

ĐẠI HỌC ĐẠI HỌC QUỐC GIA TPHCM
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



BÁO CÁO ASSIGNMENT 1
MẬT MÃ AN NINH MẠNG

GIÁO VIÊN HƯỚNG DẪN :

Nguyễn Nhật Nam

NHÓM SINH VIÊN THỰC HIỆN :

Họ Tên	Mã Số Sinh Viên
Nguyễn Hồng Phước	51202890
Trương Quang Kha	51201570
Thái Quốc Tuấn	51307129

Họ và Tên	Nội dung công việc	Dóng góp (%)
Nguyễn Hồng Phước	-Tìm hiểu các thuật toán mã hóa mà nhóm thực hiện. -Hiện thực thuật toán mã hóa DES và MD-5. -Tổng hợp chương trình	100%
Trương Quang Kha	-Tìm hiểu các thuật toán mã hóa mà nhóm thực hiện. -Hiện thực thuật toán mã hóa RSA. -Viết báo cáo	100%
Thái Quốc Tuấn	-Tìm hiểu các thuật toán mã hóa mà nhóm thực hiện. -Hiện thực thuật toán mã hóa AES -Viết báo cáo	100%

Mục lục

1 Giới Thiệu.	4
2 Giới thiệu tổng quan về các giải thuật.	4
2.1 Thuật toán mã hóa AES (Advanced Encryption Standard)	4
2.1.1 Hàm SubBytes()	6
2.1.2 Hàm ShiftRows()	7
2.1.3 Hàm MixColumns()	7
2.1.4 Hàm AddRoundKey()	8
2.1.5 Hàm InvShiftRows()	8
2.1.6 Hàm InvSubbytes()	9
2.1.7 Hàm InvMixColumns()	9
2.1.8 Hàm nghịch đảo của hàm AddRoundKey()	9
2.2 Thuật toán mã hóa DES.	10
2.3 Thuật toán mã hóa RSA	12
2.4 Hàm băm MD-5	14
3 Nội dung công việc đã làm.	14
3.1 Thuật toán mã hóa AES	14
3.2 Thuật toán mã hóa RSA	14
3.3 Thuật toán mã hóa DES	15
3.4 Hàm băm MD5	15
4 Demo chương trình	16
4.1 Giải thuật mã hóa DES (DES)	16
4.1.1 Phần mã hóa	17
4.1.2 Phần giải mã	17
4.2 Thuật toán mã hóa RSA	17
4.2.1 Phần mã hóa	18
4.2.2 Phần giải mã	19
4.3 Hàm băm MD-5	19
4.3.1 Phần file	20
4.3.2 Phần text	21
4.4 Giải thuật mã hóa AES	21
4.4.1 Phần mã hóa	22
4.4.2 Phần giải mã	23
5 Phân tích và kết luận	23
5.1 Phân tích	23
5.2 Kết luận	23
6 Hướng phát triển	23
7 Tài liệu tham khảo	24

1 Giới Thiệu.

- Môi trường thực hiện: Ngôn ngữ C#.
- Tiến trình thực hiện:
 - Tìm hiểu cách thức hoạt động của các giải thuật DES, RSA, AES, MD-5.
 - Kết hợp với thư viện System.Security.Cryptography có sẵn trong C# để hiện thực lại các giải thuật trên.
- Chương trình hỗ trợ mã hóa một tập tin với định dạng bất kỳ như: hình ảnh, âm thanh, doc, pdf...
- Quá trình mã hóa: nhận input là thư mục bất kỳ hoặc tập tin bất kỳ và key được nhập trực tiếp hoặc chọn từ file và một số option khác , output là tập tin hay thư mục chứa dữ liệu đã được mã hóa.
- File đã được mã hóa có phần mở rộng là : .tpEn
- Quá trình giải mã: nhận input là tập tin hay thư mục chứa dữ liệu đã được mã hóa và key được nhập trực tiếp hoặc tập tin text chứa chìa khóa giải mã (decryption key) và một số option khác , output chương trình là tập tin hoặc thư mục được giải mã thành công.
- Chương trình chưa thực hiện được một số các tính năng nâng cao như trong đề bài tập yêu cầu.

2 Giới thiệu tổng quan về các giải thuật.

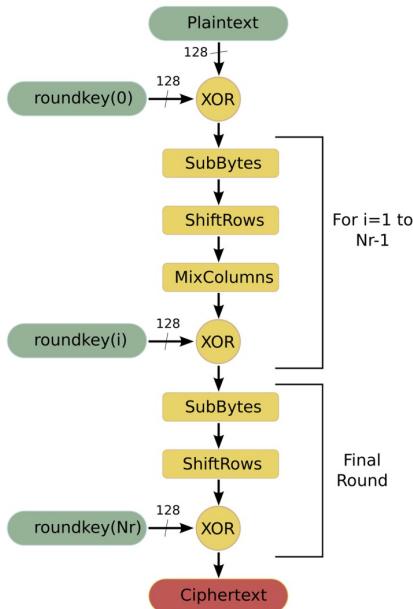
2.1 Thuật toán mã hóa AES (Advanced Encryption Standard)

Là một thuật toán mã hóa khối được chính phủ Hoa Kỳ áp dụng làm tiêu chuẩn mã hóa. Giống như tiêu chuẩn tiền nhiệm DES, AES được kỳ vọng áp dụng trên phạm vi thế giới và đã được nghiên cứu rất kỹ lưỡng. Thuật toán được đặt tên là "Rijndael" khi tham gia cuộc thi thiết kế AES.

Mặc dù 2 tên AES và Rijndael vẫn thường được gọi thay thế cho nhau nhưng trên thực tế thì 2 thuật toán không hoàn toàn giống nhau. AES chỉ làm việc với các khối dữ liệu (đầu vào và đầu ra) 128 bit và khóa có độ dài 128, 192 hoặc 256 bit trong khi Rijndael có thể làm việc với dữ liệu và khóa có độ dài bất kỳ là bội số của 32 bit nằm trong khoảng từ 128 tới 256 bit. Các khóa con sử dụng trong các chu trình được tạo ra bởi quá trình tạo khóa con Rijndael. Mỗi khóa con cũng là một cột gồm 4 byte.

Hầu hết các phép toán trong thuật toán AES đều thực hiện trong một trường hữu hạn của các byte. Mỗi khối dữ liệu 128 bit đầu vào được chia thành 16 byte (mỗi byte 8 bit), có thể xếp thành 4 cột, mỗi cột 4 phần tử hay là một ma trận 4x4 của các byte,nó

được gọi là ma trận trạng thái, hay vẫn tắt là trạng thái (tiếng Anh: state, trang thái trong Rijndael có thể có thêm cột). Trong quá trình thực hiện thuật toán các toán tử tác động để biến đổi ma trận trạng thái này.



Giải thuật Mã hóa

Khởi động vòng lặp

1. AddRoundKey — Mỗi cột của trạng thái đầu tiên lần lượt được kết hợp với một khóa con theo thứ tự từ đầu dãy khóa.

Vòng lặp

1. SubBytes — đây là phép thay đổi (phi tuyến) trong đó mỗi byte trong trạng thái sẽ được thay bằng một byte khác theo bảng tra (Rijndael S-box).
2. ShiftRows — dịch chuyển, các hàng trong trạng thái được dịch vòng theo số bước khác nhau.
3. MixColumns — quá trình trộn làm việc theo các cột trong khối theo một phép biến đổi tuyến tính.
4. AddRoundKey

Vòng lặp cuối

1. SubBytes
2. ShiftRows
3. AddRoundKey

2.1.1 Hàm SubBytes()

Hàm SubBytes() thực hiện phép thay thế các byte của mảng trạng thái bằng cách sử dụng một bảng thê S-Box, bảng thê này là khả nghịch và được xây dựng bằng cách kết hợp 2 biến đổi nhau:

- Nhận nghịch đảo trên trường hữu hạn GF (2^8), phần tử 00 được ánh xạ thành chính nó.
- Áp dụng biến đổi Affine sau (trên GF(2)):

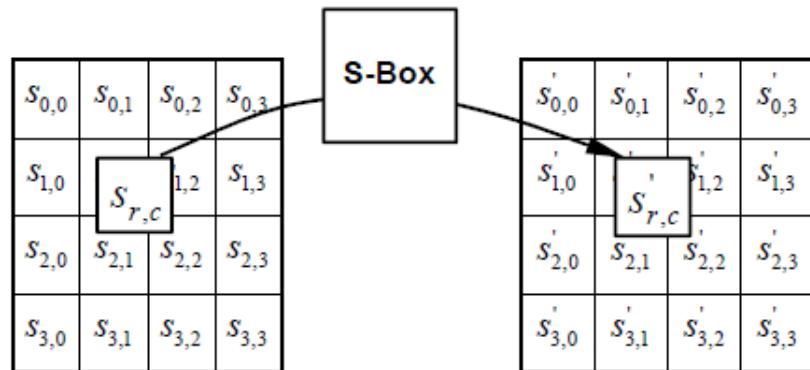
$$b'_i = b_i \oplus b_{(i+4)mod8} \oplus b_{(i+5)mod8} \oplus b_{(i+6)mod8} \oplus b_{(i+7)mod8} \oplus c_i$$

trong đó $0 \leq i \leq 8$ là bit thứ i của byte b tương ứng và c_i là bit thứ i của byte c với giá trị {63} hay {01100011}

Các phần tử biến đổi affine của S-box có thể được biểu diễn dưới dạng ma trận như sau :

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Hình sau minh họa kết quả của việc áp dụng hàm biến đổi SubBytes() đối với mảng trạng thái:



Bảng thế S-box được sử dụng trong hàm Subbytes() có thể được biểu diễn dưới dạng hexa như sau:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

trong đó chặng hạn nếu $S_{1,1} = \{53\}$ có nghĩa là giá trị thay thế sẽ được xác định bằng giao của hàng có chỉ số 5 với cột có chỉ số 3 trong bảng trên, điều này tương đương với việc $S'_{1,1} = \{ed\}$

2.1.2 Hàm ShiftRows()

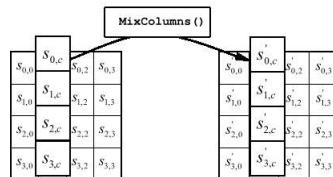
Là phép biến đổi các byte trên ba hàng cuối cùng của mảng trạng thái bằng cách dịch vòng thể hiện qua công thức:

$$S'_{r,c} = S_{r,(c+shift(r,Nb)modNb)}, \text{ với } 0 < r \leq 3 \text{ và } 0 \leq c < Nb$$

Số lần dịch vòng shift(r, Nb) phụ thuộc vào chỉ số vòng. Cụ thể shift(4,1)=1, shift(2,4)=1, shift(3,4)=3.

2.1.3 Hàm MixColumns()

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



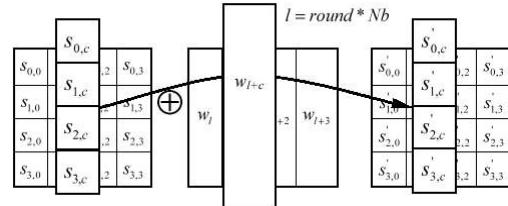
2.1.4 Hàm AddRoundKey()

Trong biến đổi AddRoundKey(), một khóa vòng được cộng với state bằng 1 phép XOR theo từng bit đơn giản. Mỗi khóa vòng gồm có 4 từ (128bit) được lấy từ lược đồ khóa. 4 từ đó được cộng vào state, sao cho:

$$[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}] = [s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}] \oplus [W_{4*i+c}] \text{ với } 0 \leq c < 4$$

Trong đó w_{4*i+c} là các từ thứ c của khóa vòng thứ i

$$W_i = [w_{4*i}, w_{4*i+1}, w_{4*i+2}, w_{4*i+3}]$$



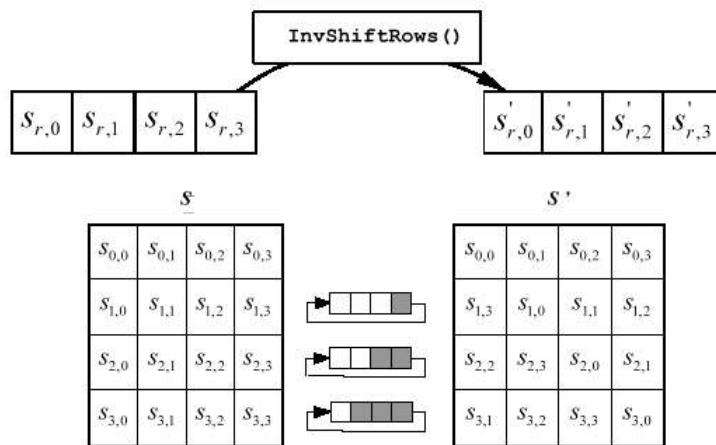
Giải thuật Giải mã

2.1.5 Hàm InvShiftRows()

Hàm này là hàm ngược của hàm ShiftRows(). Các byte của ba hàng cuối của mảng trạng thái sẽ được dịch vòng với các vị trí dịch khác nhau. Hàng đầu tiên không bị dịch, ba hàng cuối bị dịch đi Nb-Shift(r,Nb) byte.

Cụ thể hàm này tiến hành xử lý như sau:

$$S'_{r,(c+shift(r,Nb))modNb} = S_{r,c} \text{ với } 0 < r < 4, 0 \leq c < Nb$$



2.1.6 Hàm InvSubBytes()

Hàm này là hàm ngược của hàm SubBytes(), hàm sử dụng nghịch đảo của biến đổi Affine bằng cách thực hiện nhân nghịch đảo trên $GF(2^8)$

Bảng thê được sử dụng trong hàm:

x	y																	
	<table border="1"> <tr><td>0 1 2 3 4 5 6 7 8 9 a b c d e f</td></tr> <tr><td>0 52 09 6a d5 30 36 a5 38 bf 40 a3 9e 81 f3 d7 fb</td></tr> <tr><td>1 7c e3 39 82 9b 2f ff 87 34 8e 43 44 c4 de e9 cb</td></tr> <tr><td>2 54 7b 94 32 a6 c2 23 3d ee 4c 95 0b 42 fa c3 4e</td></tr> <tr><td>3 08 2e a1 66 28 d9 24 b2 76 5b a2 49 6d 8b d1 25</td></tr> <tr><td>4 72 f8 f6 64 86 68 98 16 d4 a4 5c cc 5d 65 b6 92</td></tr> <tr><td>5 6c 70 48 50 fd ed b9 da 5e 15 46 57 a7 8d 9d 84</td></tr> <tr><td>6 90 d8 ab 00 8c bc d3 0a f7 e4 58 05 18 b3 45 06</td></tr> <tr><td>7 d0 2c 1e 8f ca 3f 0f 02 c1 af bd 03 01 13 8a 6b</td></tr> <tr><td>8 3a 91 11 41 4f 67 dc ea 97 f2 cf ce f0 b4 e6 73</td></tr> <tr><td>9 96 ac 74 22 e7 ad 35 85 e2 f9 37 e8 1c 75 df 6e</td></tr> <tr><td>a 47 f1 1a 71 1d 29 c5 89 6f b7 62 0e aa 18 be 1b</td></tr> <tr><td>b fc 56 3e 4b c6 d2 79 20 9a db c0 fe 78 cd 5a f4</td></tr> <tr><td>c 1f dd a8 33 88 07 c7 31 b1 12 10 59 27 80 ec 5f</td></tr> <tr><td>d 60 51 7f a9 19 b5 4a 0d 2d e5 7a 9f 93 c9 9c ef</td></tr> <tr><td>e a0 e0 3b 4d ae 2a f5 b0 c8 eb bb 3c 83 53 99 61</td></tr> <tr><td>f 17 2b 04 7e ba 77 d6 26 e1 69 14 63 55 21 0c 7d</td></tr> </table>	0 1 2 3 4 5 6 7 8 9 a b c d e f	0 52 09 6a d5 30 36 a5 38 bf 40 a3 9e 81 f3 d7 fb	1 7c e3 39 82 9b 2f ff 87 34 8e 43 44 c4 de e9 cb	2 54 7b 94 32 a6 c2 23 3d ee 4c 95 0b 42 fa c3 4e	3 08 2e a1 66 28 d9 24 b2 76 5b a2 49 6d 8b d1 25	4 72 f8 f6 64 86 68 98 16 d4 a4 5c cc 5d 65 b6 92	5 6c 70 48 50 fd ed b9 da 5e 15 46 57 a7 8d 9d 84	6 90 d8 ab 00 8c bc d3 0a f7 e4 58 05 18 b3 45 06	7 d0 2c 1e 8f ca 3f 0f 02 c1 af bd 03 01 13 8a 6b	8 3a 91 11 41 4f 67 dc ea 97 f2 cf ce f0 b4 e6 73	9 96 ac 74 22 e7 ad 35 85 e2 f9 37 e8 1c 75 df 6e	a 47 f1 1a 71 1d 29 c5 89 6f b7 62 0e aa 18 be 1b	b fc 56 3e 4b c6 d2 79 20 9a db c0 fe 78 cd 5a f4	c 1f dd a8 33 88 07 c7 31 b1 12 10 59 27 80 ec 5f	d 60 51 7f a9 19 b5 4a 0d 2d e5 7a 9f 93 c9 9c ef	e a0 e0 3b 4d ae 2a f5 b0 c8 eb bb 3c 83 53 99 61	f 17 2b 04 7e ba 77 d6 26 e1 69 14 63 55 21 0c 7d
0 1 2 3 4 5 6 7 8 9 a b c d e f																		
0 52 09 6a d5 30 36 a5 38 bf 40 a3 9e 81 f3 d7 fb																		
1 7c e3 39 82 9b 2f ff 87 34 8e 43 44 c4 de e9 cb																		
2 54 7b 94 32 a6 c2 23 3d ee 4c 95 0b 42 fa c3 4e																		
3 08 2e a1 66 28 d9 24 b2 76 5b a2 49 6d 8b d1 25																		
4 72 f8 f6 64 86 68 98 16 d4 a4 5c cc 5d 65 b6 92																		
5 6c 70 48 50 fd ed b9 da 5e 15 46 57 a7 8d 9d 84																		
6 90 d8 ab 00 8c bc d3 0a f7 e4 58 05 18 b3 45 06																		
7 d0 2c 1e 8f ca 3f 0f 02 c1 af bd 03 01 13 8a 6b																		
8 3a 91 11 41 4f 67 dc ea 97 f2 cf ce f0 b4 e6 73																		
9 96 ac 74 22 e7 ad 35 85 e2 f9 37 e8 1c 75 df 6e																		
a 47 f1 1a 71 1d 29 c5 89 6f b7 62 0e aa 18 be 1b																		
b fc 56 3e 4b c6 d2 79 20 9a db c0 fe 78 cd 5a f4																		
c 1f dd a8 33 88 07 c7 31 b1 12 10 59 27 80 ec 5f																		
d 60 51 7f a9 19 b5 4a 0d 2d e5 7a 9f 93 c9 9c ef																		
e a0 e0 3b 4d ae 2a f5 b0 c8 eb bb 3c 83 53 99 61																		
f 17 2b 04 7e ba 77 d6 26 e1 69 14 63 55 21 0c 7d																		

2.1.7 Hàm InvMixColumns()

Hàm này là hàm ngược của hàm MixColumns(). Hàm làm việc trên các cột của mảng trạng thái, coi mỗi cột như là một đa thức 4 hạng tử. Các cột được xem là các đa thức trên $GF(2^8)$ và được nhân theo modulo $x^4 + 1$ với một đa thức cố định là $a^{-1}(x)$

$$a^{-1} = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$$

Và có thể mô tả bằng phép nhân ma trận như sau:

$$S'(x) = a^{-1}x \otimes S(x)$$

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

Trong đó $0 \leq c < Nb$.

Kết quả là bốn byte trong mỗi cột sẽ được thay thế theo công thức sau :

$$\begin{aligned} S'_{0,c} &= (\{0e\} \bullet S_{0,c}) \oplus (\{0b\} \bullet S_{1,c}) \oplus (\{0d\} \bullet S_{2,c}) \oplus (\{09\} \bullet S_{3,c}) \oplus \\ S'_{1,c} &= (\{09\} \bullet S_{0,c}) \oplus (\{0e\} \bullet S_{1,c}) \oplus (\{0b\} \bullet S_{2,c}) \oplus (\{0d\} \bullet S_{3,c}) \oplus \\ S'_{2,c} &= (\{0d\} \bullet S_{0,c}) \oplus (\{09\} \bullet S_{1,c}) \oplus (\{0e\} \bullet S_{2,c}) \oplus (\{0b\} \bullet S_{3,c}) \oplus \\ S'_{3,c} &= (\{0b\} \bullet S_{0,c}) \oplus (\{0d\} \bullet S_{1,c}) \oplus (\{09\} \bullet S_{2,c}) \oplus (\{0e\} \bullet S_{3,c}) \oplus \end{aligned}$$

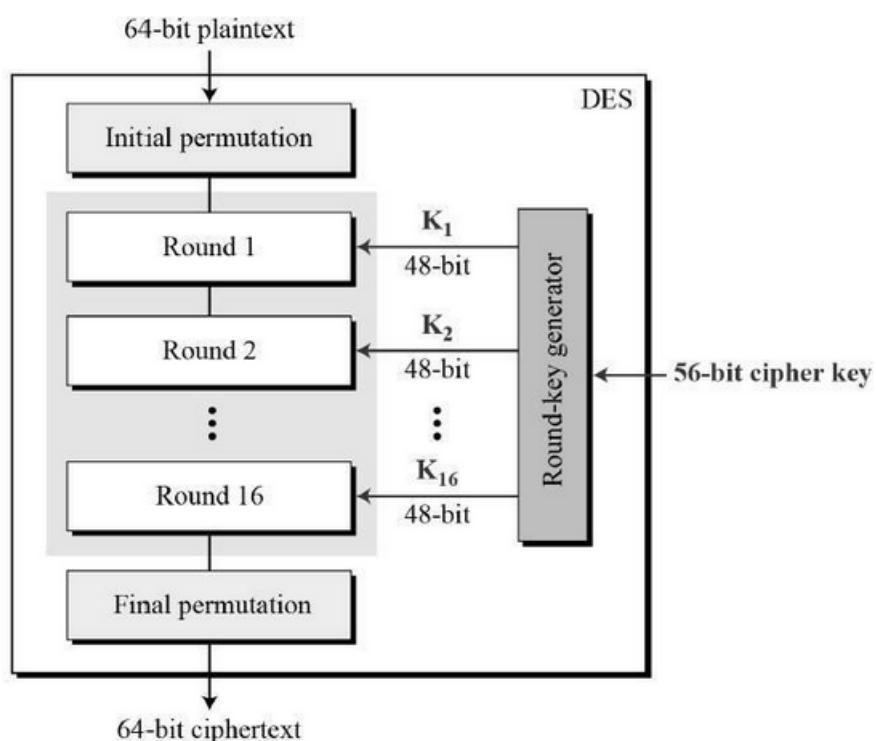
2.1.8 Hàm nghịch đảo của hàm AddRoundKey()

Thật thú vị là hàm này tự bản thân nó là nghịch đảo của chính nó là do hàm chỉ có phép toán XOR bit.

2.2 Thuật toán mã hóa 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.

Mô tả giải thuật



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 mặt 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: 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.

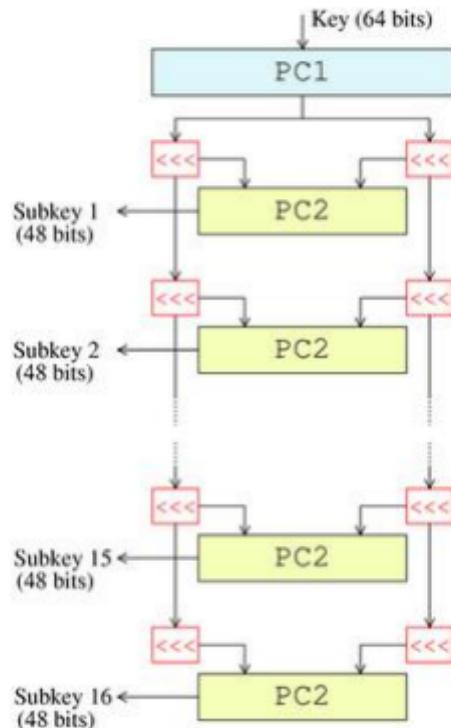
- **Hàm Feistel (F):**

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 bit 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 :**



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

2.3 Thuật toán mã hóa 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 điều kiện độ dài khóa đủ lớn .

- **Tổng thể:**

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 .

- **Tạo Khóa:**

Gọi p và q là hai số nguyên tố lớn ngẫu nhiên phân biệt.

Modun n là tích của hai số nguyên tố này:

$$n = pq$$

Hàm phi Euler (Euler's totient function) của n cho bởi:

$$\phi(n) = (p - 1)(q - 1)$$

Chọn một số $1 < e < \phi(n)$ sao cho:

$$gcd(e, \phi(n)) = 1$$

và tính d với công thức:

$$d = e^{-1} mod \phi(n)$$

Việc mã hóa được thực hiện bằng cách tính:

$$C = M^e \pmod{n}$$

với M là plaintext, C là ciphertext tương ứng của M.

Từ C, M được tính bằng công thức:

$$M = C^d \pmod{n}$$

2.4 Hàm băm MD-5

MD5 chuyển một đoạn thông tin chiều dài thay đổi thành một kết quả chiều dài không đổi 128 bit. Mẫu tin đầu vào được chia thành từng đoạn 512 bit; mẫu tin sau đó được độn sao cho chiều dài của nó chia chẵn cho 512. Công việc độn vào như sau: đầu tiên một bit đơn, 1, được gắn vào cuối mẫu tin. Tiếp theo là một dãy các số zero sao cho chiều dài của mẫu tin lên tới 64 bit ít hơn so với bội số của 512. Những bit còn lại được lấp đầy bằng một số nguyên 64-bit đại diện cho chiều dài của mẫu tin gốc.

3 Nội dung công việc đã làm.

3.1 Thuật toán mã hóa AES

Cơ bản bao gồm các hàm mã hóa-giải mã và hàm sinh key

1. *private byte[] Encrypt(byte[] clearData, byte[] Key, byte[] IV)*: mã hóa một mảng byte dữ liệu vào bằng key và IV.
2. *private void AESAlgorithm(String inputFile, String outputFile, String keys, bool isEncrypt, String mode)*
Đây là hàm dùng để mã hóa và giải mã một file bất kỳ. Với các thông số truyền vào là đường dẫn file input, chuỗi khóa (key), mode có thể là các chế độ như : ECB,CBC,CFB và thông số isEncrypt=true nếu đây là quá trình mã hóa và ngược lại.
3. Và một số hàm hỗ trợ khác.
4. *VprivatevoidbtnGenerateKeyClick(object sender, EventArgs e)*

3.2 Thuật toán mã hóa RSA

Gồm 2 hàm quan trọng nhất là encrypt và decrypt:

Đầu tiên chúng ta dùng hàm ConvertFileByte() để chuyển file cần mã hóa thành dạng byte():

```
public static byte[] ConvertFileToByte(string _FileName){...}
```

Dùng công cụ để chuyển mảng byte đó thành dạng hexadecimal string, sau đó dùng hàm encrypt() để mã hóa thuật toán RSA.

```
public string encrypt(string FileToEncrypt)
```

Ở đây, hàm BigMod() có vai trò mã hóa theo công thức: File mã hóa xong có định dạng txt chứa thông tin của file cần mã hóa dưới dạng các chuỗi hexadeciml string.

Hàm decrypt có nhiệm vụ giải mã theo thuật toán RSA, trong đó hàm BigMod() ở đây có nhiệm vụ giải mã theo công thức: $m = c^d \bmod N$

```
public string decrypt(string FileToEncrypt)
```

Sau đó, ta dùng hàm ConvertByteToFile() để chuyển mảng Byte thành File như ban đầu.

Ngoài ra, ta còn sử dụng hàm hỗ trợ auto_prime_number() để sinh khóa tự động, giảm thời gian cho việc thao tác.

3.3 Thuật toán mã hóa DES

Chứa cả 2 phần mã hóa và giải mã, dữ liệu đầu vào là file input, key với độ dài 64bit, mode ở đây có thể là ECB,CBC,CFB. isEncrypt có kiểu bool nếu isEncrypt = True là giải mã, ngược lại thì mã hóa.

```
void DESAlgorithm(string sInputFilename, string sOutputFilename, string sKey, string mode, bool isEncrypt)
```

3.4 Hàm băm MD5

Nhóm đã sử dụng thư viện mã hóa trong C# là : *System.Security.Cryptography*
Hàm chính để mã hóa dữ liệu :

1. *private string GetMD5HashData(string data)*: nhận vào chuỗi dữ liệu và trả về chuỗi string có chiều dài nhất định.
2. *private string openfile()*: hàm phân tích file thành các blocksize trả về giá chuỗi string là đầu vào của hàm GetMD5HashData

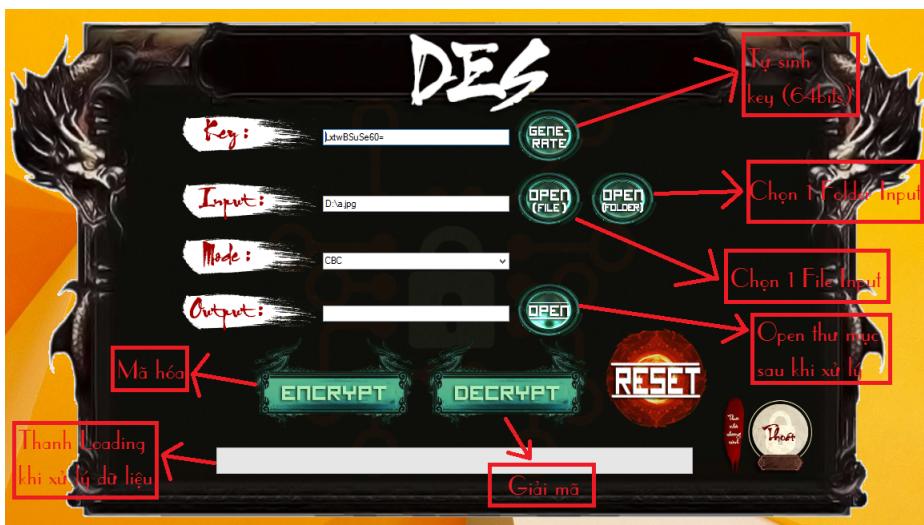
4 Demo chương trình

Giao diện chính của chương trình.



4.1 Giải thuật mã hóa DES (DES)

Giao diện chính gồm 2 phần Encrypt và Decrypt.



4.1.1 Phần mã hóa

Phần Input File : có thể mã hóa hầu hết tất cả các file như: hình ảnh, âm thanh, video, pdf, các file dạng văn bản... Còn có thể mã hóa tập tin nén dạng zip, thư mục, rar.

Phần Input Key : có thể nhập key bất kỳ có chiều dài 64bit(8byte) hoặc chọn Generate để random một key bất kì.

Phần Output File: mật định chọn nơi lưu file ciphertext là nơi chọn file plaintext để mã hóa, file ciphertext có đuôi .tpEn

Ấn Encrypt để tiến hành mã hóa.

Giao diện sau khi mã hóa thành công.



4.1.2 Phần giải mã

Phần Input File: chọn file có đuôi .tpEn để giải mã.

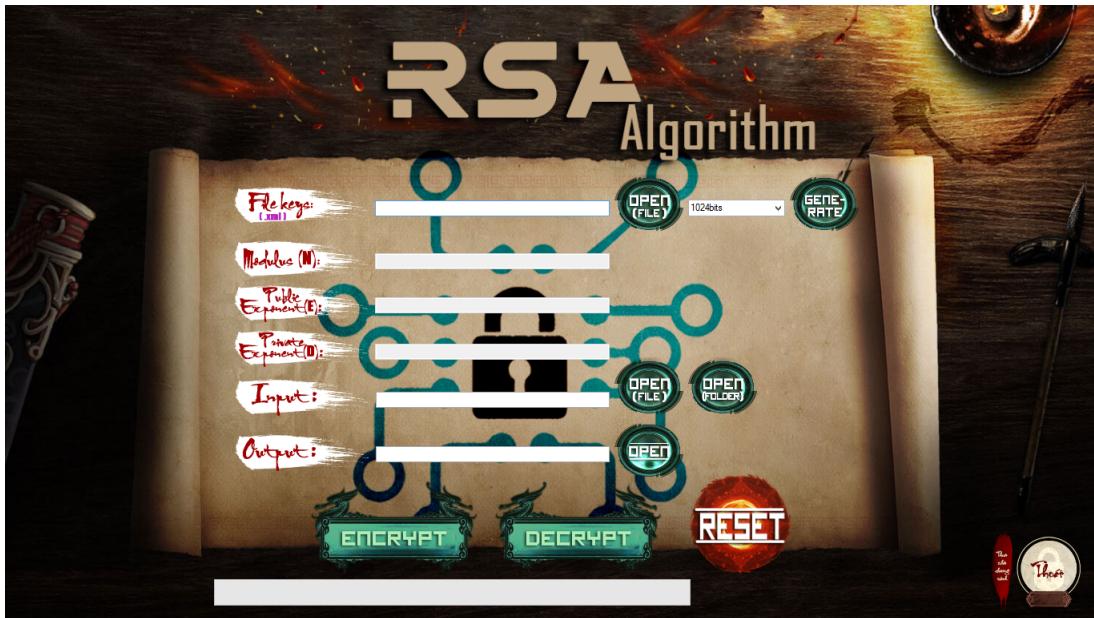
Phần Input Key: nhập key ở phần mã hóa.

Phần Output File: xuất ra file plaintext ban đầu.

Ấn Decrypt để tiến hành giải mã.

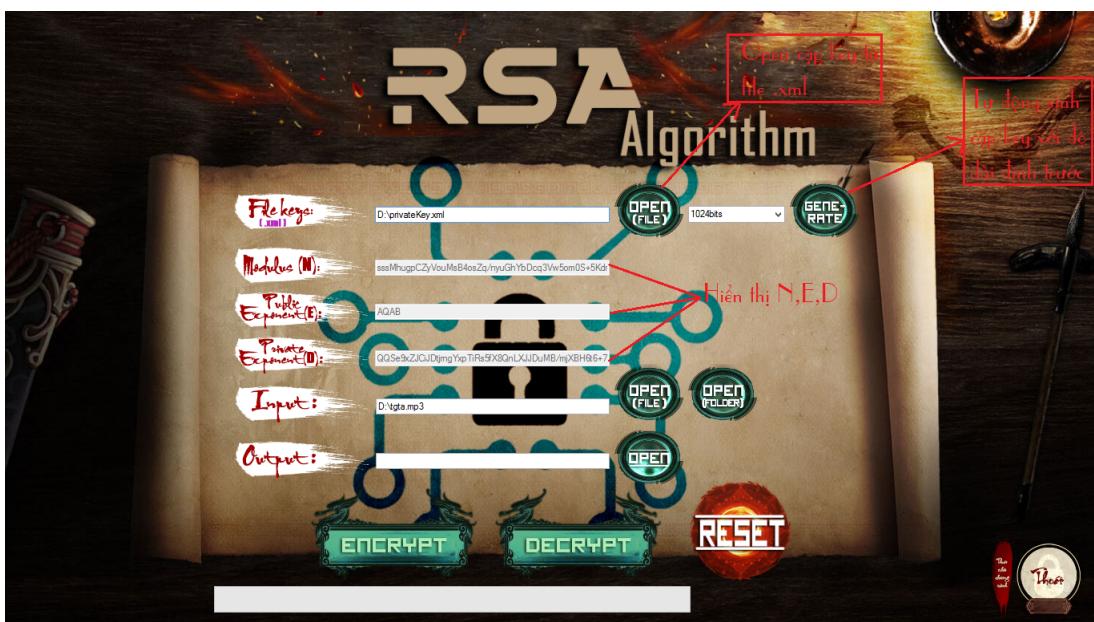
4.2 Thuật toán mã hóa RSA

Giao diện chính của giải thuật RSA .



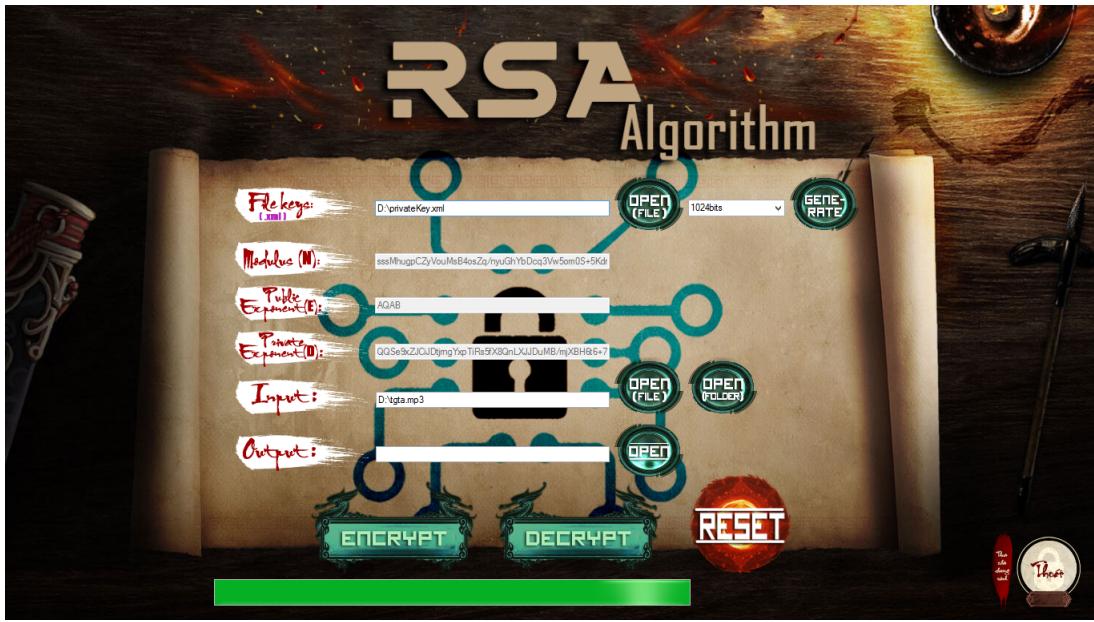
4.2.1 Phần mã hóa

Mô tả tính năng :



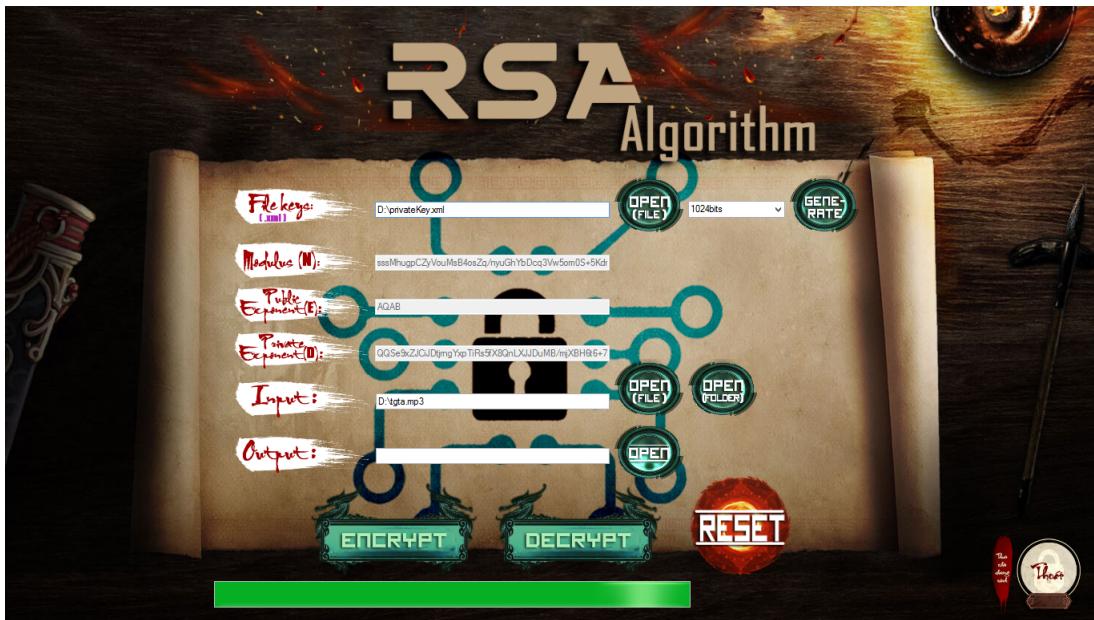
Ngay File key nhấn vào open để mở tập file key từ file .xml hoặc nhấn generate để sinh ra file key định dạng .xml. Sau đó chọn file input để mã hóa.

Giao diện sau khi mã hóa.



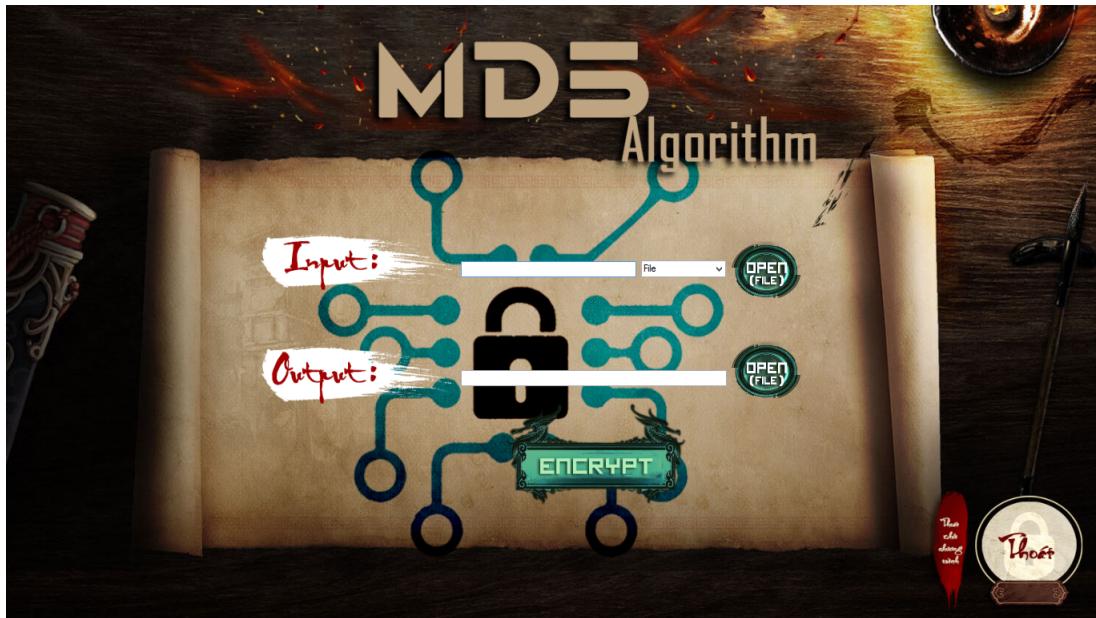
4.2.2 Phần giải mã

Tương tự như phần mã hóa. Chỉ khác ngay phần input là định dạng file .tpEn



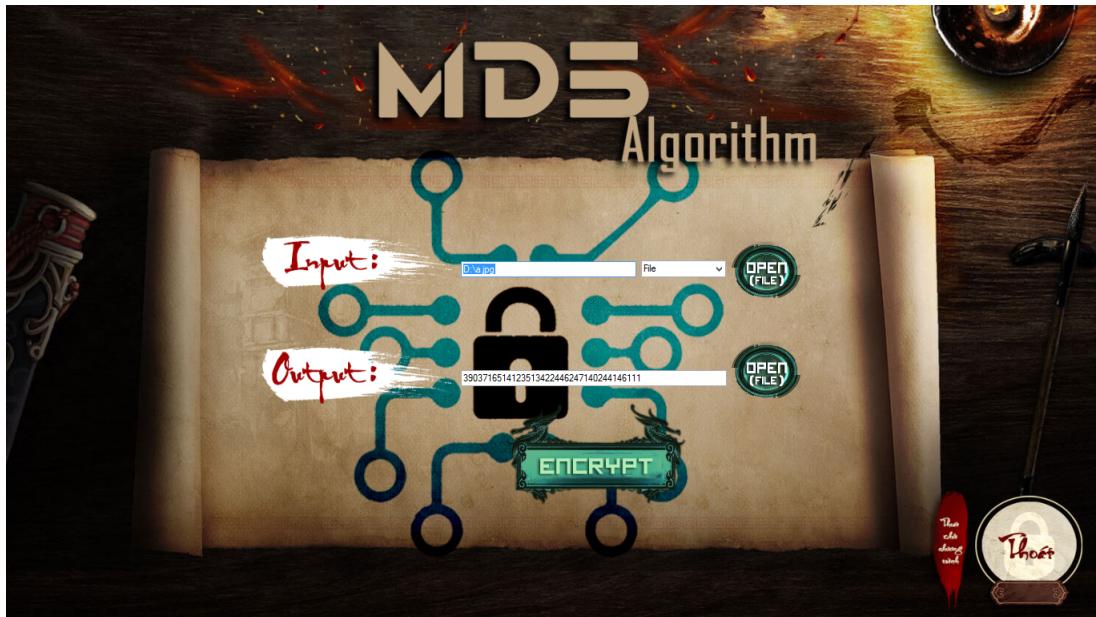
4.3 Hàm băm MD-5

Giao diện chính gồm 2 phần File và Text:



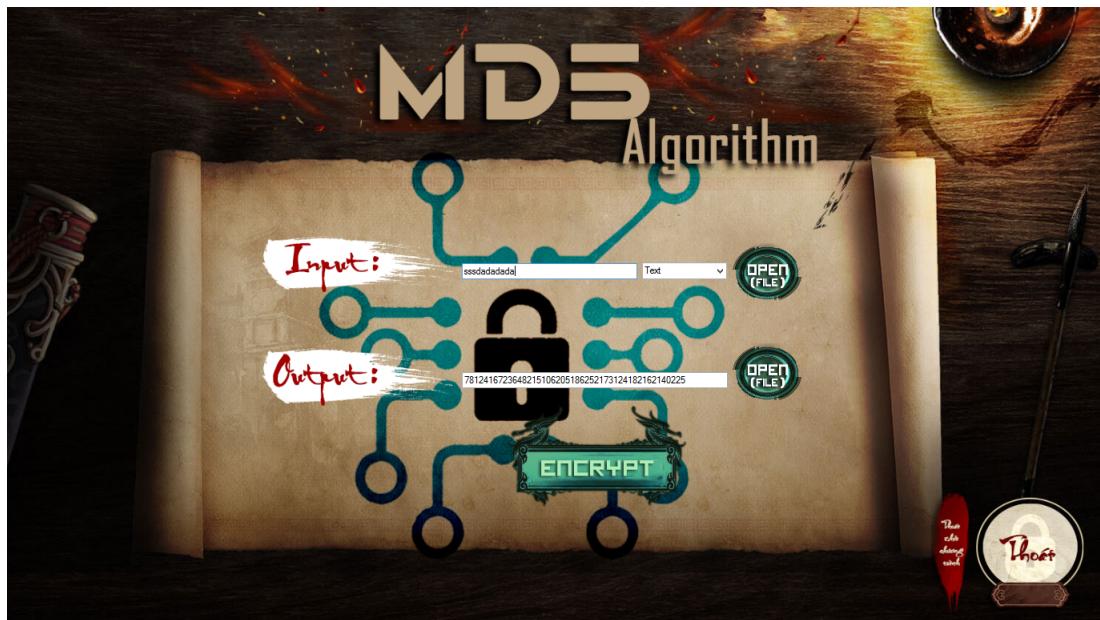
4.3.1 Phân file

Phân Input Path: có thể mã hóa hầu hết tất cả các file như: hình ảnh, âm thanh, video, pdf, các file dạng văn bản... Còn có thể mã hóa tập tin nén dạng zip,rar,thư mục...



Phân Output Path: giá trị sao khi băm.

Ấn Encrypt để bắt đầu mã hóa



4.3.2 Phần text

Phần Input Path: Nhập vào đoạn text, hay một key cần mã hóa

Phần Output Path: giá trị sao khi mã hóa.

Ấn Encrypt để bắt đầu mã hóa.

4.4 Giải thuật mã hóa AES

Giao diện chính gồm 2 phần Encrypt và Decrypt:



4.4.1 Phần mã hóa

Mô tả tính năng



Phần Input File : có thể mã hóa hầu hết tất cả các file như: hình ảnh, âm thanh, video, pdf, các file dạng văn bản... Còn có thể mã hóa tập tin nén dạng zip, thư mục, rar...

Phần Input Key : có thể nhập key bất kỳ hoặc chọn Generate key(từ độ dài định trước) để random một key bất kì.

Phần Output File: mặc định chọn nơi lưu file ciphertext là nơi chọn file plaintext để mã hóa, file ciphertext có đuôi .tpEn

Nhấn Encrypt để bắt đầu mã hóa.

4.4.2 Phần giải mã

Phần Input File: chọn file có đuôi.tpEn để giải mã.

Phần Input Key: nhập key ở phần mã hóa.

Phần Output File: xuất ra file plaintext ban đầu.

Ấn Decrypt để tiến hành giải mã.

5 Phân tích và kết luận

5.1 Phân tích

- Giải thuật DES : mã hóa thành công tất cả các file dựa trên việc chia các file thành block nhỏ.
- Giải thuật RSA : mã hóa thành công các file có kích thước nhỏ, và xử lý khóa có chiều dài ngắn.
- Hàm băm MD-5 : mã hóa thành công hầu hết các file và các đoạn text.
- Giải thuật AES : mã hóa thành công tất cả các file.

5.2 Kết luận

- Kết quả đạt được:

Mã hóa và giải mã thành công các file có kích thước vừa và nhỏ.

- Hạn chế:

Thời gian mã hóa và giải mã các file có kích thước lớn khá lâu.

Ở giải thuật RSA, khi chọn key có kích thước lớn (ví dụ: 4096) thì tốc độ mã hóa chậm.

6 Hướng phát triển

Cải thiện thời gian mã hóa cũng như giải mã các tập tin có kích thước lớn của các giải thuật nêu trên.

Ở giải thuật RSA, tìm kiếm biện pháp giải quyết vấn đề khi sử dụng key có kích thước lớn để mã hóa tập tin.(ví dụ dùng multi-threading)

Cải thiện giao diện chương trình...

7 Tài liệu tham khảo

[https://vi.wikipedia.org/wiki/AES_\(m%C3%A3_h%C3%B3a\)](https://vi.wikipedia.org/wiki/AES_(m%C3%A3_h%C3%B3a))

<https://vi.wikipedia.org/wiki/MD5>

[https://vi.wikipedia.org/wiki/RSA_\(m%C3%A3_h%C3%B3a\)](https://vi.wikipedia.org/wiki/RSA_(m%C3%A3_h%C3%B3a))

[https://en.wikipedia.org/wiki/DataEncryptionStandard](https://en.wikipedia.org/wiki/Data_Encryption_Standard)