

Họ và Tên	Nội dung công việc	Đóng góp (%)
Trần Văn Lắm	-Tìm hiểu các thuật toán mã hóa mà nhóm thực hiện. -Hiện thực thuật toán mã hóa S-DES và MD-5. -Tổng hợp chương trình	100%
Trịnh Văn Quyền	-Tìm hiểu các thuật toán mã hóa mà nhóm thực hiện. -Hiện thực thuật toán mã hóa AES. -Viết báo cáo	100%
Nguyễn Hữu Nam	-Tìm hiểu các thuật toán mã hóa mà nhóm thực hiện. -Hiện thực thuật toán mã hóa RSA -Viết báo cáo	100%

## Mục lục

---

<b>1</b>	<b>Môi trường thực hiện.</b>	<b>3</b>
<b>2</b>	<b>Giới thiệu tổng quan về các giải thuật.</b>	<b>3</b>
2.1	Giải thuật mã hóa Simplified-DES (S-DES) . . . . .	3
2.2	Thuật toán mã hóa RSA . . . . .	3
2.3	Hàm băm MD-5 . . . . .	4
2.4	Thuật toán mã hóa AES (Advanced Encryption Standard) . . . . .	4
2.4.1	Hàm SubBytes() . . . . .	6
2.4.2	Hàm ShiftRows() . . . . .	7
2.4.3	Hàm MixColumns() . . . . .	7
2.4.4	Hàm AddRoundKey() . . . . .	8
2.4.5	Hàm InvShiftRows() . . . . .	8
2.4.6	Hàm InvSubbytes() . . . . .	9
2.4.7	Hàm InvMixColumns() . . . . .	9
2.4.8	Hàm nghịch đảo của hàm AddRoundKey() . . . . .	10
<b>3</b>	<b>Nội dung công việc đã làm.</b>	<b>11</b>
3.1	Giải thuật mã hóa Simplified-DES (S-DES) . . . . .	11
3.2	Thuật toán mã hóa RSA . . . . .	12
3.3	Hàm băm MD-5 . . . . .	12
3.4	Thuật toán mã hóa AES . . . . .	12
<b>4</b>	<b>Demo chương trình</b>	<b>14</b>
4.1	Giải thuật mã hóa Simplified-DES (S-DES) . . . . .	14
4.1.1	Phần mã hóa . . . . .	14
4.1.2	Phần giải mã . . . . .	15
4.2	Thuật toán mã hóa RSA . . . . .	16
4.2.1	Phần mã hóa . . . . .	16
4.2.2	Phần giải mã . . . . .	17
4.3	Hàm băm MD-5 . . . . .	18
4.3.1	Phần file . . . . .	18
4.3.2	Phần text . . . . .	18
4.4	Giải thuật mã hóa AES . . . . .	19
4.4.1	Phần mã hóa . . . . .	19
4.4.2	Phần giải mã . . . . .	20
<b>5</b>	<b>Phân tích và kết luận</b>	<b>20</b>
5.1	Phân tích . . . . .	20
5.2	Kết luận . . . . .	20
<b>6</b>	<b>Hướng phát triển</b>	<b>21</b>
<b>7</b>	<b>Tài liệu tham khảo</b>	<b>21</b>

## 1 Môi trường thực hiện.

---

- Môi trường thực hiện: Ngôn ngữ C#.
- Tiến trình thực hiện:
  - Tìm hiểu cách thức hoạt động của các giải thuật S-DES, RSA, AES, MD-5.
  - Kết hợp với thư viện có sẵn trong C# để hiện thực lại các giải thuật trên.

## 2 Giới thiệu tổng quan về các giải thuật.

---

### 2.1 Giải thuật mã hóa Simplified-DES (S-DES)

Simplified DES (S-DES), được phát triển bởi GS. Edward Schaefer trường đại học Santa Clara. Giải thuật mã hóa S-DES nhận dữ liệu đầu vào trên bản rõ theo khối 8-bit (ví dụ: 10111101) và khóa 10 bit để tạo ra dữ liệu đầu ra là bản mã theo khối 8-bit. Ngược lại giải thuật giải mã S-DES nhận dữ liệu đầu vào bản mã theo khối 8-bit và khóa 10-bit để tạo ra dữ liệu đầu ra là bản rõ theo khối 8-bit.

Giải thuật mã hóa gồm 5 chức năng:

- Hàm initial permutation(IP)
- Một hàm phức tạp gọi là fK với tham số K1 sinh ra từ khóa "key"
- Một hàm hoán vị đơn giản (SW);
- Hàm fK một lần nữa với tham số K2 sinh ra từ khóa "key"
- Một hàm hoán vị là nghịch đảo của hàm IP gọi là  $IP^{-1}$

Có thể biểu diễn giải thuật mã hóa bằng hàm hợp như sau:

$$IP^{-1} \cdot f_{K_2} \cdot SW \cdot f_{K_1} \cdot IP$$

Hoặc dạng sau:

$$\text{ciphertext} = IP^{-1}(f_{K_2}(SW(f_{K_1}(IP(\text{plaintext}))))))$$

với:

$$K_1 = P8(Shift(P10(key)))$$

### 2.2 Thuật toán mã hóa RSA

Thuật toán RSA được đề xuất bởi Rivest, Shamir và Adleman.

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:

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

Việc mã hóa được thực hiện bằng cách tính:

$$C = M^e \bmod n$$

với  $M$  là plaintext,  $C$  là ciphertext tương ứng của  $M$ .  
 Từ  $C$ ,  $M$  được tính bằng công thức:

$$M = C^d \bmod n$$

### 2.3 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 chắn 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.

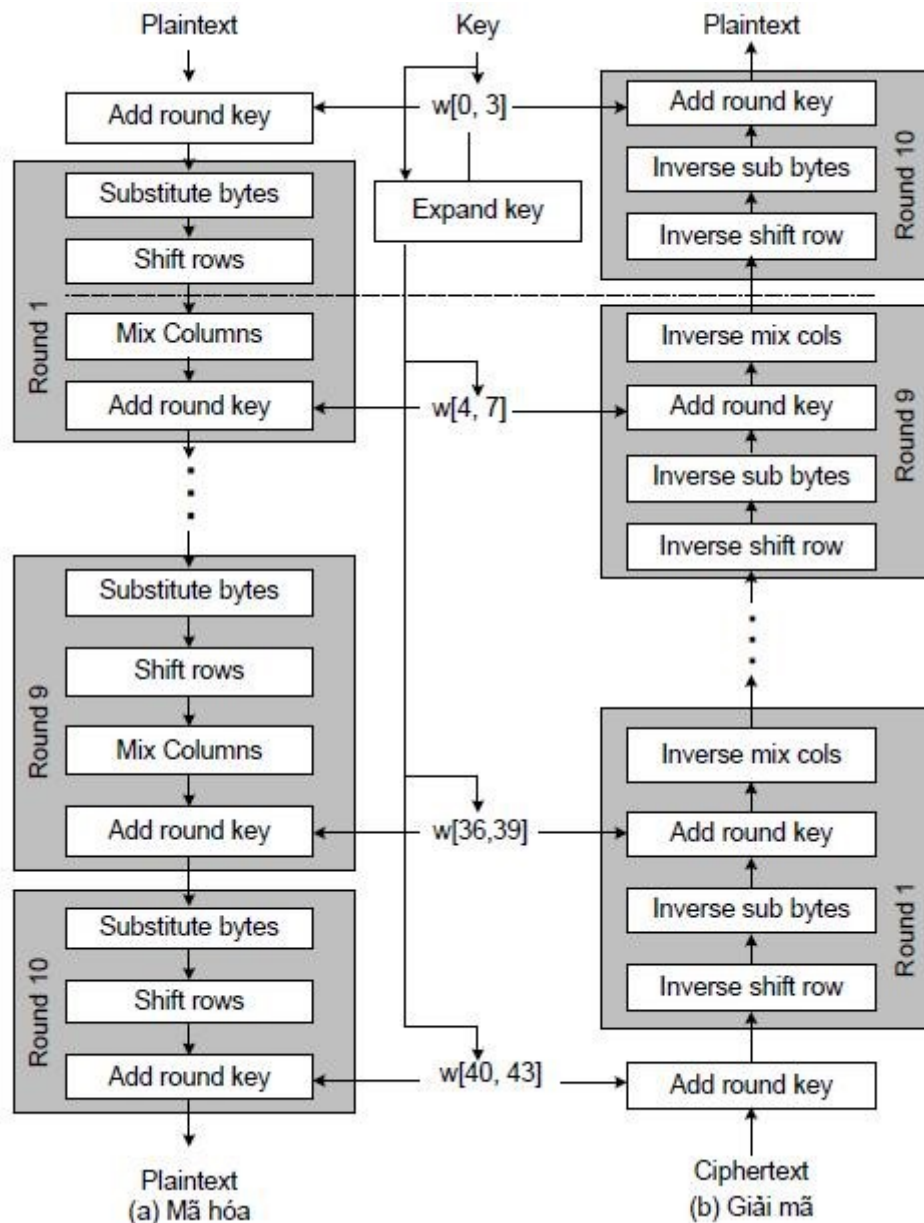
### 2.4 Thuật toán mã hóa AES (Advanced Encryption Standard)

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 vắn tắt là trạng thái (tiếng Anh: state, trạng 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 thế (phi tuyến) trong đó mỗi byte trong trạng thái sẽ được thế 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

### 2.4.1 Hàm SubBytes()

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

- Nhân 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)):

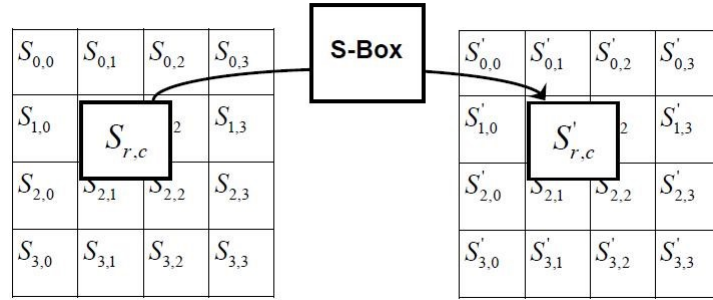
$$b'_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i$$

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:

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	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 chẳng 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  $S'_{1,1} = \{ed\}$

#### 2.4.2 Hàm ShiftRows()

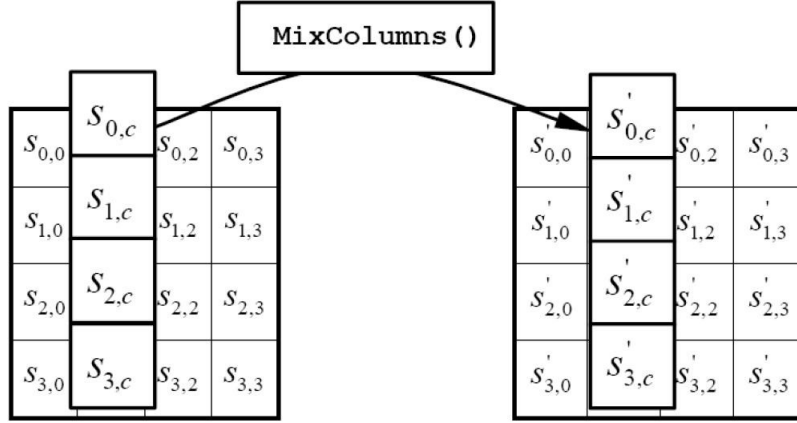
Là phép biến đổi các byte trên ba hàng cuối cùng của mảng trạng thái bằng cách dịch vòng thể hiện qua công thức:

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

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.

#### 2.4.3 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.



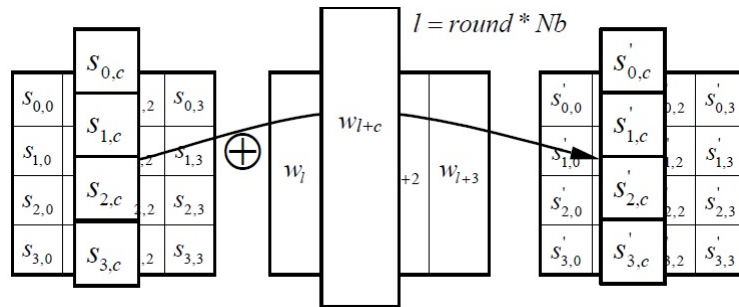
#### 2.4.4 Hàm AddRoundKey()

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

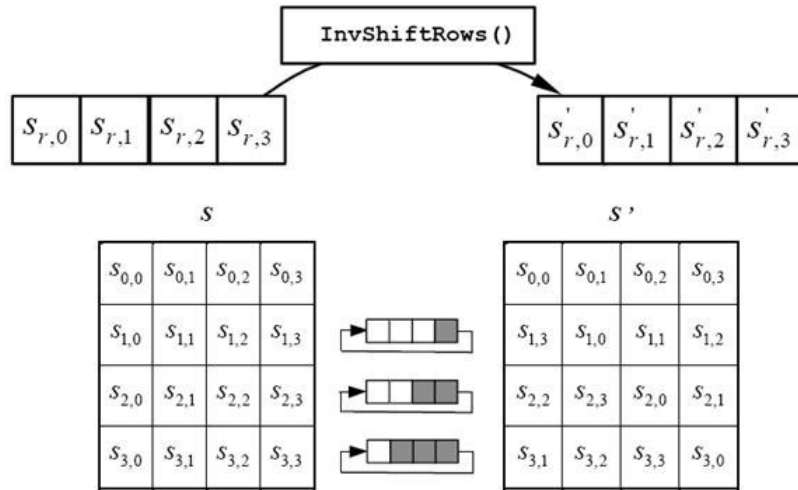
#### 2.4.5 Hàm InvShiftRows()

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:

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





#### 2.4.6 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  $GF(2^2)$

Bảng thể được sử dụng trong hàm:

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	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	b8	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

#### 2.4.7 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 :

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

#### 2.4.8 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.

### 3 Nội dung công việc đã làm.

---

#### 3.1 Giải thuật mã hóa Simplified-DES (S-DES)

Gồm 2 hàm chính là encrypt và decrypt :

- *public byte* Encrypt( *byte* block): mã hóa từng block input plaintext được đưa vào, và xuất ra block ciphertext
- *public byte* Decrypt( *byte* block): giải mã các block ciphertext thành các block plaintext

Các hàm mã hóa chính được định nghĩa:

1. *BitArray* P10(*BitArray* key): Hàm hoán vị P10 được định nghĩa:

$$P10(k1, k2, k3, k4, k5, k6, k7, k8, k9, k10) = (k3, k5, k2, k7, k4, k10, k1, k9, k8, k6)$$

2. *BitArray* P8(*BitArray* part1, *BitArray* part2): hàm P8 để lấy ra và hoán vị 8 trong số 10 bit: P8(k6, k3, k7, k4, k8, k5, k10, k9).
3. *BitArray* P4(*BitArray* part1, *BitArray* part2): hàm P4 là hàm biến đổi giá trị đầu ra của S-box S0 và S1: P4(k2, k4, k3, k1).
4. *BitArray* IP(*BitArray* plainText) và *BitArray* RIP(*BitArray* permutedText): IP(p2, p6, p3, p1, p4, p8, p5, p7) và RIP là hàm nghịch đảo của hàm IP, RIP(p4, p1, p3, p5, p7, p2, p8, p6).
5. *BitArray* Fk(*BitArray* IP, *BitArray* key): Thành phần phức tạp nhất trong S-DES là hàm fk, gồm nhiều hàm permutation và substitution kết hợp với nhau. Hàm fk được mô tả như sau. Gọi L và R là phần 4-bit trái và phần 4-bit phải của dữ liệu đầu vào fk 8-bit; gọi F là một ánh xạ (không bắt buộc là ánh xạ 1:1) từ chuỗi 4-bit đến chuỗi 4-bit:

$$f_k(L, R) = (L \oplus F(R, SK), R)$$

với SK là khóa con (subkey) và  $\oplus$  là hàm exclusive-OR trên bit.

6. *BitArray* S\_Boxes(*BitArray* input, int no): Hàm S-box hoạt động như sau: bit thứ nhất và bit thứ tư được dùng là 2-bit xác định dòng của S-box, và bit thứ hai và bit thứ ba xác định cột của S-box.

Và một số hàm chuyển đổi cần thiết khác.

### 3.2 Thuật toán mã hóa RSA

Gồm 2 hàm quan trọng nhất là hàm encrypt và hàm decrypt:

Đầu tiên, chúng ta sử dụng hàm ConvertFileToByte() 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 theo 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 hexadecimal 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 Hàm băm MD-5

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

### 3.4 Thuật toán mã hóa AES

Bao gồm các hàm mã hóa sinh key và giải mã:

1. *private byte[]* Encrypt(*byte[]* clearData, *byte[]* Key, *byte[]* IV): mã hóa một mảng byte dữ liệu vào bằng key và IV.
2. *public byte[]* Encrypt(*byte[]* Data, *string* Password, *int* Bits): hàm trả về ciphertext khi nhận vào mảng plaintext và key và độ dài của key.
3. *private byte[]* Decrypt(*byte[]* cipherData, *byte[]* Key, *byte[]* IV) : giải mã một mảng byte dữ liệu vào bằng key và IV.

4. *public byte[] Decrypt(byte[] Data, string Password, int Bits)*: hàm trả về plaintext khi nhận vào mảng byte ciphertext và key và độ dài của key.  
Trong đó hàm:

```
PasswordDeriveBytes pdb = new PasswordDeriveBytes(Password, new byte[]  
0x00, 0x01, 0x02, 0x1C, 0x1D, 0x1E, 0x03, 0x04, 0x05, 0x0F, 0x20, 0x21, 0xAD,  
0xAF, 0xA4 )
```

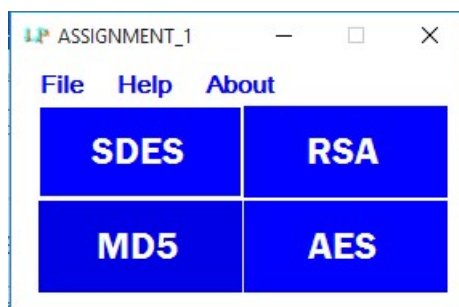
khởi tạo key có độ dài Bits.

Và một số hàm hỗ trợ khác.

## 4 Demo chương trình

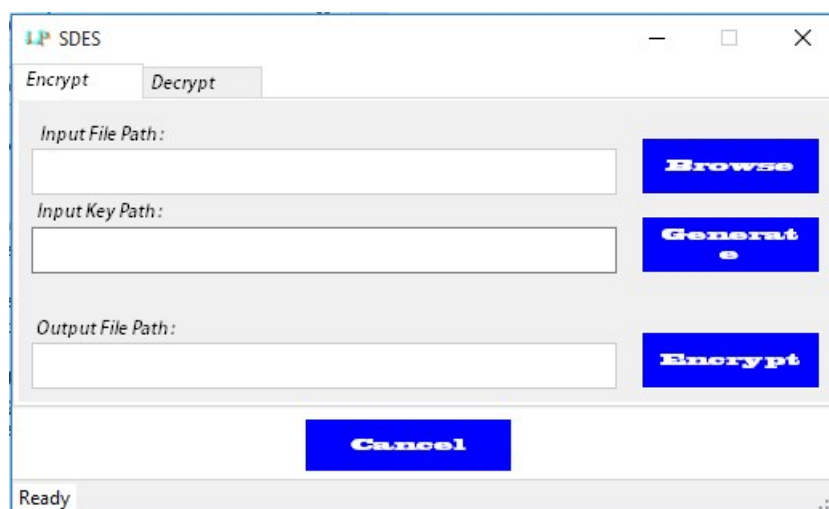
---

Giao diện chính của chương trình.



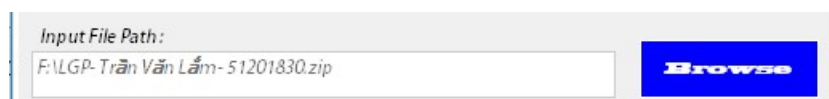
### 4.1 Giải thuật mã hóa Simplified-DES (S-DES)

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,rar...



Phần Input Key : có thể nhập key bất kỳ có chiều dài 10bit hoặc chọn Generate để random một key bất kì

Input Key Path:

1110111110

Generate

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

Output File Path:

F:\LGP-Trần Văn Lắm- 51201830.zip.##

Encrypt

Ấn Encrypt để tiến hành mã hóa

#### 4.1.2 Phần giải mã

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

Encrypt Decrypt

Input File Path:

F:\LGP-Trần Văn Lắm- 51201830.zip.##

Browse

Phần Input Key: nhập key ở phần mã hóa.

Input Key Path:

1110111110

Phần Output File: xuất ra file plaintext ban đầu.

Output File Path:

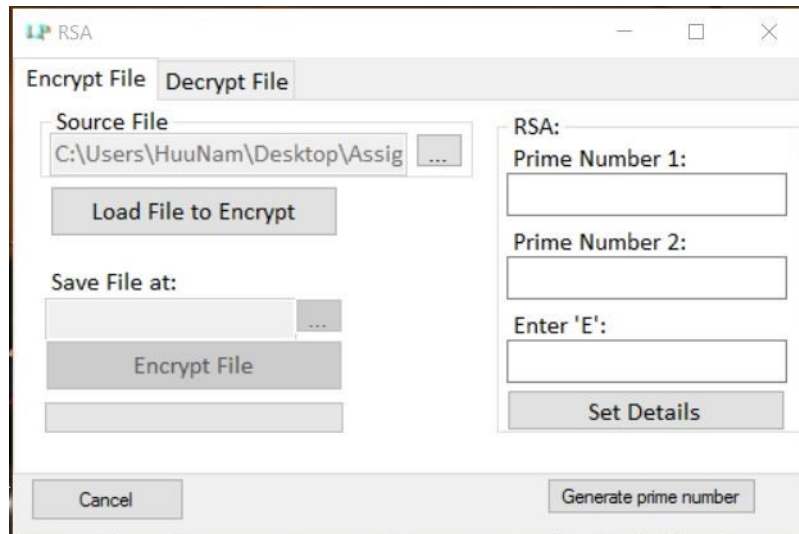
F:\LGP-Trần Văn Lắm- 512018301.zip

Decrypt

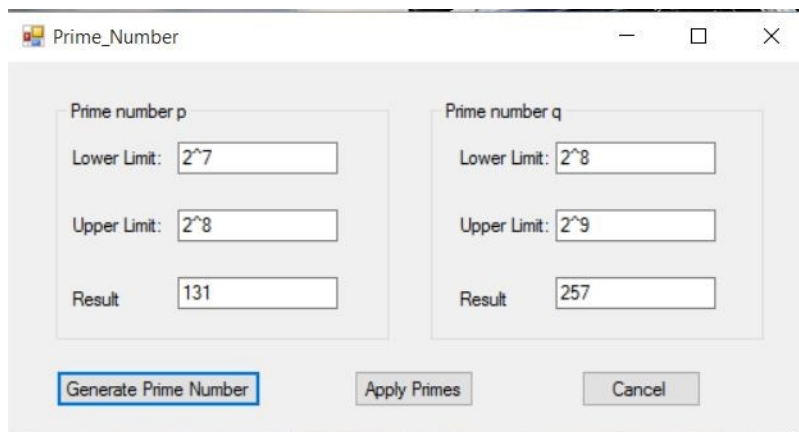
Ấn Decrypt để tiến hành giải mã.

## 4.2 Thuật toán mã hóa RSA

### 4.2.1 Phần mã hóa



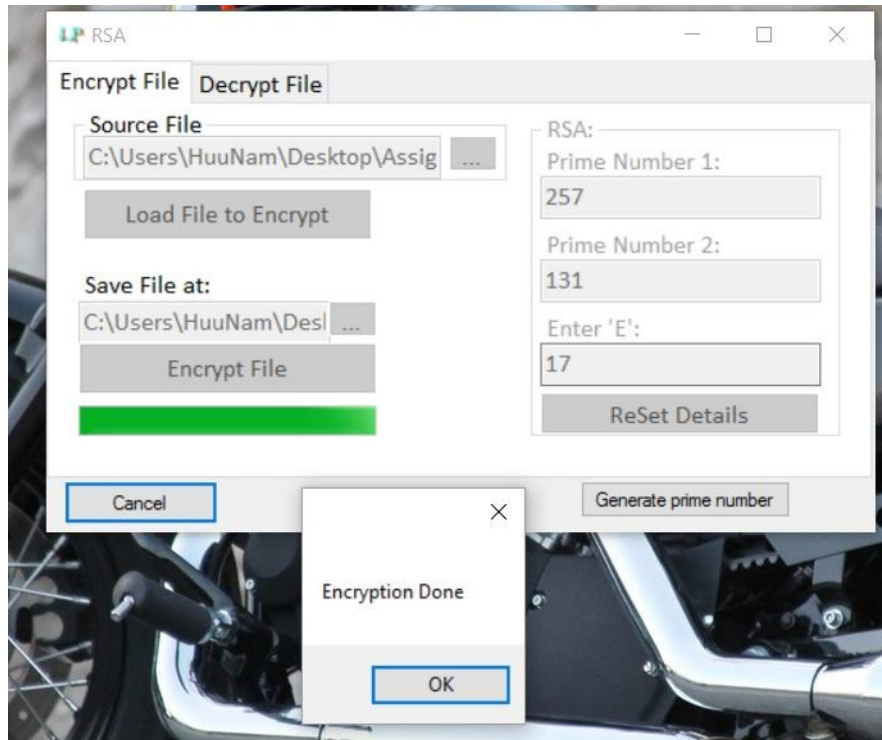
Bấm Generate prime number để tạo p,q tự động, nhập lower và Upper limit sau đó ấn Apply primes.



Ấn load File to Encrypt để chọn file cần mã hóa.

Nhập file mã hóa vào Save File at, rồi nhấn Encrypt File để tiến hành mã hóa. Mã hóa xong xuất hiện bảng thông báo Encrypt Done như hình.



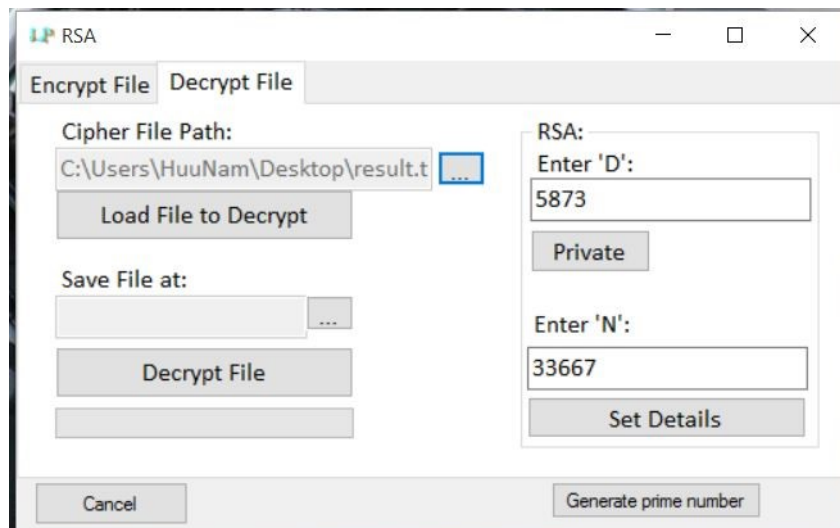


#### 4.2.2 Phần giải mã

Ta nhập địa chỉ Cipher File vào Cipher File Path

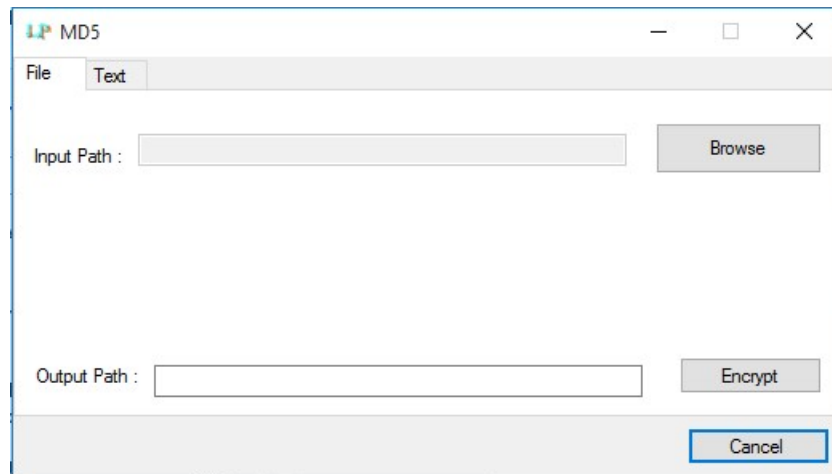
Nhập D và N ( nếu sử dụng Genarate prime number thì ấn Private để tự sinh mã).

Sau đó nhập địa chỉ lưu file vào Save File at và nhấn Decrypt File để tiến hành mã hóa File



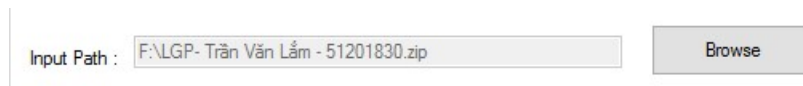
### 4.3 Hàm băm MD-5

Giao diện chính gồm 2 phần File và Text:

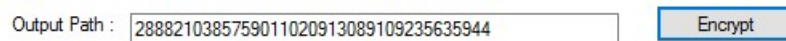


#### 4.3.1 Phần file

Phần 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...



Phần 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



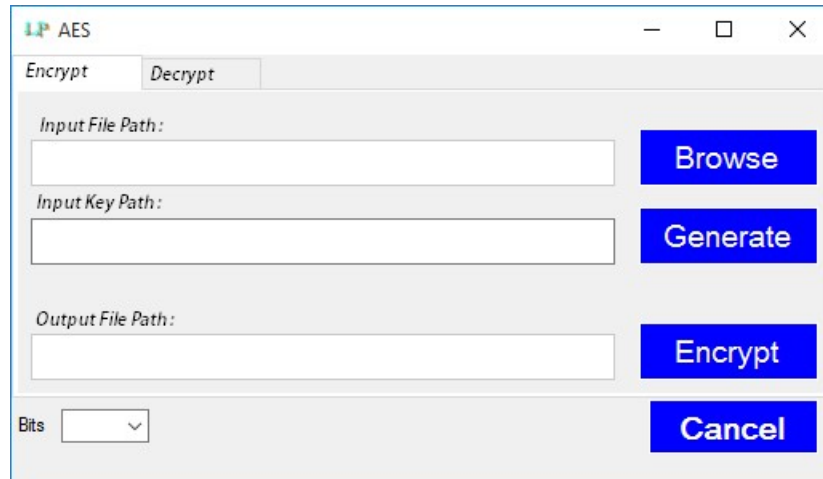
Phần Output Path: giá trị sao khi mã hóa

Output Path :

Ấn Encrypt để bắt đầu mã hóa.

#### 4.4 Giải thuật mã hóa AES

Giao diện chính gồm 2 phần Encrypt và Decrypt:



The screenshot shows the AES application window with the 'Encrypt' tab selected. It contains three input fields: 'Input File Path:', 'Input Key Path:', and 'Output File Path:'. Each field has a 'Browse' button to its right. Below these fields is a 'Bits' dropdown menu and a 'Cancel' button. The 'Encrypt' button is also visible to the right of the 'Output File Path' field.

##### 4.4.1 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,rar...



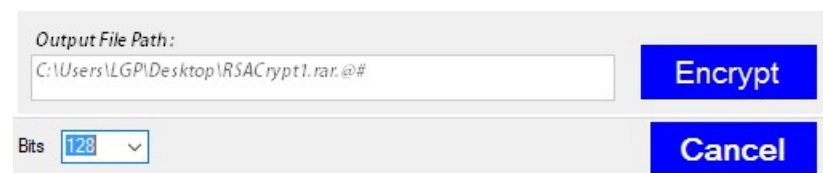
This screenshot shows the 'Input File Path:' section. The text field contains the path 'C:\Users\LGPI\Desktop\RSACrypt.rar'. A 'Browse' button is located to the right of the text field.

Phần Input Key : có thể nhập key bất kỳ hoặc chọn Generate để random một key bất kỳ.



This screenshot shows the 'Input Key Path:' section. The text field contains the key 'aes'. A 'Generate' button is located to the right of the text field.

Phần Output File: mật định chọn nơi lưu file ciphertext là nơi chọn file plaintext để mã hóa, file ciphertext có đuôi .# và chọn chiều dài của key.



This screenshot shows the 'Output File Path:' section. The text field contains the path 'C:\Users\LGPI\Desktop\RSACrypt1.rar.@'. An 'Encrypt' button is to the right. Below the text field, there is a 'Bits' dropdown menu set to '128' and a 'Cancel' button.

Nhấn Encrypt để bắt đầu mã hóa.

#### 4.4.2 Phần giải mã

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



The screenshot shows a software window with two tabs: 'Encrypt' and 'Decrypt'. The 'Decrypt' tab is active. Below the tabs, there is a label 'Input File Path:' followed by a text input field containing the path 'C:\Users\LGPI\Desktop\RSACrypt.rar.@#'. To the right of the input field is a blue button labeled 'Browse'.

Phần Input Key: nhập key ở phần mã hóa.



The screenshot shows a text input field labeled 'Input Key Path:' containing the text 'aes'.

Phần Output File: xuất ra file plaintext ban đầu.



The screenshot shows a software window with a label 'Output File Path:' followed by a text input field containing the path 'C:\Users\LGPI\Desktop\RSACrypt12.rar'. To the right of the input field is a blue button labeled 'Decrypt'. Below the input field, there is a 'Bits' label and a dropdown menu showing '128'. To the right of the dropdown menu is a blue button labeled 'Cancel'.

Ấn Decrypt để tiến hành giải mã.

## 5 Phân tích và kết luận

### 5.1 Phân tích

- Giải thuật S-DES : đã 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 : đã hóa thành công tất cả các file.

### 5.2 Kết luận

- Kết quả đạt được:  
Đã 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, chưa xử lý được key có kích thước lớn, chưa mã hóa được file có kích thước lớn.  
Và một số hạn chế về giao diện. . .

## 6 Hướng phát triển

---

Cải thiện thời gian mã hóa cũng như giải mã các tập tin có kích thước lớn. Ở giải thuật RSA, tìm kiếm biện pháp giải quyết vấn đề mã hóa tập tin có kích thước lớn cũng như sử dụng key có chiều dài cao hơn. 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://www.youtube.com/user/theRSAorg>

<http://homepage.smc.edu>

<http://www.md5.net/>