công khai) của Alice. Đồng thời Eve có khả năng đọc được thông tin trao đổi giữa Bob và Alice. Khi đó, Eve sẽ gửi cho Bob khóa công khai của chính mình (mà Bob nghĩ rằng đó là khóa của Alice). Sau đó, Eve đọc tất cả văn bản mã hóa do Bob gửi, giải mã với khóa bí mật của mình, giữ 1 bản copy đồng thời mã hóa bằng khóa công khai của Alice và gửi cho Alice. Về nguyên tắc, cả Bob và Alice đều không phát hiện ra sự can thiệp của người thứ ba. Các phương pháp chống lại dạng tấn công này thường dựa trên các chứng thực khóa công khai (digital certificate) hoặc các thành phần của hạ tầng khóa công khai (public key infrastructure - PKI).

4.1.3 Giải thuật mã hóa AES:

4.1.3.2 Một vài đặc điểm AES :

Nếu các kỹ thuật tấn công được cải thiện thì AES có thể bị phá vỡ vì ranh giới giữa số chu trình của thuật toán và số chu trình bị phá vỡ quá nhỏ. Tháng 10 năm 2005, Adi Shamir và 2 nhà nghiên cứu khác có một bài tấn công minh họa một vài dạng khác. Với tấn công thực hiện 2^{120} là thành công dù tấn công này chưa thể thực hiện trong thực tế.

Cấu trúc toán học của AES có mô tả khá đơn giản. Điều này có thể là nguy cơ để phá mã AES trong truong lai.

Bảng so sánh các giải thuật DES, RSA, AES

	DES	RSA	AES
Loại giải thuật	Mã hóa đối xứng	Mã hóa bất đối xứng	Mã hóa đối xứng
Key	1 Key.Độ dài khóa 64 bits	2 Key: Key Public và Key Private. Độ dài khóa không cố định.	1 Key. Độ dài khóa 128 bits, 192 bits, 256 bits.
Thời gian mã hóa	Nhanh.	Lâu.	Nhanh.
Tấn công	-Duyệt toàn bộ. -Vi sai. -Tuyến tính. -Davies.	-Dựa vào thời gian. -Lựa chọn thích nghi mã.	-Tấn công kênh bên.
Độ phức tạp giải thuật.	Đơn giản. Có 16 chu trình giống nhau trong quá trình xử lý.	Phức tạp. Dựa trên 2 vấn đề toán học: bài toán phân tích ra thừa số nguyên tố các số nguyên tố và bài toán tính căn bậc e môđun n (m^e=c mod n).	Đơn giản. Quy trình mã hóa và giải mã đảo ngược nhau.

KẾT LUẬN:

Hiện nay DES được xem là không đủ an toàn cho nhiều ứng dụng. Nguyên nhân chủ yếu là độ dài 56 bit của khóa là quá nhỏ. Khóa DES đã từng bị phá trong vòng chưa đầy 24 giờ. Đã có rất nhiều kết quả

phân tích cho thấy những điểm yếu về mặt lý thuyết của mã hóa có thể dẫn đến phá khóa, tuy chúng không khả thi trong thực tiễn. Thuật toán được tin tưởng là an toàn trong thực tiễn có dạng Triple DES (thực hiện DES ba lần), mặc dù trên lý thuyết phương pháp này vẫn có thể bị phá. Gần đây DES đã được thay thế bằng AES (*Advanced Encryption Standard*, hay Tiêu chuẩn Mã hóa Tiên tiến).

AES đang là tiêu chuẩn của giải thuật mã hóa và giải mã. Nhưng với những điểm yếu phân tích ở trên trong tương lai không xa nó có thể bị phá vỡ. Chúng ta cần nghiên cứu để tối ưu hóa AES khắc phục những hạn chế của nó.

Một hướng mã hóa đang được ứng dụng rộng là dạng Hypric(lai). Kết hợp giữa mã hóa đối xứng và bất đối xứng. Chúng ta dùng giải thuật RSA để mã hóa Key của giải thuật đối xứng (ví dụ: AES) gửi qua bên nhận. RSA khi mã hóa key sẽ có được tính bảo mật cao. Sau có key hai bên sẽ gửi dữ liệu qua lại bằng giải thuật mã hóa đối xứng. Điều này giúp quá trình mã hóa diễn ra nhanh hơn. Vì giải thuật mã hóa đối xứng đòi hỏi thời gian mã hóa và giải mã nhanh hơn so với giải thuật bất đôi xứng.

II. DES, AES VÀ RSA TRONG CHƯƠNG TRÌNH :

- Khi demo chương trình, DES và AES (mã hóa đối xứng) tỏ ra ưu việt hơn các khoảng về hiệu suất, thời gian rất nhiều so với RSA.
- RSA không thể mã hóa dữ liệu lớn, nên thường sử dụng để mã hóa thông tin cá nhân người dùng, password, key của mã hóa đối xứng.
- Khi tiến hành với các file lớn, AES cho thời gian tốt hơn DES, tuy nhiên các file nhỏ, sự khác biệt không nhiều.
- RSA không tự mình mã hóa được các file lớn, mà người dùng chủ yếu file dùng các giải thuật mã hóa đối xứng khác.
- Chương trình mã hóa có tính chính xác cao, đặc biệt đối với dữ liệu vừa phải, vì không có điều kiện thử nghiệm đối với dự liệu lớn hoặc rất lớn nên chưa thể kết luận độ chính xác lúc này.

❖ Ưu điểm chương trình:

- Nhóm đã hoàn thành 3 giải thuật mã hóa cơ bản: DES, AES, RSA.
- Chương trình có khả năng mã hóa hầu như tất cả các file, từ : ảnh, pdf, word, excel, text, binary….
- Chương trình mã hóa và giải mã ra có độ chính xác cao.
- Sử dụng các hàm hash như: MD5, SHA,… kiểm tra tính toàn vẹn trong hầu hết cho kết qua tốt.

➢ Nhược điểm chương trình:

- Chưa hoàn thành các tính năng nâng cao trong bài tập lớn đưa ra.
- Chưa nén dữ liệu để tiết kiệm thời gina và hiệu suất.
- Chỉ dừng lại ở mức cơ bản.

PHẦN 5: HƯỚNG PHÁT TRIỂN

- Tìm hiểu về phương pháp nén dữ liệu trong mã hóa để tăng hiệu năng.
- Mô tả đồ họa về quá trong sinh key, mã hóa, giải mã.
- Thêm thanh trạng thái miêu tả quá trình.
- Bổ sung các giải thuật mã hóa hiện đại, hiện nay đang được sử dụng.
- Tìm hiểu cơ chế của hàm 1 chiều, để tăng tính bảo mật của Key bằng cách cho Key qua hàm 1 chiều, chỉ tiến hành so sánh kết quả đó, để khi bị lộ, hacker cũng không thể đoán ra Key đó.
- Tìm hiểu thêm về cơ chế bảo mật mật khẩu hiện nay đang sử dụng rộng rãi.

PHẦN 6: THAM KHẢO

1.Cryptography and Network Security principles and practices 6th edition - William stallings.

2. https://vi.wikipedia.org/wiki/AES_(mã_hóa).

3. http://aita.gov.vn/tin-tuc/1452/tieu-chuan-ky-thuat-ve-udcntt-trong-cqnn-tieu-chuan-aes-%E2%80%93-tieu-chuan-ma-hoa-tien-tien-19 .

4. http://aita.gov.vn/tin-tuc/1454/tieu-chuan-ky-thuat-ve-udcntt-trong-cqnn-tieu-chuan-rsa-giai-thuat-ma-hoa-cong-khai-rsa-21 .

5. http://aita.gov.vn/tin-tuc/1453/tieu-chuan-ky-thuat-ve-udcntt-trong-cqnn-tieu-chuan-3des-%E2%80%93-thuat-toan-ma-hoa-khoi-3-lan-20 .

6. http://voer.edu.vn/m/md5/9fa0cb36 .

PHẦN 7: PHỤ LỤC I

Nhiệm vụ, công việc, và hoàn thành các nội dung BTL:

- **Nhiệm vụ**:

  - Đặng Xuân Ánh: Tham gia tìm hiểu các giải thuật, lựa chọn ngôn ngữ, tìm hiểu thư viện trong C#, lập trình, hoàn thiện báo cáo.
  - Đặng Minh Việt: Tham gia tìm hiểu các giải thuật, tìm hiểu thư viện, lập trình, báo cáo.
  - Lê Tuấn Vũ: Tham gia tìm hiểu các giải thuật, Tester, bổ sung, nêu ý kiến, hoàn thiện demo, báo cáo.

- **Hoàn thành:**

  Cơ bản các thành viên đã hoàn thành công việc.

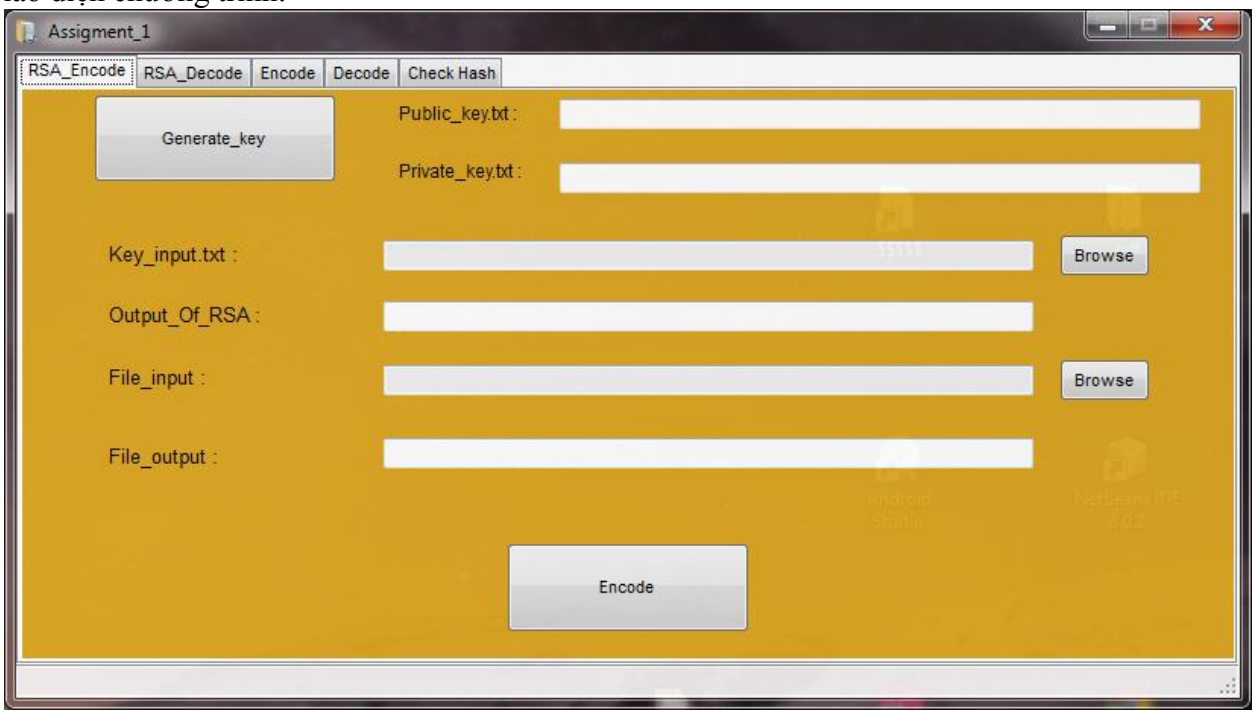
- **Đánh giá:**

	Đặng Xuân Ánh	Đặng Minh Việt	Lê Tuấn Vũ
Mức độ hoàn thành dealine về mặt thời gian	100%	100%	90%
Mức độ hoàn thành công việc được giao	100%	90%	90%
Đánh giá chất lượng công việc hoàn thành	Tốt	Tốt	Khá
Đánh giá khả năng làm việc nhóm	Tốt	Tốt	Tốt
Đánh giá mức độ tuân thủ nội quy do nhóm đề ra	Tốt	Tốt	Tốt

PHẦN 8: PHỤ LỤC II

Hướng dẫn demo và chạy chương trình:

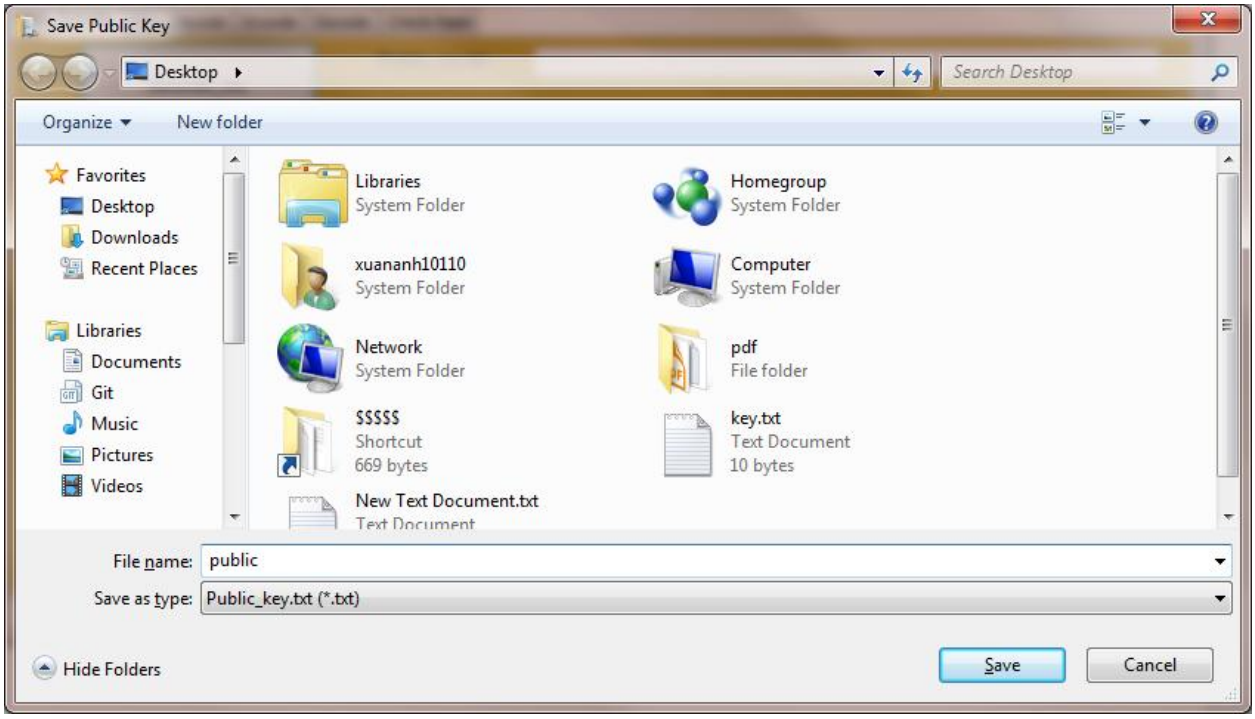
- Giao diện chương trình:



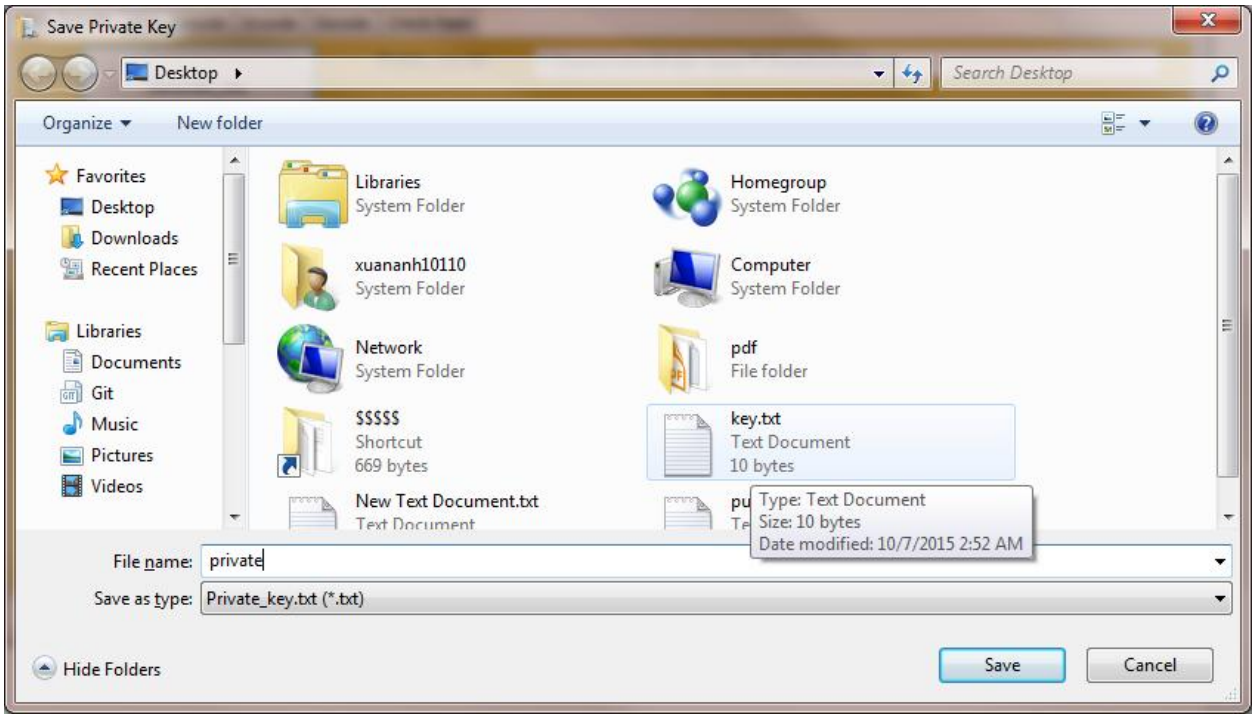
-Đầu tiên là mã hóa RSA:

+ Đầu tiên người dùng click button Generate_key để chương trình sinh khóa private key và public key.

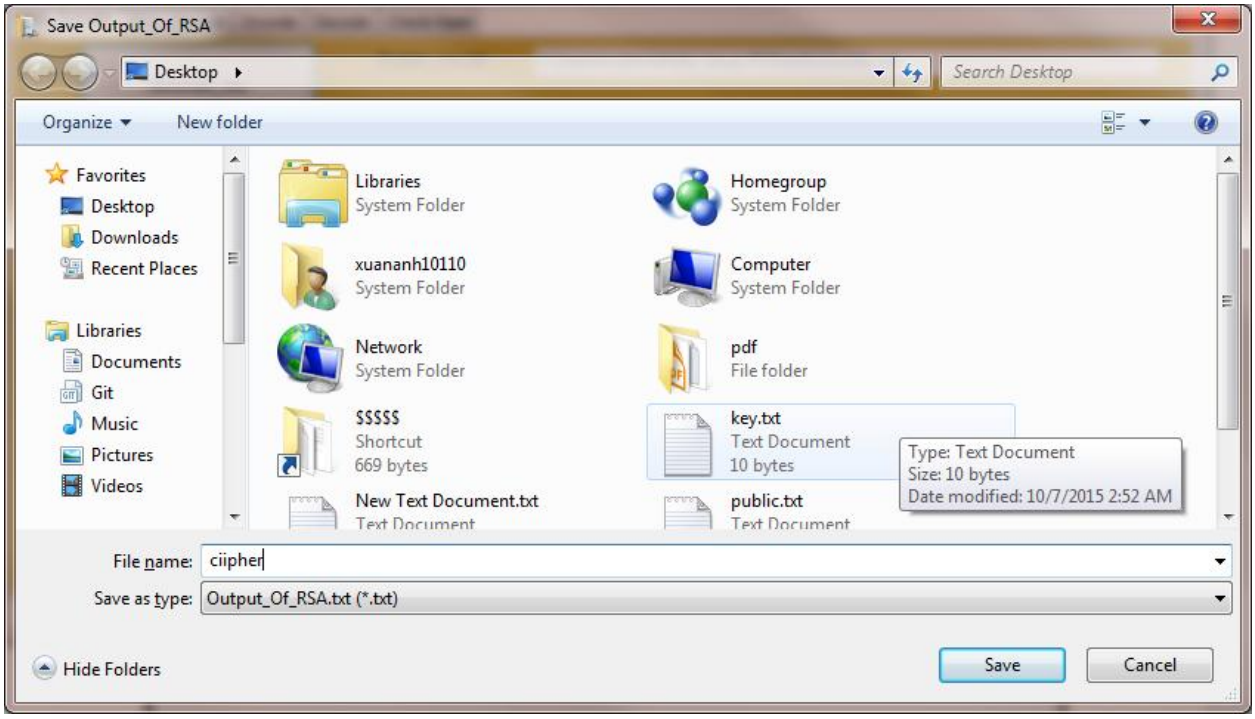
+ Một bảng chọn hiện ra, người dùng Browse đến nơi lưu file, nhập tên file publickey -> Save.



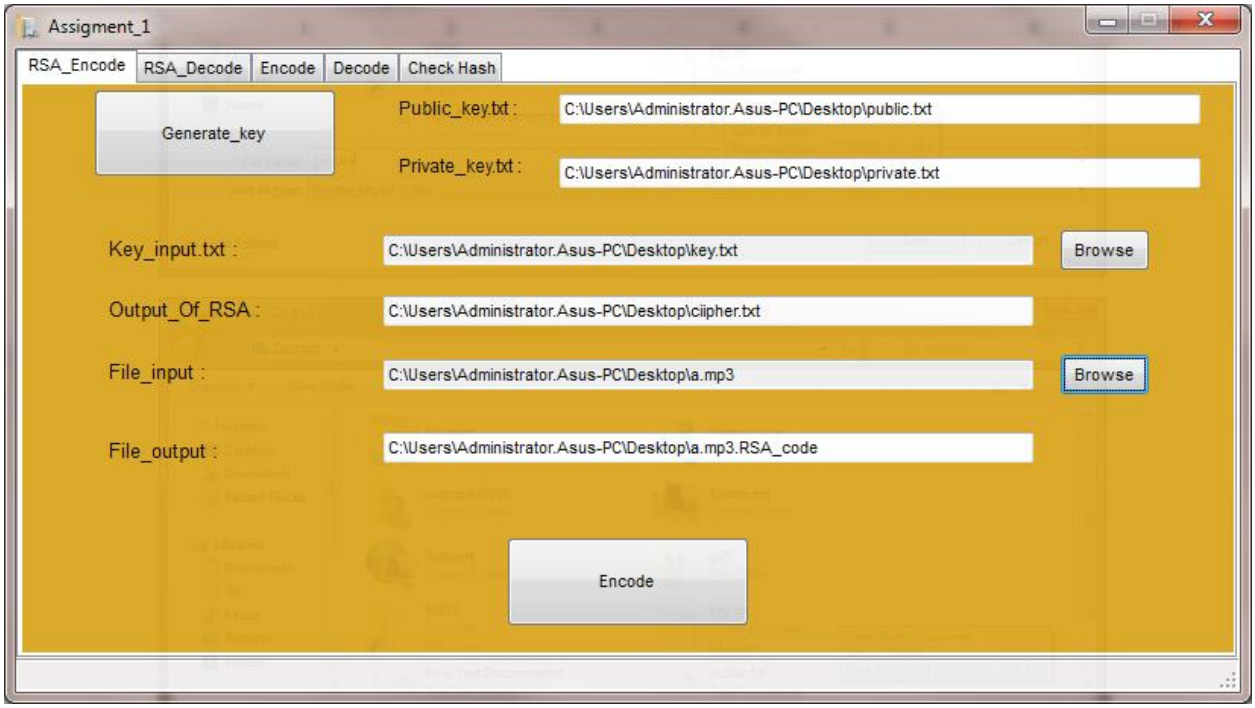
-Tương tự lưu private key.



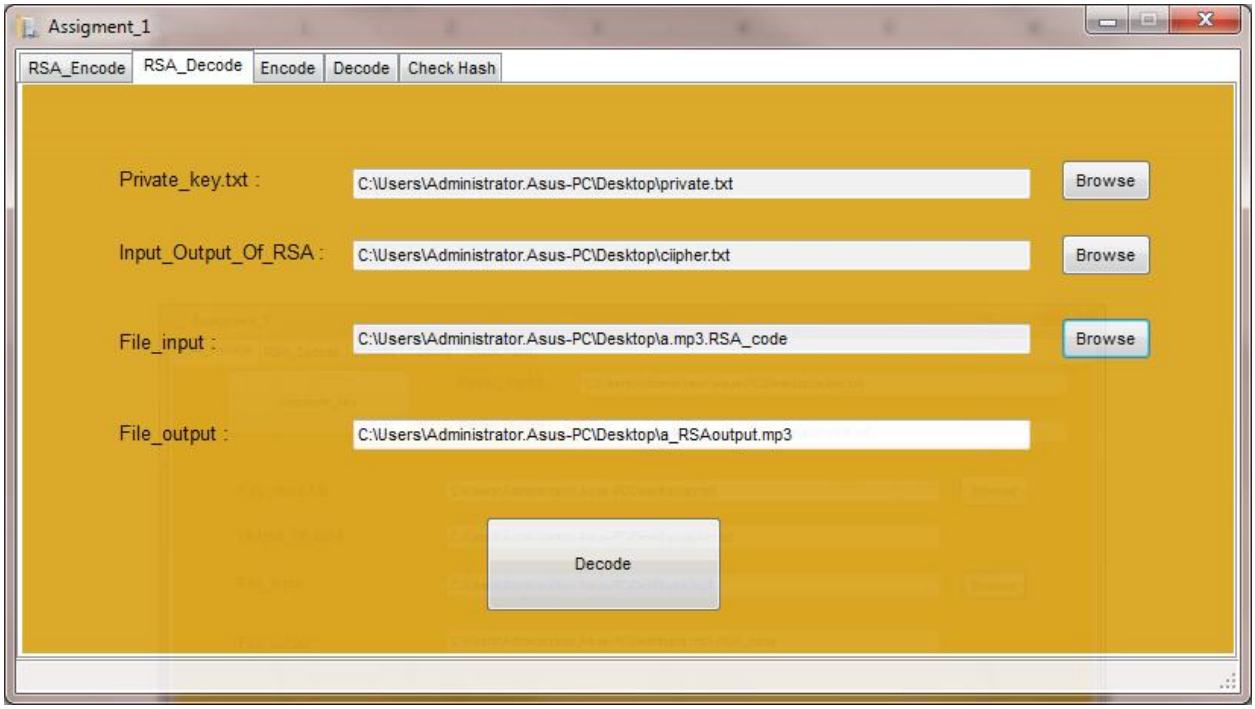
-Tương tự chọn nơi lưu file kết quả của quá trình mã hóa RSA (mã hóa key dự định sử dụng của mã hóa đối xứng).



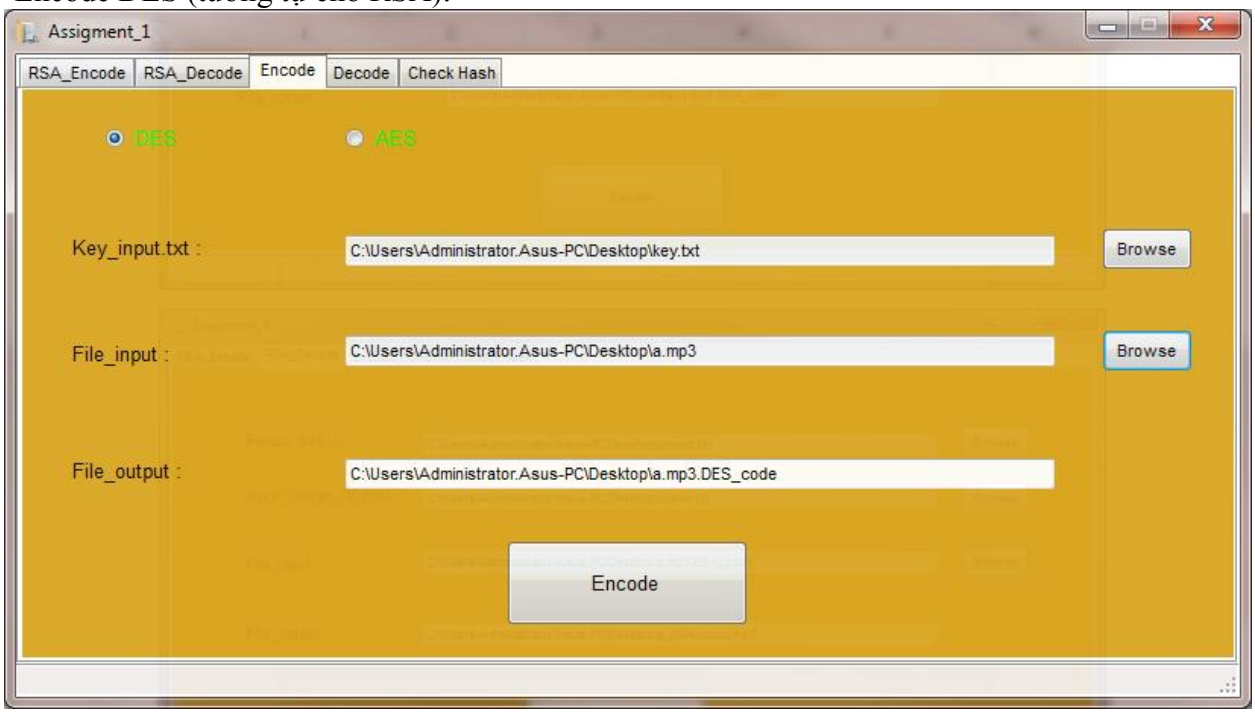
-Người dùng Browse Key_input.txt (key dùng để mã hóa đối xứng – cụ thể là Rijndael), Bowse File_input chứa dữ liệu cần mã hóa -> Encode



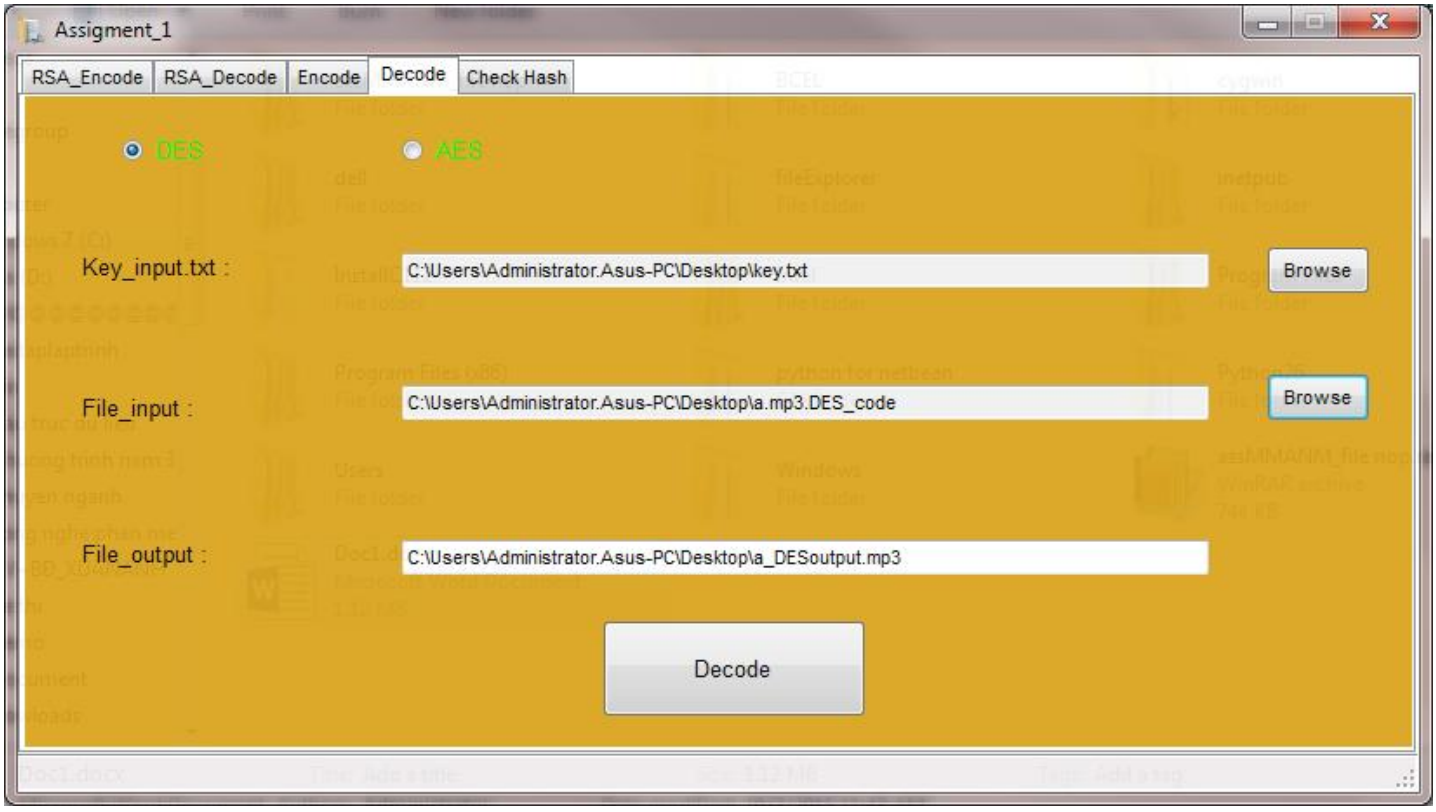
-Decode RSA



-Encode DES (tương tự cho RSA):



-Decode DES (tương tự cho RSA):



- Check hash:
- Người dùng Browse đến 2 file source(file nguồn) và destination file (file đích) để tiến hành check. Sau đó ấn vào các button Check MD5, CheckSHA1, Check SHA256…. 9e63 lựa chọn kiểu Check.
- 2 textbox Code hash input file và Code Hash output hash hiện 2 mã lần lượt của file nguồn và file đích.
- Nếu thành công, hiện ra như sau: