BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG

ĐỒ ÁN TỐT NGHIỆP

NGÀNH: ĐIỆN TỬ VIỄN THÔNG

Người hướng dẫn : Thạc sỹ Nguyễn Văn Dương

Sinh viên : Lê Thế Trị

HẢI PHÒNG – 2018

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG

XÂY DỰNG CHƯƠNG TRÌNH MÃ HÓA VÀ GIẢI MÃ RSA

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC CHÍNH QUY
NGÀNH: ĐIỆN TỬ VIỄN THÔNG

Người hướng dẫn : Thạc sỹ Nguyễn Văn Dương

Sinh viên : Lê Thế Trị

HẢI PHÒNG – 2018

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG

NHIỆM VỤ ĐỀ TÀI TỐT NGHIỆP

Sinh viên : Lê Thế Trị. Mã sinh viên: 1212103004.

Lớp : ĐT1601. Ngành : Điện tử viễn thông.

Tên đề tài : Xây dựng chương trình mã hóa và giải mã RSA

NHIỆM VỤ ĐỀ TÀI

1. Nội dung và các yêu cầu cần giải quyết trong nhiệm vụ đề tài tốt nghiệp (về lý luận, tính thực tiễn, tính học và các sơ đồ).

.....

.....

.....

.....

.....

.....

.....

.....

2. Các định lý toán học và thuật toán để xây dựng chương trình

.....

.....

.....

.....

.....

.....

.....

3. Địa điểm

.....

.....

.....

.....

CÁN BỘ HƯỚNG DẪN ĐỀ TÀI TỐT NGHIỆP

Người hướng dẫn :

Họ và tên: Nguyên Văn Dương.

Học hàm, học vị: Thạc sỹ.

Cơ quan công tác: Trường Đại học Dân lập Hải Phòng.

Nội dung hướng dẫn:

……
……
……
……
……
……
……
……
……
……
……
……

Đề tài tốt nghiệp được giao ngày ..... tháng ..... năm 2018.

Yêu cầu phải hoàn thành xong trước ngày ..... tháng ..... năm 2018.

Đã nhận nhiệm vụ ĐTTN	Đã giao nhiệm vụ ĐTTN
Sinh viên	Người hướng dẫn

Hải Phòng, ngày ..... tháng ..... năm 2018.

HIỆU TRƯỞNG

GS.TS.NGƯT Trần Hữu Nghị

PHẦN NHẬN XÉT TÓM TẮT CỦA CÁN BỘ HƯỚNG DẪN

1. Tinh thần thái độ của sinh viên trong quá trình làm đề tài tốt nhiệp:

.....

.....

.....

.....

.....

.....

.....

.....

.....

2. Đánh giá chất lượng của đồ án:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

3. Cho điểm của cán bộ hướng dẫn. (Điểm ghi cả chữ và số)

Hải Phòng, ngày.....tháng....năm 2018

Cán bộ hướng dẫn

PHẦN NHẬN XÉT TÓM TẮT CỦA NGƯỜI CHẤM PHẢN BIỆN

1. Đánh giá chất lượng đề tài tốt nghiệp:

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

2. Cho điểm của cán bộ phản biện. (Điểm ghi cả chữ và số).

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Hải Phòng, ngày ..... tháng .... năm 2018.

Người chấm phản biện.

MỤC LỤC

LỜI NÓI ĐẦU

Trước đây khi công nghệ máy tính chưa phát triển, khi nói đến vấn đề an toàn bảo mật thông tin, chúng ta thường hay nghĩ đến các biện pháp nhằm đảm bảo cho thông tin được trao đổi hay cất giữ một cách an toàn và bí mật, chẳng hạn là các biện pháp như: Đóng dấu và ký niêm phong một bức thư để biết rằng lá thư có được chuyển nguyên vẹn đến người nhận hay không, dùng mật mã mã hóa thông điệp để chỉ có người gửi và người nhận hiểu được thông điệp, lưu giữ tài liệu trong các két sắt có khóa tại nơi được bảo vệ nghiêm ngặt.

Ngày nay với sự phát triển của khoa học công nghệ, đặt biệt là sự phát triển của Internet, việc sử dụng máy tính và điện thoại cá nhân càng trở lên rộng rãi, dẫn đến càng nhiều thông tin được lưu trữ trên máy tính và gửi đi trên mạng Internet. Do đó nhu cầu về an toàn và bảo mật thông tin trên máy tính càng nhiều và việc sử dụng mật mã mã hóa càng được phổ biến. Trong đồ án này, em thực hiện xây dựng chương trình mã hóa và giải mã mật mã hóa công khai RSA. Đồ án này gồm 3 chương:

Chương 1 : Tổng quan về mật mã học.

Chương 2 : Mật mã hóa công khai RSA.

Chương 3 : Chương trình RSA.

Em xin cám ơn thầy Nguyễn Văn Dương, giảng viên hướng dẫn, đã rất nhiệt tình chỉ bảo em hoàn thành đề tài này, cũng như các thầy cô khác trong bộ môn đã tạo điều kiện cho em trong suốt thời gian làm đề tài.

Hải Phòng, ngày .... tháng .... năm 2018

Sinh viên

Trị

Lê Thế Trị

Chương 1.

Tổng quan về mật mã học

1.1. Giới thiệu về mật mã học.

An toàn thông tin là bảo vệ các đặc tính riêng tư (confidentialy), toàn vẹn (intergrity) và khả dụng (availabity) của thông tin.

- C: (Confidentialy) bảo vệ tính riêng tư của dữ liệu thông qua các cơ chế chứng thực và mã hóa, ngăn ngừa những người không hợp lệ sẽ không được đọc những thông tin. Giống như các bì thư khi phát lương thưởng được dán chữ Confidentialy, chúng ta có thể hình dung trong môi trường công nghệ thông tin là một người chưa đăng nhập vào Domain sẽ không được truy cập những dữ liệu chỉ chia sẻ cho các Domain User.

- I: (Intergrity) bảo vệ tính toàn vẹn của dữ liệu thông qua các thuật toán RSA, SHA, MD5 ... ngăn ngừa attacker thay đổi các thông tin nhạy cảm trong quá trình truyền.

- A: (Available) bảo đảm dữ liệu luôn ở trong trạng thái sẵn sằng đáp ứng nhu cầu của người dùng.

- Non-Repudiation: Tính không thể chối bỏ, nghĩa là dữ liệu người nào gửi đi thì họ phải có trách nhiệm với các thông tin của mình thông qua các xác nhận nguồn gốc như chữ kí điện tử.

Để thực hiện điều này chúng ta áp dụng các biện pháp xác thực và mã hóa. Và mật mã học là nghiên cứu về vấn đề mã hóa. Mã hóa là một tiến trình biến đổi dữ liệu từ dạng plaintext (văn bản thuần túy dễ dàng nhận biết) thành kết quả ciphertext, dạng dữ liệu không thể đọc được nếu không được giải mã bằng các khóa thích hợp. Mục tiêu của mã hóa là ngăn ngừa việc tấn công đánh cắp dữ liệu trái phép hoặc phòng ngừa việc mất mát dữ liệu khi bị tấn công vật lý như trộm đĩa cứng, máy tính xách tay hay thậm chí đột nhập vào hệ thống vẫn không thể xem được dữ liệu riêng tư, bí mật đã được bảo vệ bằng các thuật toán mã hóa mạnh mẽ.

1.2. Khái niệm cơ bản về mật mã học

Kỹ thuật mật mã thông qua việc biến đổi hoặc mã hoá thông tin, biến đổi những thông tin nhạy cảm, vấn đề cơ mật thành những văn tự mã hoá có dạng hỗn loạn, làm cho tin tặc khó lòng mà đọc hiểu được, từ đó sẽ đạt được hai mục đích: một là, làm cho tin tặc không biết làm thế nào để giải mã nên cũng không thể thu được những thông tin có bất kỳ ý nghĩa nào trong chuỗi mật mã hỗn loạn đó; hai là làm cho tin tặc không có khả năng làm giả thông tin với chuỗi mật mã hỗn loạn như thế. Khoa học nghiên cứu kỹ thuật mật mã gọi là mật mã học.

Mật mã học bao gồm hai nhánh, là mật mã học lập mã và mật mã học phân tích. Mật mã học lập mã với ý là tiến hành mã hoá thông tin để thực hiện việc che

giấu thông tin, còn mật mã học phân tích là ngành học nghiên cứu phân tích giải dịch mật mã. Hai cái đối lập với nhau, nhưng lại thúc đẩy lẫn nhau.

Dùng phương pháp mật mã có thể che dấu và bảo hộ những thông tin cơ mật, làm cho người chưa được uỷ quyền không thể lấy được thông tin, những thông tin được giấu kín kia được gọi là văn bản rõ, mật mã có thể đem văn bản rõ biến đổi thành một loại hình khác, gọi là văn bản mật. Sự biến đổi văn bản rõ thành văn bản mật gọi là mã hoá bảo mật, quá trình người thu nhận hợp pháp khôi phục từ văn bản mật trở thành văn bản rõ được gọi là quá trình giải mã (hoặc giải mật). Người thu nhận phi pháp có ý đồ phân tích từ văn bản mật ra thành văn bản rõ, gọi là giải dịch.

1.3. Các thành phần của một hệ mật mã

Một hệ mật là một bộ 5 (P, C, K, E, D) thoả mãn các điều kiện sau:

+ P là một tập hữu hạn các bản rõ có thể

+ C là tập hữu hạn các bản mã có thể

+ K (không gian khoá) là tập hữu hạn các khoá có thể

+ Đối với mỗi $k \in K$ có một quy tắc mã ek: P -> C và một quy tắc giải mã tương ứng $dk \in D$. Mỗi ek: P -> C và dk: C -> P là những hàm mà: dk(ek(x)) = x với mọi bản rõ $x \in P$.

Điều kiện thứ 4 là tính chất chủ yếu. Nội dung của nó là nếu một bản rõ x được mã hoá bằng ek và bản mã nhận được sau đó được giải mã bằng dk thì ta phải thu được bản rõ ban đầu x. Trong trường hợp này hàm mã hoá ek phải là hàm đơn ánh, nếu không việc giải mã sẽ không thể thực hiện được một cách tường minh.

1.4. Phân loại các hệ mật mã

Hiện nay người ta đã thiết kế ra nhiều loại hệ thống mật mã, nếu như lấy khoá mật mã làm tiêu chuẩn có thể phân các hệ mật mã thành hai loại:

+ *Hệ mật mã đối xứng* (còn gọi là mật mã khoá đơn hoặc là mật mã khoá riêng): Trong các hệ mật mã này, khoá mật mã mã hoá bảo mật giống với khoá giải mã hoặc trên thực tế là cùng đẳng cấp. Lúc này khoá mật mã cần phải có một đường truyền an toàn để truyền đưa khoá mật mã từ phía người truyền cho phía người nhận. Đặc điểm của mật mã đối xứng là bất luận khi gia công bảo mật hay là khi giải mã đều sử dụng cùng một khoá mật mã. Do đó tính an toàn của mật mã này là sự an toàn của khoá mật mã. nếu như khoá mật mã bị tiết lộ, thì hệ thống mật mã này sẽ bị phá vỡ. Mật mã đối xứng có ảnh hưởng nhất là phép tính DES do cục tiêu chuẩn quốc gia Mỹ công bố vào năm 1977.

- *Ưu điểm:* Tính an toàn cao, tốc độ giải mã nhanh.

- *Nhược điểm:*

  - Theo sự mở rộng của quy mô mạng lưới, việc quản lý khoá mật mã trở thành một việc khó khăn.

- Không có cách nào giải quyết vấn đề xác nhận thông tin.
- Thiếu năng lực kiểm tra tự động sự tiết lộ khoá mật mã.

+ *Hệ mật mã bất đối xứng* (còn gọi là mật mã khoá công khai hoặc mật mã khoá đôi):

Trong các hệ mật mã này quá trình mã hoá và giải mã có chìa khoá khác nhau, lúc này không cần có đường truyền an toàn để truyền đưa khoá mật mã mà chỉ cần bộ phát sinh khoá mã tại chỗ để tạo ra khoá giải mã đồng thời lấy đó để khống chế các thao tác giải mã. Mật mã bất đối xứng là một thể chế mật mã loại mới do W.Diffie và M.E Hellman đề xuất năm 1976. Do quá trình mã hoá và giải mã của thể chế mật mã bất đối xứng không như nhau và khoá mã bảo mật là công khai, hơn nữa, chỉ yêu cầu bảo mật khoá giải mã, cho nên mật mã bất đối xứng không tồn tại vấn đề quản lý khoá mật mã. Mật mã bất đối xứng còn một ưu điểm nữa là có thể có khả năng ký tên chữ số và một số chức năng mới. Mật mã bất đối xứng nổi tiếng nhất là thể chế mật mã RSA do ba người là Rivest, Shamir và Adleman đề xuất năm 1977. Khuyết điểm của mật mã bất đối xứng là: phép tính mật mã là tương đối phức tạp, tốc độ giải mã chậm.

Do đó, việc bảo mật dữ liệu trên mạng nên dùng cơ chế bảo mật hỗn hợp kết hợp giữa mật mã đối xứng và mật mã bất đối xứng, tức là khi giải mã thì dùng mật mã đối xứng, khi truyền đưa khoá mật mã thì dùng mật mã bất đỗi xứng. Như thế tức là đã giải quyết được khó khăn trong việc quản lý khoá mật mã, lại vừa giải quyết được vấn đề tốc độ giải mã. Không còn hoài nghi gì nữa, nó là một phương pháp tương đối tốt để giải quyết vấn đề an toàn thông tin khi truyền đưa trên mạng hiện nay.

1.5. Một số phương pháp mã hóa

1.5.1. Mã hóa cổ điển

Mã hoá cổ điển là phương pháp mã hoá đơn giản nhất xuất hiện đầu tiên trong lịch sử ngành mã hoá. Thuật toán đơn giản và dễ hiểu. Những phương pháp mã hoá này là cở sở cho việc nghiên cứu và phát triển thuật toán mã hoá đối xứng được sử dụng ngày nay.

Trước khi mã hoá một bản rõ thành bản mã bằng các phương pháp mã hoá, ta xét một sự thiết lập tương ứng giữa các ký tự và các thặng dư theo modulo 26 như sau: A$\leftrightarrow$0, B$\leftrightarrow$1, …, Z$\leftrightarrow$25 hoặc theo bảng

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

+ Mật mã CAESAR

Một trong số những người sử dụng mật mã được biết sớm nhất, đó là Julias Caesar (Xê-Da). Ông đã làm cho các bức thư trở nên bí mật bằng cách dịch mỗi chữ cái đi ba chữ cái về phía trước trong bảng chữ cái (và ba chữ cái cuối cùng thành ba chữ cái đầu tiên). Đây là 1 ví dụ về sự mã hoá, tức là quá trình làm cho bức thư trở nên bí mật. Phương pháp mã hoá của CAESAR có thể được biểu diễn bởi hàm f, hàm này gán cho số nguyên không âm p, $p \leq 25$, số nguyên f(p) trong tập {0, 1, 2, … , 25} sao cho: $f(p) = (p+3) \bmod 26$.

Như vậy, trong phiên bản mã hoá của bức thư, chữ cái được biểu diễn bởi p sẽ được thay bằng chữ cái được biểu diễn bởi: $(p+3) \bmod 26$

Để phục hồi lại bức thư gốc đã được mã hoá theo mật mã của CAESAR, ta cần phải dùng hàm ngược f^{-1} của f: $f^{-1}(p) = (p-3) \bmod 26$. Nói cách khác, để tìm lại bức thư gốc, mỗi một chữ cái lùi lại ba chữ trong bảng chữ cái, với ba chữ cái đầu tiên chuyển thành ba chữ cái cuối cùng tương ứng của bảng chữ cái.

Nhận xét: phương pháp mã hoá của CAESAR không có độ an toàn cao. Phương pháp mã hoá này dễ bị khám phá bằng cách dựa vào tần xuất xuất hiện của các chữ cái trong bức thư.

+ Mã thay thế

Mã thay thế có thể được mô tả như sau: Cho P = C = Z26. K chứa mọi hoán vị có thể của 26 kí hiệu 0, 1, …, 25. với mỗi hoán vị $\pi \in K$, ta định nghĩa:

$e\pi(x) = \pi(x)$ và $d\pi(y) = \pi^{-1}(y)$

trong đó π^{-1} là hoán vị ngược của π

Ví dụ: mã hoá bản rõ: illustrate sử dụng mã thay thế với khoá là 1 hoán vị bất kì sau:

a	b	c	d	e	f	g	h	I	j	k	l	m	n	o	p	q	r	s	t
X	N	Y	A	H	P	O	G	Z	Q	W	B	T	S	F	L	R	C	V	M

u	v	w	x	y	z
U	E	K	J	D	I

Với khoá là một hoán vị bất kì ở trên thì bản rõ: illustrate sẽ tương ứng với bản mã sau (sử dụng hàm mã hoá $e\pi(x) = \pi(x)$): ZBBUVMCXMH

Hàm giải mã là phép hoán vị ngược, điều này được thực hiện bằng cách viết hàng thứ hai lên trước rồi sắp xếp theo thứ tự chữ cái. Ta nhận được:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
d	l	r	y	v	o	h	l	z	x	w	p	t	b	g	f	j	q	n	m

U	V	W	X	Y	Z
u	s	k	a	c	i

Sử dụng phép hoán vị ngược này ta biến đổi bản mã: ZBBUVMCXMH thành bản rõ là: illustrate

Nhận xét: Với mã thay thế, ta có một không gian khoá tương đối lớn (mỗi khoá là một hoán vị của 26 kí hiệu 0, 1, …, 25) do đó nó khó có thể bị thám theo phương pháp tìm khóa vét cạn, thậm chí cả bằng máy tính.

+ Mã vigenere

Sử dụng mã vigenere, ta có thể gán cho mỗi khoá k một chuỗi kí tự có độ dài m được gọi là từ khoá. Mật mã vigenere sẽ mã hoá đồng thời m kí tự: mỗi phần tử của bản rõ tương đương với m kí tự. Mã vigenere có thể được mô tả như sau:

Cho m là một số nguyên dương cố định nào đó. định nghĩa P=C=K=(Z)m với khoá k = (k1, k2, …, km), ta xác định:

ek(x1, x2, …, xm) = (x1+k1, x2+k2, …, xm+km) và

dk(y1, y2, …, ym) = (y1-k1, y2-k2, …, ym-km)

trong đó tất cả các phép toán được thực hiện trong Z26

Ví dụ: Mã hoá bản rõ: thiscryptosystemisnotsecure với m=6 và từ khoá là CIPHER bằng mã vigenere.

Từ khóa CIPHER tương ứng với dãy số k=(2, 18, 15, 7, 4, 17)

Biến đổi các phần tử của bản rõ thành các thặng dư theo modulo 26, viết chúng thành các nhóm 6 rồi cộng với từ khoá theo modulo 26 như sau:

19	7	8	18	2	17	24	15	19	14	18	24	18	19
2	8	15	7	4	17	2	8	15	7	4	17	2	8
21	15	23	25	6	8	0	23	8	21	22	15	20	1

4	12	8	18	13	14	19	18	4	2	20	17	4
15	7	4	17	2	8	15	7	4	17	2	8	15
19	19	12	9	15	22	8	25	8	19	22	25	19

Dãy kí tự: 21, 15, 23, 25, 6, 8, 0, 23, 8, 21, 22, 15, 20, 1, 19, 19, 12, 9, 15, 22, 8, 25, 8, 19, 22, 25, 19. Sẽ tương ứng với xâu bản mã là:

VPXZGIAXIVWPUBTTMJPWIZITWZT

Để giải mã ta biến đổi các phần tử của bản mã thành các thặng dư theo modulo 26, viết chúng thành các nhóm 6 rồi trừ với từ khoá theo modulo 26. Kết quả ta sẽ ra được bản rõ như sau: thiscryptosystemisnotsecure

Nhận xét: Ta thấy rằng số các từ khoá có thể với số độ dài m trong mật mã vigenere là 26m , bởi vậy, nó khó có thể bị thám theo phương pháp tìm khoá vét cạn, thậm chí với các giá trị m khá nhỏ, phương pháp tìm khoá vét cạn cũng phải yêu cầu thời gian khá lớn.

+ Mã hoán vị

Ý tưởng của mã hoán vị là giữ các ký tự của bản rõ không thay đổi nhưng sẽ thay đổi vị trí của chúng bằng cách sắp xếp lại các ký tự này. Mã hoán vị có thể được mô tả như sau:

Cho m là một số nguyên dương xác định nào đó. Cho P=C=(Z26)m và K gồm tất cả các hoán vị của {1, …, m}. Đối với một khoá π (tức là một hoán vị) ta xác định: $e_\pi(x1, \ldots, xm)=(x\pi(1), \ldots, x\pi(m))$ và

$$d_\pi(y1, \ldots, ym)=(y\pi^{-1}(1) , \ldots , y\pi^{-1}(m))$$

trong đó π^{-1} là hoán vị ngược của π.

Ví dụ: Mã hoá bản rõ: shesellsseashellsbytheseashore, sử dụng mã hoán vị, với m=6 và khoá là phép hoán vị π sau:

1	2	3	4	5	6
3	5	1	6	4	2

Trước tiên ta nhóm bản rõ thành các nhóm 6 ký tự:

shesel / lsseas / hellsb / ythese / ashore

Bây giờ mỗi nhóm 6 chữ cái được sắp xếp lại theo hoán vị π, ta có:

EESLSH / SALSES / LSHBLE / HSYEET / HRAEOS

Như vậy bản mã là: EESLSHSALSESLSHBLEHSYEETHRAEOS

Để giải mã ta sử dụng phép hoán vị ngược của π là π^{-1} có dạng:

1	2	3	4	5	6
3	6	1	5	2	4

Ta cũng nhóm bản mã thành nhóm 6 ký tự:

EESLSH / SALSES / LSHBLE / HSYEET / HRAEOS

Mỗi nhóm 6 chữ cái được sắp xếp lại theo hoán vị ngược π^{-1} ta có:

shesel / lsseas / hellsb / ythese / ashore

Cuối cùng ta thu được bản rõ là: shesellsseashellsbytheseashore

+ DES (Data Encryption Standard)

Lược đồ mã hoá được sử dụng phổ biến nhất dựa trên cơ sở của DES được phát triển vào năm 1977 bởi cục tiêu chuẩn quốc gia Mỹ, bây giờ là học viện tiêu chuẩn và công nghệ quốc gia (NIST), chuẩn xử lý thông tin liên bang. Đối với DES, dữ liệu được mã hoá trong khối 64 bit sử dụng khoá 56 bit. Thuật toán chuyển 64 bit đầu vào, biến đổi và đưa ra 64 bit đầu ra. DES được sử dụng phổ biến. Nó cũng là

chủ đề của rất nhiều cuộc tranh luận về mức độ an toàn. Để hiểu rõ giá trị của những cuộc tranh luận về DES chúng ta xem qua lại lịch sử của DES.

Cuối những năm 1960, IBM đã đưa ra dự án nghiên cứu trong bảo mật máy tính. Dự án kết thúc vào năm 1971 với việc cho ra đời thuật toán gọi là LUCIFER, hệ mật LUCIFER đã được sử dụng trong hệ thống phân phát tiền, cũng được phát triển bởi IBM. LUCIFER là một khối mã hoá Feistel được thực hiện trên khối 64 bit, sử dụng khoá có độ dài 128 bit. Những kết quả đầy hứa hẹn đưa ra bởi dự án LUCIFER, IBM đã bắt tay vào công việc đầy nỗ lực để phát triển thành một sản phẩm mã hoá thương mại có thể bán được, đó là một sản phẩm lý tưởng có thể thực hiện được trên một chíp đơn. Công đầu phải kể đến Walter Tuchman và Carl Meyer, nó không chỉ làm rắc rối cho những nhà thiết kế mà còn cần phải có những lời khuyên của những nhà kỹ thuật và tư vấn ở bên ngoài đó là NSA. Kết quả của nỗ lực này là một phiên bản LUCIFER có chọn lọc kỹ lưỡng, phiên bản này có thể chống lại các phương pháp giải dịch, nhưng nó cũng làm giảm độ dài khoá xuống còn 56 bit, để phù hợp trên một chip đơn. Năm 1973 cục tiêu chuẩn quốc gia Mỹ (NBS) đưa ra một yêu cầu đề nghị cho một chuẩn mã hoá quốc tế. IBM đã đưa ra xem xét những kết quả của dự án Tuchman-Meyer. Kết quả nó được đề nghị là thuật toán tốt nhất và được công nhận vào năm 1977 như là một chuẩn mã hoá dữ liệu.

Trước khi được công nhận như là một chuẩn mã hoá dữ liệu, DES đã trở thành chủ đề của nhiều cuộc phê bình mạnh mẽ, và sự phê bình này vẫn chưa lắng xuống cho đến ngày hôm nay. Có hai vấn đề được đưa ra không làm hài lòng những nhà phê bình. Đầu tiên, chiều dài khoá của thuật toán LUCIFER nguyên thuỷ của IBM là 128 bit nhưng hệ thống được đề nghị chỉ dùng 56 bit, một sự giảm rất lớn trong độ dài khoá 72 bit. Những nhà phê bình lo sợ rằng (và vẫn sợ) chiều dài khoá quá nhỏ để chống lại những cuộc tấn công quy mô lớn. Mặt thứ 2 cần quan tâm là tiêu chuẩn thiết kế cho cấu trúc bên trong của DES, những hộp S phải được coi là mật. Như vậy, những người sử dụng không thể chắc chắn rằng cấu trúc bên trong của DES là tự do cho bất kỳ những điểm yếu được che dấu, điều này sẽ cho phép NSA hướng tới những thông báo giải mã không có lợi cho khoá. Những sự kiện xảy ra sau, đặc biệt gần đây làm việc trên những sự giải dịch khác nhau, dường như chỉ rõ rằng DES có một cấu trúc bên trong rất mạnh.

1.5.2. Thuật toán mã hóa công khai

+ Hệ mật RSA

Ý tưởng về một hệ mật khoá công khai đã được Diffie và Hellman đưa ra vào 1976. Còn việc hiện thực hóa hệ mật khoá công khai thì do Rivest, Shamir và Adleman đưa ra đầu tiên vào 1977, họ đã tạo nên hệ mật RSA nổi tiếng.

Hệ mật này sử dụng các tính toán trong Zn, trong đó n là tích của 2 số nguyên tố phân biệt p và q. Ta có thể mô tả hệ mật RSA như sau:

Cho n = p.q trong đó p và q là các số nguyên tố. Đặt P=C=Zn và định nghĩa:

K={(n,p,q,a,b):n=pq, p, q là các số nguyên tố, $ab \equiv 1 (\mod \phi(n))$}

Với K =(n,p,q,a,b) ta xác định

ek(x)=xb mod n

dk(y)=ya mod n

($x,y \in Zn$) các giá trị n và b được công khai và các giá trị p, q, a được giữ kín,

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

Quá trình thực hiện hệ mật RSA: (người gửi: Alice; người nhận: Bob)

+ Bob tạo hai số nguyên tố lớn p và q

+ Bob tính n = pq và $\phi(n) = (p-1)(q-1)$

+ Bob chọn một số ngẫu nhiên b ($0<b<\phi(n)$) sao cho UCLN(b, $\phi(n)$)=1

+ Bob tính a = b-1 mod $\phi(n)$ bằng cách dùng thuật toán Euclide

+ Bob công bố n và b trong một danh bạ và dùng chúng làm khoá công khai.

Ví dụ: giả sử Bob chọn p=101 và q=113

Khi đó n = 11413 và $\phi(n)$ = 100x112=11200. Vì 11200 = 26527, nên có thể dùng một số nguyên b như một số mũ mã hoá khi và chỉ khi b không chia hết cho 2, 5 hoặc 7. Vì thế trong thực tế Bob sẽ không phân tích $\phi(n)$, anh ta sẽ kiểm tra điều kiện UCLN($\phi(n)$,b) = 1 bằng thuật toán Euclide. Giả sử Bob chọn b = 3533, khi đó theo thuật toán Euclide mở rộng: b-1=6597 mod 11200. Bởi vậy, số mũ mật để giải mã cho Bob là a=6597. Bob sẽ công bố n = 11413 và b = 3533 trong một danh bạ. Bây giờ, giả sử Alice muốn gửi bản rõ 9726 tới Bob. Cô ta sẽ tính
97263533 mod 11413 = 5761 rồi gửi bản mã 5761 trên kênh. Khi Bob nhận được bản mã 5761, anh ta sử dụng số mũ a mật để tính: 57616597 mod 11414 = 9726.

Với hệ mật RSA được trình bày ở trên ta thấy cách tấn công dễ thấy nhất đối với hệ mật này là thám mã cố gắng phân tích n ra các thừa số nguyên tố. Nếu thực hiện được phép phân tích này thì có thể dễ dàng tính được $\phi(n) = (p-1)(q-1)$ và rồi tính số mũ a và b đúng như Bob đã làm.Vì thế để hệ RSA được coi là mật thì nhất thiết n = pq phải là một số đủ lớn để việc phân tích nó sẽ không có khả năng về mặt tính toán.

+ Hệ mật Elgamal

Hệ mật Elgamal là một hệ mật mã công khai dựa trên bài toán logarithm rời rạc. Nó là một hệ mật không tất định vì bản mã phụ thuộc vào cả bản rõ x lẫn giá trị ngẫu nhiên k do người gửi chọn. Bởi vậy sẽ có nhiều bản mã được mã từ cùng bản rõ.

Bài toán logarithm rời rạc trong Zp:

Đặc trưng của bài toán: I = (p,α,β) trong đó p là số nguyên tố $\alpha \in Zp$ là phần tử nguyên thuỷ, $\beta \in Zp^*$

Mục tiêu: hãy tìm một số nguyên duy nhất a, $0 \le a \le p-2$ sao cho

$$\alpha a \equiv \beta \pmod p$$

Ta sẽ xác định số nguyên a bằng $\log_\alpha \beta$.

Hệ mật khoá công khai Elgamal trong Zp*:

Cho p là số nguyên tố sao cho bài toán logarithm rời rạc trong Zp là khó giải. cho $\alpha \in Zp^*$ là phần tử nguyên thuỷ. Giả sử P=Zp*, C= Zp* x Zp*. Ta định nghĩa:

$K = \{(p, \alpha, a, \beta): \beta \equiv \alpha a \pmod p\}$

Các giá trị p, α, β được công khai, còn a giữ kín

Với $K = (p, \alpha, a, \beta)$ và một số ngẫu nhiên bí mật $k \in Zp-1$ ta xác định:

$$ek(x,k)=(y1,y2)$$

Trong đó

$$y1 = \alpha k \bmod p$$

$$y2 = x\beta k \bmod p$$

Với $y1, y2 \in Zp^*$ ta xác định:

$$dk(y1,y2) = y2(y1a)\text{-}1 \bmod p$$

Ví dụ:

Cho p = 2579, α = 2, a = 765. Khi đó β=2765 mod 2579 = 949

Bây giờ giả sử người gửi muốn gửi thông báo x=1299 tới người nhận. Giả sử số ngẫu nhiên k mà người gửi chọn là k=853. Sau đó người gửi tính:

$$y1 = 2853 \bmod 2579 = 435$$

$$y2 = 1299 \text{ x } 949853 \bmod 2579 = 2396$$

Khi người nhận thu được bản mã y = (435, 2396), người nhận tính

$$x = 2396 \text{ x } (435765)\text{-}1 \bmod 2579 = 1299$$

Đây chính là bản rõ mà người gửi đã mã hoá.

Chương 2.

Mã hóa RSA

2.1. Lý thuyết số

2.1.1. Thuật toán Euclid

Thuật toán Euclid là một thuật toán để xác định ước chung lớn nhất (GCD – Greatest Common Divisor) của 2 phần tử thuộc vùng Euclid (ví dụ: các số nguyên). Thuật toán Euclid là một trong những thuật toán cổ nhất được biết đến, từ khi xuất hiện trong cuốn Euclid's Elements khoảng năm 300 trước công nguyên. Euclid khởi đầu đã trình bày rõ ràng vấn đề về phương diện hình học, như vấn đề tìm ra một thước đo chung có độ dài hai đường thẳng, và thuật toán của ông đã xử lý bằng cách lặp lại phép trừ đoạn dài hơn cho đoạn ngắn hơn. Tuy nhiên , thuật toán đã hầu như không được phát hiện ra bởi Euclid và nó đã có thể được biết đến sớm hơn 200 năm. Nó cũng đã được biết đến bởi Eudoxus of Cnidus (khoảng 375 trước công nguyên) và Aristotle (khoảng 330 trước công nguyên).

Thuật toán Euclid sử dụng để giải một phương trình vô định nguyên (còn được gọi là phương trình Đi-ô-phăng) có dạng: $a \times x + b \times y = c$,

trong đó *a, b, c* là các hệ số nguyên, *x, y* là các ẩn nhận giá trị nguyên. Điều kiện cần và đủ để phương trình này có nghiệm (nguyên) là *gcd(a,b)* là ước của *c*. Khẳng định này dựa trên một mệnh đề sau:

Trong số học đã biết rằng nếu $d = gcd(a,b)$ thì tồn tại các số nguyên x, y sao cho $a \times x + b \times y = d$

Giải thuật Euclid mở rộng kết hợp quá trình tìm gcd(a,b) trong thuật toán Euclid với việc tìm một cặp số x, y thoản mãn phương trình Đi-ô-phăng. Giả sử cho hai số tự nhiên a, b, ngoài ra a >b>0. Đặt r_0 = a, r_1 = b, chia r_0 cho r_1 được số dư r_2 và thương số nguyên q_1. Nếu $r_2 = 0$ thì dừng lại, nếu r_2 khác không, chia r_1 cho r_2 được số dư r_3 ..... Vì dãy các r_i là giảm thực sự nến sau hữu hạn bước ta được số dư $r_{m+2} = 0$.

$r_o = q_1 \times r_1 + r_2, 0 < r_2 < r_1;$

$r_1 = q_2 \times r_2 + r_3, 0 < r_3 < r_2;$

....;

$r_{m-1} = q_m \times r_m + r_{m+1} 0 < r_{m+1} < r_m$

$r_m = q_{m+1} \times r_{m+1}$

trong đó số dư cuối cùng khác 0 là $r_{m+1} = d$. Bài toán đặt ra là tìm x, y sao cho

$a \times x + b \times y = r_{m+1} (= d)$

Để làm điều này, ta tìm x, y theo công thức truy hồi, nghĩa là sẽ tìm x_i và y_i sao cho:

$a \times x_i + b \times y_i = r_i$ với $i = 0,1,....$

Ta có

$a \times 1 + b \times 0 = a = r_o$ và $a \times 0 + b \times 1 = b = r_1$,

Nghĩa là:

$x_o = 1, x_1 = 0$ và $y_o = 0, y_1 = 1.$ (2.1)

Tổng quát, giả sử có

$a \times x_i + b \times y_i = r_i$ với $i = 0,1,....$

$a \times x_{i+1} + b \times y_{i+1} = r_{i+1}$ với $i = 0,1,....$

Khi đó từ

$r_i = q_{i+1} \times r_{i+1} + r_{i+2}$

Suy ra

$r_i - q_{i+1} \times r_{i+1} = r_{i+2}$

$(a \times x_i + b \times y_i) - q_{i+1} \times (a \times x_{i+1} + b \times y_{i+1}) = r_{i+2}$

$a \times (x_i - q_{i+1} \times x_{i+1}) + b \times (y_i - q_{i+1} \times y_{i+1}) = r_{i+2}$

từ đó, có thể chọn

$x_{i+2} = x_i - q_{i+1} \times x_{i+1}$ (2.2)

$y_{i+2} = y_i - q_{i+1} \times y_{i+1}$ (2.3)

Khi $i = m - 1$ ta có được x_{m+1} và y_{m+1}. Các công thức (1), (2), (3) là công thức truy hồi để tính x,y.

Áp dụng giải thuật Euclid mở rộng tìm số nghịch đảo trong vành Z_m

Số nghịch đảo trong vành $\boldsymbol{Z_m}$

Trong lý thuyết số, vành Z_m được định nghĩa là vành thương của $\mathbb{Z}$ với quan hệ đồng dư theo môđun m (là quan hệ tương đương) mà các phần tử của nó là các lớp đồng dư theo mođun m (m là số nguyên dương lớn hơn 1). Ta cũng có thể xét Z_m chỉ với các đại diện của nó. Khi đó

$Z_m = \{0, 1, ..., m - 1\}$

Phép cộng và nhân trong Z_m là phép toán thông thường được rút gọn theo mođun m:

$a + b = (a + b) \bmod m$ (2.4)

$a \times b = (a \times b) \bmod m$ (2.5)

Phần tử a của Z_m được gọi là khả nghịch trong Z_m hay khả nghịch theo mođun m nếu tồn tại phần tử a' trong Z_m sao cho $a \times a' = 1$ trong Z_m hay $a \times a' \equiv 1 (\bmod m)$. Khi đó a' được gọi là nghịch đảo modulo m của a. Trong lý thuyết số đã chứng minh rằng, số a là khả nghịch theo mođun m khi và chỉ khi GCD của a và m bằng 1. Khi đó tồn tại các số nguyên x, y sao cho

$m \times x + a \times y = 1$

Đẳng thức này lại chỉ ra x là nghịch đảo của a theo mođun m. Do đó có thể tìm được phần tử nghịch đảo của a theo mođun m nhờ thuật toán Euclid mở rộng khi chia m cho a.

2.1.2. Số nguyên tố

Một số nguyên tố p > 1 là số nguyên tố khi và chỉ khi số chia của nó là 1 và p. Số nguyên tố đóng vai trò quan trọng trong lý thuyết số và trong các kỹ thuật. Bất cứ số nguyên a>1 đều đươc tính theo cách duy nhất.

$a = q_1^{a_1} \times q_2^{a_2} \times \ldots \times q_t^{a_t}$ (2.6)

trong đó : $q_1 < q_2 < \ldots\ldots < q_t$ là số nguyên tố

a_i là 1 số nguyên dương

Ví dụ: $91 = 7^1 \times 13^1$

$3600 = 2^4 \times 3^2 \times 5^2$

$11011 = 7^1 \times 11^2 \times 13^1$

Nó còn được biểu diễn theo một cách khác phổ biến hơn. Nếu P là tập hợp của tất cả các số nguyên tố, sau đó bất kì số nguyên dương a có thể được viết duy nhất trong cách trong cách sau:

$a = \prod_{p \in P} p^{a_p}$ trong đó: $a_p \geq 0$

Phía bên phải là kết quả trên tất cả các số nguyên tố p, khi cho bất kì số nguyên dương a, hầu hết các số mũ a_p sẽ là 0. Giá trị của bất kì số nguyên dương nào đó có thể được xác định bằng cách chỉ đơn giản liệt kê tất cả các số mũ không phải là ngoại lệ trong các công thức nêu trên.

Số nguyên 12 được biểu diễn bởi $\{a_2 = 2 , a_3 = 1\}$

Số nguyên 18 được biểu diễn bởi $\{a_2 = 1, a_3 = 2 \}$

Số nguyên 91 được biểu diễn bởi $\{a_7 = 1, a_{13} = 1\}$

Nhân 2 số tương đương với việc cộng 2 số mũ .

Ví dụ: $a = \prod_{p \in P} p^{a_p}$ $b = \prod_{p \in P} p^{b_p}$

Với $k = a \times b$. Ta biết 1 số nguyên k có thể biểu diễn qua các số nguyên tố:

$k = \prod_{p \in P} p^{k_p}$ trong đó $k_p = a_p + b_p$ với $\forall p \in P$

Bảng 2.1 Bảng số nguyên tố nhỏ hơn 2000

2	101	211	307	401	503	601	701	809	907	1009	1103	1201	1301	1409	1511	1601	1709	1801	1901
3	103	223	311	409	509	607	709	811	911	1013	1109	1213	1303	1423	1523	1607	1721	1811	1907
5	107	227	313	419	521	613	719	821	919	1019	1117	1217	1307	1427	1531	1609	1723	1823	1913
7	109	229	317	421	523	617	727	823	929	1021	1123	1223	1319	1429	1543	1613	1733	1831	1931
11	113	233	331	431	541	619	733	827	937	1031	1129	1229	1321	1433	1549	1619	1741	1847	1933
13	127	239	337	433	547	631	739	829	941	1033	1151	1231	1327	1439	1553	1621	1747	1861	1949
17	131	241	347	439	557	641	743	839	947	1039	1153	1237	1361	1447	1559	1627	1753	1867	1951
19	137	251	349	443	563	643	751	853	953	1049	1163	1249	1367	1451	1567	1637	1759	1871	1973
23	139	257	353	449	569	647	757	857	967	1051	1171	1259	1373	1453	1571	1657	1777	1873	1979
29	149	263	359	457	571	653	761	859	971	1061	1181	1277	1381	1459	1579	1663	1783	1877	1987
31	151	269	367	461	577	659	769	863	977	1063	1187	1279	1399	1471	1583	1667	1787	1879	1993
37	157	271	373	463	587	661	773	877	983	1069	1193	1283		1481	1597	1669	1789	1889	1997
41	163	277	379	467	593	673	787	881	991	1087		1289		1483		1693			1999
43	167	281	383	479	599	677	797	883	997	1091		1291		1487		1697			
47	173	283	389	487		683		887		1093		1297		1489		1699			
53	179	293	397	491		691				1097				1493					
59	181			499										1499					
61	191																		
67	193																		
71	197																		
73	199																		
79																			
83																			
89																			
97																			

$k = 12 \times 18 = (2^2 \times 3^1) \times (2^1 \times 3^2) = 216$

$k_2 = 2 + 1 = 3 \qquad k_3 = 1 + 2 = 3$

$216 = 2^3 + 3^3 = 8 \times 27$

Bất kì số nguyên nào có dạng p^n chỉ có thể chia 1 số nguyên nhỏ hơn hoặc số nguyên có dạng p^j với $j \leq n$.

Cho $a = \prod_{p \in P} p^{a_p} \qquad b = \prod_{p \in P} p^{b_p}$

Nếu $a|b$ thì $a_p \leq b_p$ với $\forall p$

Ví dụ: $a = 12; \qquad b = 36; \qquad 12|36$

$12 = 2^2 \times 3; \qquad 36 = 2^2 \times 3^2$

$a_2 = 2 = b_2; \; a_3 = 1 \leq 2 = b_3$

Như vậy, bất đẳng thức: $a_p \leq b_p$ đúng với mọi số nguyên tố. Nó rất dễ để xác định số ước chung lớn nhất của số nguyên dương nếu chúng ta thể hiện mỗi số nguyên dưới dạng của số nguyên tố

Ví dụ: $300 = 2^2 \times 3^1 \times 5^2; \qquad 18 = 2^1 \times 3^2$

$\gcd(18,300) = 2^1 \times 3^1 \times 5^0 = 6$

Ta có mối liên kết: Nếu $k = \gcd(a,b)$ thì $k_p = \min(a_p, b_p) \quad \forall p$

Xác định các phần tử nguyên tố của một số lớn không phải là một việc dễ dàng do đó mối liên kết trên không phải là phương pháp thực tế để tìm các ước chung lớn nhất.

2.1.3. Định lý Fermat và định lý Euler.

Hai định lý đóng vai trò quang trọng nhất trong mã hóa khóa công khai là Fermat và Euler.

2.1.3.1. Định lý Fermat.

Định lý Fermat được phát biểu như sau: Nếu p là số nguyên tố và a là số nguyên dương không thể chia hết cho p, thì: $a^{p-1} \equiv 1 (\text{mod } p)$

Ví dụ: a = 5; p =3

Ta có $a^{p-1} = 5^{3-1}=5^2 = 25 \equiv 1(\text{mod } p) = \equiv 1(\text{mod } 3)$

Chứng minh: Ta xét các số nhỏ hơn p: {1, 2, ..., p-1} và nhân mỗi phần tử với a, (modulo p) để có được tập X = {a mod p, 2a mod p, ..., (p-1)a mod p }. Không phần tử nào của X bằng 0 bởi vì a không chia hết cho p. Hơn nữa không có hai số nguyên nào của X bằng nhau. Để xét điều này ta giả sử: $ja \equiv ka (\text{mod } p)$, với $1 \leq j < k \leq p-1$. Bởi vì a tương đối nguyên tố với p, chúng ta có thể loại bỏ a từ cả 2 vế của phương trình $j \equiv k(\text{mod } p)$. Sự cân bằng cuối cùng này là không thể, bởi vì j và k là hai số nguyên dương nhỏ hơn p. Vì vậy, ta biết rằng các phần tử (p-1) của X là tất cả các số nguyên dương không có hai phần tử bằng nhau. Chúng ta có thể kết luận X bao gồm các tập số nguyên (1, 2, ..., p-1) theo thứ tự nào đó. Nhân các số trong cả hai tập (p và X) và lấy kết quả mod p.

$a \times 2a \times .... \times (p-1)a \equiv [(1 \times 2 \times ..... \times (p-1)](\text{mod } p)$

$a^{p-1}(p-1)! \equiv (p-1)!(\text{mod } p)$

Bỏ (p-1)! => $a^{p-1} \equiv 1(\text{mod } p)$ (2.7)

Một dạng thay thế của định lý Fermat: Nếu p là số nguyên tố, a là một số nguyên dương thì:

$$a^p \equiv a (\text{mod } p) \quad (2.8)$$

Lưu ý rằng hình thức đầu của định lý Fermat: $a^{p-1} \equiv 1(\text{mod } p)$ yêu cầu rằng a tương đối nguyên tố với p, nhưng dạng này không đúng.

p=5; a = 3; $a^p = 3^5 = 243 \equiv 3(\text{mod } 5) = a(\text{mod } p)$

p=5; a=10; $a^p = 10^5 = 100000 \equiv 10 (\text{mod } 5) \equiv 0(\text{mod } 5) = a(\text{mod } 5)$

2.1.3.2. Định lý Euler

Trước khi trình bày định lý Euler, ta cần cần giới thiệu một điều quan trọng gọi là hàm số của Euler, được viết $\emptyset(n)$ và được định nghĩa là số lượng các số nguyên dương nhở hơn n và tương đối nguyên tố với n. Theo quy ước $\emptyset(1)=1$

Ví dụ: Xác định $\emptyset(37)$ và $\emptyset(35)$

Bởi vì 37 là số nguyên tố, suy ra các số nguyên từ 1 đến 36 đều tương đối nguyên tố với 37. Như vậy $\emptyset(37)=36$.

Với $\emptyset(35)$, ta có một danh sách những số nguyên dương nhỏ hơn 35 và tương đối nguyên tố với 35 là:

{1,2,3,4,6,8,9,11,12,13,16,17,18,19,22,23,24,26,27,28,29,31,32,33,34}

có 24 số trong danh sách. Như vậy $\emptyset(35)$=24.

Bảng 2.2. Một số giá trị hàm số của Euler

n	$\emptyset(n)$	n	$\emptyset(n)$	n	$\emptyset(n)$
1	1	11	10	21	12
2	1	12	4	22	10
3	2	13	12	23	22
4	2	14	6	24	8
5	4	15	8	25	20
6	2	16	8	26	12
7	6	17	16	27	18
8	4	18	6	28	12
9	6	19	18	29	28
10	4	20	8	30	8

Trong bảng 2.2. là danh sách 30 giá trị đầu tiên của $\emptyset(n)$. Giá trị $\emptyset(1)$ là không có nghĩa nhưng định nghĩa có một giá trị 1. Cần phải rõ ràng rằng, đối với một số nguyên tố p: $\emptyset(p)$ = p – 1. Bây giờ giả sử ta có hai số nguyên tố p và q khác nhau. Sau đó chúng ta có thể thấy rằng:

Với: n = p×q.

Ta có: $\emptyset(n)= \emptyset(pq) = \emptyset(p) \times \emptyset(q)$= (p – 1) × (q – 1)

Để thấy rằng: $\emptyset(n)= \emptyset(p) \times \emptyset(q)$.

Hãy xét rằng tập hợp các số nguyên dương ít hơn n là tập {1, 2, ..., (pq–1)}. Tập các số nguyên không tương đối nguyên tố với n là tập {p, 2p, ..., (q–1)p} và tập {q, 2q, ..., (p–1)q}.

Suy ra: $\emptyset(n)$ = (pq–1) - [(p–q)+(q–1)]

= pq - (p+q)+1

= (p–1)×(q–1)

= $\emptyset(p) \times \emptyset(q)$

Ví dụ: $\emptyset(21) = \emptyset(3) \times \emptyset(7)$ = (3–1)×(7–1) = 2×6 =12. Mười hai số nguyên trong đó là: {1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20}

Định lý Euler được phát biểu như sau: Với mỗi số a và n tương đối nguyên tố với nhau thì:

$$a^{\emptyset(n)} \equiv 1 \text{ (mod n)} \qquad (2.9)$$

Chứng minh: Phương trình (2.9) là đúng nếu n là số nguyên tố, bởi vì trong trường hợp $\emptyset(n)$ = (n–1) và định lý của Fermat. Tuy nhiên nó cũng đúng với bất kì số nguyên n. Vì $\emptyset(n)$ là số lượng các số nguyên nhỏ hơn n và tương đối nguyên tố với n. Xét tập các số nguyên như vậy: R=$\{x_1, x_2, \ldots\ldots, x_{\emptyset(n)}\}$. Tức là, mỗi phần tử x_i của R là một số nguyên nhỏ hơn n và gcd(x_i,n) = 1. Bây giờ nhân mỗi phần tử với a, mod n:

$$S = \{(ax_1 \bmod n), (ax_2 \bmod n),\ldots\ldots\ldots (ax_{\emptyset(n)} \bmod n)\}.$$

Tập S là một phép hoán vị của R,bởi các chứng minh sau:

- Bởi vì a là số tương đối nguyên tố n và x_i tương đối nguyên tố với n, ax_i cũng phải tương đối nguyên tố với n. Như vậy, tất cả các phần tử của S là những số nguyên nhỏ hơn n và tương đối nguyên tố với n.
- Không có nhân đôi trong S. Với công thức nếu (a×b)≡(a×c) (mod n) và a tương đối nguyên tố với n thì b ≡ c(mod n).

Vì thế ta sẽ có:

$$\Pi_{i=1}^{\emptyset(n)}(ax_i \bmod n) = \Pi_{i=1}^{\emptyset(n)} x_i$$
$$\Pi_{i=1}^{\emptyset(n)} ax_i \equiv \Pi_{i=1}^{\emptyset(n)} x_i (\text{mod n})$$
$$a^{\emptyset(n)} \times [\Pi_{i=1}^{\emptyset(n)} x_i] \equiv \Pi_{i=1}^{\emptyset(n)} x_i (\text{mod n})$$
$$a^{\emptyset(n)} \equiv 1 \text{ (mod n)}$$

2.1.4. Kiểm tra số nguyên tố

Đối với nhiều thuật toán mật mã, nó là cần thiết để chọn một hoặc nhiều số nguyên tố lớn một cách ngẫu nhiên. Vì vậy, chúng ta phải đối mặt với một vấn đề về xác định một con số lớn có là số nguyên tố hay không. Trong phần này, ta trình bày một thuật toán rất được quan tâm và phổ biến, có thể thấy rằng thuật toán này mang lại một số không nhất thiết là một số nguyên tố. Tuy nhiên, thuật toán có thể mang lại một số gần như chắc chắn là một số nguyên tố.

2.1.4.1. Thuật toán Miller-Rabin

Thuật toán Miller-Rabin thường được sử dụng để kiểm tra một số lớn là số nguyên tố. Trước khi giải thích thuật toán, ta cần một số kiến thức căn bản. Bất kì một số dương lẻ nào n ≥ 3 có thể được biểu diễn như sau:

$$n - 1 = 2^k q \text{ với } k > 0, q \text{ là số lẻ.}$$

Để thấy điều này, lưu ý rằng (n – 1) là 1 số nguyên. Sau đó chia (n–1) cho 2 đến khi kết quả là số lẻ q, với tổng số lần chia là k. Nếu n được biểu diễn dưới dạng nhị phân

số, thì kết quả đạt được bằng cách dịch chuyển số sang bên phải cho đến khi số bên phải là số 1, với tổng số k lần dịch chuyển.

Ví dụ: n = 23 => n-1 = 22 => n-1 = $2^1 \times 11$ => q=11 và k = 1

+ Hai tính chất của số nguyên tố.

- *Tính chất thứ nhất:* Nếu p là số nguyên tố và a là số nguyên dương nhỏ hơn p thì: a^2 mod p = 1 khi và chỉ khi a mod p = 1 hoặc a mod p = -1 mod p = p–1. Theo quy tắc của module số học (a mod p)(a mod p) = a^2 mod p.

Do đó: Nếu a mod p = 1 hoặc a mod p = -1 thì a^2 mod p = 1.

Ngược lại: Nếu a^2 mod p = 1 thì (a mod p)^2 = 1. Nó chỉ đúng với a mod p = 1 hoặc a mod p = -1

- *Tính chất thứ hai:* Cho p là số nguyên tố lớn hơn 2. Sau đó ta có thể viết p – 1 = 2^kq với k > 0 và q là số lẻ. Cho a là số nguyên bất kì trong dãy. 1 < a < p–1 thì một trong hai điều kiện sau là đúng:

1. a^q là phù hợp với 1 modulo p đó là a^q mod p =1 hoặc $a^q \equiv 1 (\text{mod } p)$

2. Một trong những số a^q, a^{2q}, a^{4q} ,.....$a^{2^{k-1}q}$ thích hợp với -1 modulo p, tức là có 1 số j trong dải ($1 \leq j \leq k$). Như vậy $a^{2^{j-1}q}$ mod p = - 1 mod p = p – 1. Tương đương với $a^{2^{j-1}q} \equiv -1 (\text{mod } p)$

Chứng minh: Theo định lý Fermat ta thấy $a^{n-1} \equiv 1 (\text{mod } n)$. Nếu n là số nguyên tố ta có: p – 1 = 2^kq. Do đó: a^{p-1} mod p = $a^{2^k q}$ mod p = 1. Vì vậy nếu nhìn vào dãy số ta thấy:

a^q mod p, a^{2q} mod p, a^{4q} mod p ..., $a^{2^{k-1}q}$ mod p, $a^{2^k q}$ mod p (2.10)

Trong biểu thức trên ta có con số cuối cùng bằng 1. Hơn nữa mỗi số trước là căn bậc 2 của số đứng sau. Do đó một trong những khả năng sau phải là đúng.

1. Số đầu tiền trong danh sách và tất cả các số tiếp theo trong danh sách bằng 1.

2. Một số trong danh sách không bằng 1, nhưng bình phương mod p của nó bằng 1. Nhờ tính chất đầu tiên của các số nguyên tố được định nghĩa ở trên, chúng ta biết rằng số duy nhất thỏa mãn điều kiện này là (p–1). Vì vậy, trong trường hợp này danh sách chứa một phần tử bằng (p–1).

+ Chi tiết về thuật toán

Nếu n là số nguyên tố thì phần tử đầu tiên trong danh sách dư lượng hoặc phần dư, (a^q,a^{2q},a^{4q}, ..., $a^{2^k q}$) modulo n bằng một hoặc một số phần tử trong danh sách bằng (n–1), nếu không n là hỗn hợp (như là, không phải là số nguyên tố). Mặt

khác, nếu điều kiện được đáp ứng, điều đó không nhất thiết có nghĩa là n là số nguyên tố.

Cho ví dụ:

Nếu n = 2047 = 23 × 89, thì n – 1 = 2 × 1023

Ta tính 2^{1023} mod 2047 = 1, lên 2047 thỏa mãn điều kiện nhưng phải là số nguyên tố.

Ta có thể sử dụng thuộc tính trước để tạo ra một phép thử cho tính nguyên tố. Cách thức kiểm tra: Lấy 1 số nguyên tố n là đầu vào và trả kết quả "hỗn hợp" nếu n chắc chắn không phải số nguyên tố và kết quả trả về là "chưa đi đến kết quả" nếu n có thể hoặc không thể là 1 số nguyên tố

Các bước để kiểm tra như sau:

1. Tìm số nguyên k, q với k > 0, q la số lẻ. Sao cho (n – 1)= 2^k.
2. Lựa chọn 1 số nguyên ngẫu nhiên a: 1< a < n-1
3. Nếu a^q mod n thì trả về kết quả là "chưa đi đến kết quả"
4. Với j: 0 < j < k-1.
5. Nếu $a^{2^j q}$ mod n = n–1 thì trả về "chưa đi đến kết quả".
6. Ngược lại trả về "hỗn hợp".

Ví dụ 1: n = 29. Ta có (n – 1) = 28 = 2^2 × 7=2^kq

- Đầu tiên ta thử a = 10
- Ta tính 10^7 mod 29 = 17 (a^q mod n): Kết quả này không phải 1 hay 28
- Vậy nên ta tiếp tục kiểm tra
- Tính toán tiếp theo ta thấy $(10^7)^2$ mod 29 = 28 và kiểm tra trả về là "chưa đi đến kết quả". Tức là 29 có thể là số nguyên tố.
- Ta thử với a =2. (a^q mod n) = 2^7 mod 29 = 12;

$$[(a^q)^2 \bmod n] = 2^{14} \bmod 29 = 28$$

Lần này vẫn trả về là chưa đi đến kết quả.

- Nếu chúng ta thực hiện kiểm tra cho tất cả các số nguyên a trong khoảng từ 1 đên 28, chúng ta nhận được 1 kết quả là "chưa đi đến kết quả", điều này tương đương với n là số nguyên tố.

Ví dụ 2: Ta có n = 13 × 17 = 221

(n–1) = 220 = 2^2 × 55 = 2^kq

- Thử a = 5, ta có a^q = 5^{55} mod 221 = 112, nó không phải là 1 hay 220.
- Ta thử tiếp $(5^{55})^2$ mod 221 = 168. Bởi vì chúng ta đã dùng tất cả các giá trị của j (j = 0 và j = 1) trong bước thứ 4 trong quá trình kiểm tra số nguyên tố. Việc kiểm tra sẽ trả về là số hỗn hợp. Nhưng giả sử ta đã chọn a = 21 thì ta có 21^{55} mod

221 mod = 200; $(21^{55})^2$ mod 221 = 220, và việc kiểm tra sẽ trả về là không xác định được số. Cho biết rằng 221 có thể là số nguyên tố. Trong thực tế, với 218 số nguyên từ 2 đến 219 ta sẽ có 4 số trong 218 số đó sẽ trả về là không xác định được kiểu số là 21, 47, 174, 200.

+ Sử dụng thuật toán Miller-Rabin lặp

Làm thế nào ta có thể sử dụng thuật toán Miller-Rabin để xác định với một mức độ tin cậy cao hoặc không phải là một số nguyên tố. Có thể hiển thị cho một lẻ n không phải là số nguyên tố và một số ngẫu nhiên được chọn a với 1 < a < n-1. Xác suất mà việc kiểm tra trả về không xác định được (tức là không phát hiện ra rằng n không phải là số nguyên tố) là ít hơn $\frac{1}{4}$. Do đó nếu t khác với với giá trị của a đã chọn, xác suất mà tất cả sẽ qua việc kiểm tra (trả về là chưa đi đến kết quả) với n nhỏ hơn $(\frac{1}{4})^t$. Ví dụ, đối với t = 10, xác suất là một con số không phải ngoại lệ vượt qua tất cả việc kiểm tra là nhỏ hơn 10^{-6}. Do đó với a đủ lớn với giá trị của t, chúng ta có thể tin cậy rằng n là số nguyên tố nếu việc kiểm tra của Miller luôn trả về là chưa đi đến kết quả. Điều này cho chúng ta một cơ sở để xác định xem một số lẻ n là số nguyên tố với mức độ tin cậy hợp lý. Thủ tục như sau:

Lặp đi lặp lại việc kiểm tra n bằng cách sử dụng các giá trị ngẫu nhiên cho a. Nếu ở bất kì thời điểm nào việc kiểm tra trả về là số hỗn hợp, sau đó n được xác định là không phải số nguyên tố. Nếu việc kiểm tra tiếp tục trả về là chưa đi đến kết quả cho các thực nghiệm t, sau đó với a đủ lớn với giá trị của t, ta thừa nhận n là số nguyên tố.

2.1.4.2. Thuật toán xác định tính nguyên tố.

Trước năm 2002, không có phương pháp nào chứng minh hiệu quả tính nguyên tố của những con số lớn. Tất cả các thuật toán được sử dụng, bao gồm cả thuật toán phổ biến nhất là (Miller-Rabin) tạo ra một kết quả xác suất.

Năm 2002 (công bố 2002, xuất bản 2004) Agrawal, Kayal và Saxena đã phát triển một phương pháp xác định tương đối đơn giản thuật toán xác định có hiệu quả sẽ xác định xem một số lớn là số nguyên tố. Thuật toán được gọi là thuật toán AKS, dường như không hiệu quả như thuật toán Miller-Rabin.

+ Sự phân bố của số nguyên tố

Cần lưu ý có bao nhiêu con số có thể bị từ chối trước khi có 1 số nguyên tố được tìm thấy bằng cách sử dụng việc kiểm tra Miller-Rabin hoặc bất kì bài kiểm tra nào khác. Kết quả từ lý thuyết số được gọi là định lý số nguyên tố nói rằng số nguyên tố gần n được chia đều cho mỗi số trung bình mỗi ln(n) số nguyên. Như vậy trung bình người ta sẽ phải kiểm tra vị trí của số nguyên ln(n) "trước khi một số nguyên tố được tìm thấy. Bởi vì tất cả các số nguyên thậm chí có thể được loại bỏ

ngay lập tức, con số chính xác là 0,5ln(n)". Ví dụ, nếu một số nguyên tố ở vị trí độ lớn 2^{200} được tìm thấy. Sau đó khoảng 0,5ln(n) = 0,5ln(2^{200}) = 69 thử nghiệm sẽ cần thiết để tìm một số nguyên tố.

2.1.4.3. Định lý còn lại của Trung Hoa

Một trong những kết quả hữu ích nhất của lý thuyết số là định lý còn lại của Trung Hoa (CRT). Về bản chất (CRT) nói rằng: Có thể tìm lại các số nguyên trong một phạm vi nhất định từ dư lượng của chúng theo mod có một bộ các module tương đối quan trọng theo cặp.

Mười số nguyên trong Z_{10}, đó là số nguyên từ 0 đến 9. Có thể được khôi phục lại từ 2 dư lượng mod 2 và mod 5 là các phần tử tương đối nguyên tố của 10. Nói các dư lượng tồn tại của một chữ số thập phân x là $r_2 = 0$ và $r_5 = 3$.

Đó là: x mod 2 = 0 và x mod 5 = 3. Vì vậy x là số nguyên dương trong Z_{10}, chia cho 5 dư 3, suy ra x = 8

CRT có thể được thể hiện bằng nhiều cách. Dưới đây là một công thức hữu ích khác: M = $\Pi_{i=1}^{k} m_i$ trong đó: m_i là cặp tương đối nguyên tố gcd(m_i, m_j)=1 với $1 \leq i, j \leq k$ và $i \neq j$. Ta có thể biểu diễn bất kì số nguyên A trong Z_M bởi 1 bộ k có các phần tử trong Z_{m_i} sử dụng tương ứng sau:

$$A \leftrightarrow (a_1, a_2, ..., a_k) \qquad (2.11)$$

Trong đó: A ϵ Z_M ; $a_i \epsilon Z_{m_i}$ và a_i = A mod m_i với $1 \leq i \leq k$

+ *Hai khẳng định của CRT như sau:*

1. Lập bản đồ của phương trình (2.11) là một sự tương thích một – một giữa Z_M và sản phẩm Cartesia $Z_{m_1} \times Z_{m_2} \times .....\times Z_{m_k}$. Nghĩa là đối với mỗi số nguyên A sao cho $0 \leq A \leq M$, có 1 bộ k ($a_1, a_2, ..., a_k$) với $0 \leq a_i \leq m_i$ đại diên cho nó và đối với mỗi bộ ($a_1, a_2, ..., a_k$) có 1 số nguyên độc nhất A trong Z_M

2. Các hoạt động thực hiện trên các thành phần của Z_M có thể được thực hiện tương đương trên bộ k tương ứng bằng cách thực hiện các hoạt động độc lập trong mỗi vị trí ngang nhau trong hệ thống thích hợp.

- *Chứng minh khẳng định đầu tiên:* Chuyển đổi từ A sang ($a_1, a_2, ..., a_k$) là ràng buộc độc nhất. Nghĩa là, mỗi a_i được tính duy nhất là, a_i = A mod m_i. Tính A từ ($a_1, a_2, ..., a_k$), có thể được thực hiện như sau. Để cho M_j = M/m_j với $1 \leq i \leq k$.

Lưu ý $M_j = m_1 \times m_2 \times ..... m_{i-1} \times m_{i+1} .....\times m_k$

nên $M_i \equiv 0$ (mod m_j) với tất cả $j \neq i$.

Sau đó $c_i = M_i \times (M_i^{-1} \bmod m_i)$ với $1 \leq i \leq k$ (2.12)

Theo định nghĩa của M_i, nó tương đối nguyên tố với m_i và do đó có duy nhất một nhân nghịch đảo mod m_i. Công thức (2.12) được xác định rõ ràng và tạo ra một giá trị C_i . Bây giờ ta có thể tính:

$$A \equiv (\sum_{i-1}^{k} a_i c_i) (\text{mod } M) \qquad (2.13)$$

Để chứng minh giá trị của A là đúng ở công thức (2.13), ta thấy a_i = A mod m_i với $1 \leq i \leq k$

Lưu ý: $C_j \equiv M_j \equiv 0$ (mod m_i) nếu $j \neq i$ và $C_i \equiv 1$ (mod m_i). Sau đó a_i = A mod m_i.

- *Chứng minh khẳng định thứ 2 của CRT:* Liên quan đên các phép tính số học từ các quy tắc cho số học modul. Nghĩa là, khẳng định thứ hai có thể được ghi lại như sau:

Nếu $A \leftrightarrow (a_1, a_2 \ldots\ldots a_k)$ và $B \leftrightarrow (b_1, b_2 \ldots\ldots b_k)$

Thì $(A + B) \bmod M \leftrightarrow ((a_1+b_1) \bmod m_1, \ldots\ldots\ldots (a_k + b_k) \bmod m_k)$

$(A – B) \bmod M \leftrightarrow ((a_1 – b_1) \bmod m_1 , \ldots\ldots\ldots (a_k – b_k) \bmod m_k)$

$(A \times B) \bmod M \leftrightarrow ((a_1 \times b_1) \bmod m_1), \ldots\ldots\ldots\ldots (a_k \times b_k) \bmod m_k)$

Một trong những tính năng hữu ích của định lý dư lượng của Trung Hoa là nó cung cấp một cách để thực hiện (có tiềm năng rất lớn) số mod M trong điều kiện của những bộ với số lượng nhỏ hơn. Điều này có thể hữu ích khi M là 150 chữ số trở lên. Tuy nhiên, lưu ý rằng nó là cần thiết để biết các phần tử của M. Ví dụ, để diễn tả 973 mod 1813 nhu một cặp số mod 37 và 49. Ta xác định $m_1 = 37$; $m_2 = 49$; M = 1813; A=973. Chúng ta có $M_1 = 49$ và $M_2 = 37$, ta dùng thuật toán Euclide mở rộng. Ta tính $M_1^{-1} = 34$ mod m_1 và $M_2^{-1} = 4$ mod m_2 (Lưu ý: Ta chỉ cần tính mỗi M_i và mỗi M_i^{-1} một lần). Lấy dư lượng mod 37 và 49, biểu thức của 973 là (11, 42) bởi vì 973 mod 37 = 11 và 973 mod 49 = 42.

Bây giờ giả sử ta muốn thêm 678 vào 973. Ta sẽ xử lý bộ (11,42) như sau:

Đầu tiên ta tính 678 ↔ (678 mod 37, 678 mod 49)=(12,41). Sau đó ta thêm những bộ số vào và giảm ((11+12) mod 37, (42+41) mod 49) = (23,34)

Để xác minh rằng điều này chính xác, ta tính:

$$(23,34) \Leftrightarrow (a_1 M_1 M_1^{-1} + a_2 M_2 M_2^{-1}) \bmod M$$
$$= [(23)(49)(34) + (34)(37)(4)] \bmod 1813$$
$$= 43350 \bmod 1813$$
$$= 1651$$

Và kiểm tra nó bằng với (973 + 678) mod 1813 = 1651. Nhớ rằng phép lấy đại số ở trên; M_i^{-1} là phép nhân nghịch đảo của M_1 mod m_1 mod M_2^{-1} là phép nhân nghịch đảo của M_2 mod m_2.

Giả sử ta muốn nhân 1651 (mod 1813) với 73. Chúng ta nhân (23,34) với 73 và giảm đi có ((23 × 73) mod 37, (34 × 73) mod 49) = (14, 32). Ta chứng minh như sau:

(14, 32)⇔ [(14)(49)(34) + (32)(37)(4)] mod 1813 = 856

=(1651 × 73) mod 1813

2.1.5. Logarithms rời rạc.

Logarithms rời rạc là cơ bản của một số thuật toán khóa công khai bao gồm trao đổi khóa Diffie_Hellman và thuật toán chữ kí số (DSA). Sau đây là tổng quát ngắn gọn về logarithms rời rạc.

2.1.5.1. Khả năng của một số nguyên (mod n).

Ta có công thức $a^{\emptyset(n)} \equiv 1 \pmod n$

∅(n): là số những số nguyên nhỏ hơn n và tương đối nguyên tố với n

Biểu thức tổng quát hơn :

$a^m \equiv 1 \pmod n$ (2.14)

Nếu a và n tương đối nguyên tố với nhau, thì có ít nhất 1 số nguyên m thỏa mãn (2.14). Cụ thể là: M=∅(n). Số mũ m cho số phương trình (2.14) được đề cập đến trong một số cách sau:

- Thứ tự của a (mod n)
- Số mũ của a(mod)
- Chiều dài thời hạn tạo ra bởi a

Ví dụ: Khả năng của 7, (mod 19)

$7^1 \equiv 7 \pmod{19}$

$7^2 = 49 = 2 \times 19 + 11 \equiv 11 \pmod{19}$

$7^3 = 343 = 18 \times 19 + 1 \equiv 1 \pmod{19}$

$7^4 = 2401 = 126 \times 19 + 7 \equiv 7 \pmod{19}$

$7^5 = 16807 = 884 \times 19 + 11 \equiv 11 \pmod{19}$

Không có điểm để tiếp tục vì trình tự lặp đi lặp lại.

Điều này có thể chứng minh bằng cách rằng :

$7^3 \equiv 1 \pmod{19}$ và vì thế $7^{3+j} = 7^3 \times 7^j \equiv 7^j \pmod{19}$

Do đó bất kì 2 khả năng nào của 7 có số mũ khác nhau bằng 3 (hoặc 1 bội số của 3) tương ứng với nhau (mod 19), vài chiều dài của khoảng thời gian tỉ số mũ nhỏ nhất của m:

$7^m \equiv 1 \pmod{19}$

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	17	7	8	1	12	11	17	7	8	1	12	11	17	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

Bảng 2.3 Khả năng của những số nguyên, (mod 19)

Bảng trên cho thấy tất cả các khả năng của a, mod 19 cho tất cả số dương a<19. Chiều dài của chuỗi cho mỗi giá trị cơ bản được đánh dấu màu. Lưu ý những điều dưới đây:

- Tất cả các trình tự kết thúc bằng 1.

- Chiều dài của 1 dãy phân chia $\emptyset(19) = 18$. Tức là 1 số nguyên của các dãy xảy ra trong mỗi hàng của bảng.

- Một số dãy có chiều dài là 18. Trong trường hợp này, người ta nói rằng số nguyên cơ bản tạo ra tập các số nguyên không theo mod 19. Mỗi số nguyên này được gọi là gốc nguyên tố của mod 19.

Nói chung, chúng ta có thể nói rằng số mũ lớn nhất của một số (mod n) là $\emptyset(n)$. Nếu một số nằm trong vị trí này, nó được gọi là gốc nguyên tố của n. Tầm quan trọng của khái niệm này là nếu a là gốc nguyên tố của n, thì khả năng của nó: a, a^2, ..., $a^{\emptyset(n)}$ là khác biệt (mod n) và tất cả tương đối nguyên tố với n. Đặc biệt, với một số nguyên tố p, nếu a là một gốc nguyên tố của p, thì: a, a^2, ..., a^{p-1} là khác biệt (mod p). Cho số nguyên tố 19, gốc nguyên tố của nó là 2, 3, 10, 13, 14, và 15. Không phải tất cả các số nguyên đều có gốc nguyên tố. Trong thực tế, chỉ những

số nguyên với gốc nguyên tố dạng: 2, 4, p^{α}, $2p^{\alpha}$. Trong đó: p là bất kì số nguyên tố lẻ và α là một số nguyên dương.

2.1.5.2. Logariths cho mô đun số học

Với số thực, chức năng logarithms là nghịch đảo của lũy thừa. Một chức năng tương tự tồn tại cho mô đun số học. Ta xem xét ngắn gọn các tính chất của các logarithms thông thường của một số được định nghĩa là khả năng mà một số dương căn bản (trừ 1) được lũy thừa lên để bằng một số. Tức là, đối với cơ số x và cho một giá trị y: y = $x^{log_x(y)}$

Các thuộc tính của logarithms bao gồm:

$log_x(1) = 0$

$log_x(yz) = log_x(y) + log_x(z) = 0$ (2.15)

$log_x(x) = 1$

$log_x(y)^r$ =r × $log_x(y)$ (2.16)

Hãy xem xét 1 gốc nguyên tố cho một số nguyên tố p. Sau đó chúng ta biết rằng các khả năng của a từ 1 đến (p – 1) tạo ra mỗi số nguyên: từ 1 đến (p – 1) chính xác mỗi lần. Ta biết bất cứ số nào đáp ứng được: b≡ r (mod p) với 1 số r; 0 ≤ r ≤ (p-1) theo định nghĩa mô đun số học. Sau đó đối với bất kì số nguyên b và gốc nguyên tố a của số nguyên tố p, chúng ta có thể tìm thấy một số mũ duy nhất i như vậy: $b \equiv a^i$ (mod p); trong đó 0 ≤ i ≤ (p-1). Số mũ i này được gọi là logarithms rời rạc của số b cho cơ số a (mod p). Chúng ta biểu thị giá trị: d × $log_{a,p}$(b)[9]

Lưu ý:

d × $log_{a,p}(1) = 0$; bởi vì a^0 mod p = 1 mod p = 1 (2.17)

d × $log_{a,p}(0) = 1$; bởi vì a^1 mod p = a mod p = a (2.18)

Đây là ví dụ sử dụng một mô đun không trọng số: n= 9. Ở đây ∅(n) = 6 và a = 2 là gốc nguyên tố. Chúng ta tính khả năng khác nhâu của a được tìm thấy:

$2^0 = 1$ $2^2 = 4$ $2^4 \equiv 7$ (mod 9) $2^6 \equiv 1$ (mod 9)

$2^1 = 2$ $2^3 = 8$ $2^5 \equiv 5$ (mod 9)

Điều này cho chúng ta bảng sau đây của các số với logarithms rời rạc (mod 9) với gốc a = 2

Logarithms	0	1	2	3	4	5
Number	1	2	4	8	7	5

Làm nó cho dễ dàng để có được logarithms rời rạc của một số nhất định, ta sắp xếp lạ bảng:

Number	1	2	4	5	7	8

Logarithms 0 1 2 5 4 3

Bây giờ ta xét:

x = $a^{d \times log_{a,p}(x)}$ mod p

y = $a^{d \times log_{a,p}(y)}$ mod p

xy = $a^{d \times log_{a,p}(xy)}$ mod p

Sử dụng các quy tắc của phép nhân mô đun:

xy mod p = [(x mod p)×(y mod p)] mod p

$a^{d \times log_{a,p}(xy)}$ mod p = [($a^{d \times log_{a,p}(x)}$ mod p)($a^{d \times log_{a,p}(y)}$ mod p)]mod p

= ($a^{d \times log_{a,p}(x) + d \times log_{a,p}(y)}$)mod p

Nhưng giờ xem xét định lý Euler nói rằng: Đối với mỗi a và n tương đối nguyên tố với nhau thì $a^{\emptyset(n)} \equiv 1$ (mod n). Bất kì sô nguyên dương z có thế được thể hiện dưới dạng z = q + k∅(n) với 0 ≤ q ≤ ∅(n).

Như vậy theo định lý Euler: $a^z \equiv z^q$ (mod n) nếu z ≡ q (mod ∅(n))

Áp dụng điềunày với phương trình ở trên ta có:

$d \times log_{a,p}(xy) \equiv [d \times log_{a,p}(x) + d \times log_{a,p}(y)]$ (mod ∅(n))

Điều này chúng tỏ sự tương tự giữa logarithms và logarithms rời rạc.

2.1.5.3. Tính toán logarithms rời rạc

Xét phương trình: $y = g^x$ mod p

Cho g, x, p tính y. Vấn đề là ta phải nhân lặp đi lặp lại và các thuật toán tồn tại để đạt được hiệu quả lớn hơn. Tuy nhiên, cho g, y, p rất dễ tính x (lấy logarithms rời rạc): $e^{((lnp)^{1/3}(ln(lnp))^{2/3}))}$. Nhưng lại không khả thi với những con số nguyên tố lớn. Bảng (2.4) được rút ngắn bảng (2.3). Cho thấy các tập logarithms rời rạc có thể được định nghĩa cho mod 19.

a. Logarithms rời rạc của cơ số 2 , mod 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$log_{2,19}(a)$	18	1	13	2	16	14	6	3	8	17	12	15	5	7	11	4	10	9

b. Logarithms rời rạc của cơ số 3 , mod 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$log_{3,19}(a)$	18	7	1	14	4	8	6	3	2	11	12	15	17	13	5	10	16	9

c. Logarithms rời rạc của cơ số 10, mod 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$log_{10,19}(a)$	18	17	5	16	2	4	12	15	10	1	6	3	13	1	7	14	8	9

d. Logarithms rời rạc của cơ số 13, mod 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$log_{13,19}(a)$	18	11	17	4	14	10	12	15	16	7	6	3	1	5	13	8	2	9

e. Logarithms rời rạc của cơ số 14, mod 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$log_{14,19}(a)$	18	13	7	8	10	2	6	3	14	5	12	15	11	1	17	16	4	9

f. Logarithms rời rạc của cơ số 15, mod 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$log_{15,19}(a)$	18	5	11	10	8	16	12	15	4	13	6	3	7	17	1	2	14	9

Bảng 2.4. Khả năng của những số nguyên tố, (mod 19)

2.2 Mật mã khóa công khai và RSA.

Sự phát triển của mật mã khóa công khai là lớn nhất và có lẽ là cuộc cách mạng thực sự trong toàn bộ lịch sử của mật mã học. Từ những khởi đầu mới nhất đến thời hiện đại, hầu như tất cả các hệ thống mật mã đều dựa trên các công cụ cơ bản để thay thế và hoán vị. Sau một thiên niên kỉ làm việc với các thuật toán có thể được tính bằng tay, một tiến bộ lớn trong mã hóa đối xứng xảy ra và với sự phát triển của máy mã hóa và giải mã tự động . Các máy tính cho phép sự phát triển của hệ thống mật mã cực kì phức tạp. Với sự sẵn có của máy tính, thậm chí các hệ thống phức tạp hơn đã được đưa ra, nổi bật nhất là nỗ lực của Lucifer tại IBM đạt được kết quả là tiêu chuẩn mã hóa dữ liệu (DES). Nhưng cả hai máy tính và DES mặc dù đại diện cho cho những sự tiến bộ nhưng vẫn dựa vào các công cụ thay thế và hoán vị.

Thuật toán mật mã khóa công khai được dựa trên các chức năng toán học thay vì thay thế và hoán vị. Quan trọng hơn, mật mã khóa công khai là không đối xứng bao gồm việc sử dụng hai khóa riêng biệt, trái ngược với mã hóa đối xứng, chỉ sử dụng một khóa. Trước khi tiến hành, ta nên đề cập đến một số khái niệm sai lầm phổ biến liên quan đến mã hóa khóa công khai. Một quan niệm sai lầm như vậy là: Mật mã khóa công khai an toàn hơn từ việc phân tích mật mã hơn mã hóa đối xứng. Trong thực thế, bảo mật của bất kì chương trình mã hóa nào phụ thuộc vào độ dài của khóa và công việc tính toán liên quan đến phã vỡ một mật mã. Không có gì về nguyên tắc của mã hóa đối xứng hoặc mã hóa khóa công khai mà làm cho những cấp trên khác chống lại phân tích mật mã.

Một quan niệm sai lầm thứ hai khác là mã hóa khóa công khai là một kĩ thuật đa mục đích đã làm mã hóa đối xứng lỗi thời. Ngược lại, vì những tính toán trên cơ sở tính toán của các chương trình mã hóa khóa công khai hiện tại, dường như không có khả năng dự đoán được rằng mã hóa đối xứng sẽ bị bỏ đi. Như một trong những nhà phát minh mã hóa khóa công khai đã cho rằng "hạn chế của mật mã khóa công khai để quản lý chính và các ứng dụng chữ ký gần như được chấp nhận rộng rãi."

Cuối cùng, có một cảm giác rằng phân phối chủ chốt là tầm thường khi sử dụng mã hóa khóa công khai, so với việc bắt tay phức tạp hơn với các trung tâm phân phối chính cho mã hóa đối xứng. Trên thực tế, một số dán giao thức là cần thiết, thường bao gồm một tác nhân trung tâm và các thủ tục liên quan không đơn giản hơn cũng không hiệu quả hơn các thủ tục cần thiết cho mã hóa đối xứng.

Chúng ta nhìn vào khái niệm của nó, khái niệm về kĩ thuật này đã được phát triển và công bố trước khi nó được chấp nhận để thực hiện. Tiếp theo, ta kiểm tra các thuật toán RSA, thuật toán mã hóa và giải mã quan trọng nhất đã được chứng minh là khả thi đối với mật mã khóa công khai.

Khóa bất đối xứng	-Hai khóa liên quan (khóa công khai và khóa cá nhân) được sử dụng để thực hiện các thao tác bổ sung, chẳng hạn mã hóa và giải mã hoặc tạo chữ kí và xác minh chữ ký.
Chứng thực khóa công khai	-Tài liệu kĩ thuật số và chữ kí số được cấp bởi các cơ quan chứng nhận liên kết tên của người đăng kí với khóa công khai. Chứng chỉ cho biết người đăng kí đã xác định trong chứng nhận đã được kiểm soát và truy cập vào khoá cá nhân tương ứng.
Thuật toán mật mã khóa công khai (bất đối xứng)	-Thuật toán mã hóa sử dụng hai khóa liên qua(khóa công khai và khóa cá nhân). Hai khóa này có đặc tính là lấy khóa cá nhân từ khóa công khai là không khả thi về mặt tính toán.
Cơ sở hạ tầng công cộng (PKI)	-Một bộ chính sách, quy trình, nền tảng máy chủ, phần mềm và máy trạm làm việc được sử dụng cho mục đích quản lý các chứng chỉ và các cặp khóa công khai và cá nhân, bao gồm khả năng phát hành, duy trì và thu hồi giấy chứng nhận khóa công khai.

Bảng 2.5 Các thuật ngữ liên quan đến mã hóa bất đối xứng

2.2.1. Nguyên tắc của các hệ thống mật mã khóa công khai.

Khái niệm mật mã khóa công khai phát triển từ một nỗ lực tấn công hai trong số những vấn đề khó khăn nhất liên quan đến mã hóa đối xứng. Phân phối khóa, phân phối khóa theo mã hóa đối xứng yêu cầu thứ nhất là: Hai người giao tiếp đã chia sẻ một khóa bằng cách nào đó đã được phân phối cho họ. Thứ hai là: Việc sử dụng một trung tâm phân phối chính

2.2.1.1. Hệ thống mật mã khóa công khai

Các thuật toán bất đối xứng dựa vào khóa cho mã hóa và các thuật toán khác nhưng khóa có liên quan cho giải mã. Các thuật toán này có đặc điểm quan trọng sau đây:

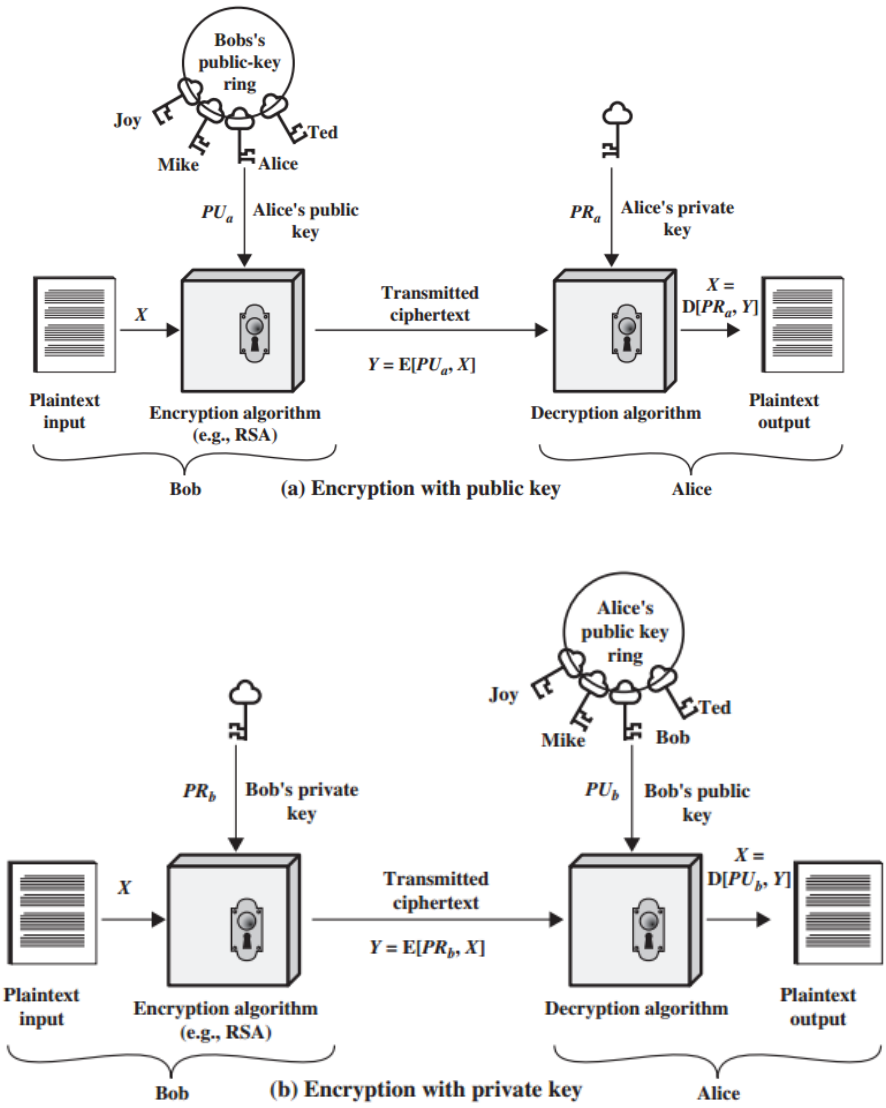
- Tính toán không thể thực hiện được để xác định khóa giải mã chỉ được đưa ra kiến thức về thuật toán mật mã và khóa mật mã.

Ngoài ra, một số thuật toán, chẳng hạn như RSA cũng có các đặc điểm sau đây:

- Một trong hai khóa liên quan dùng để mã hóa, khóa còn lại được dùng để giải mã.

Một chương trình mã hóa khóa công khai có sáu thành phần (Hình 2.1. (a))

- Bản rõ (plaintext): Đây là thông điệp có thể đọc hoặc dữ liệu được đưa vào thuật toán như đầu vào.



Hình 2.1 Mật mã khóa công khai

Thuật toán mã hóa (Encryption algorithm): Thuật toán mã hóa thực hiện các phép biến đổi khác nhau trên plaintext.

Khóa công khai và khóa cá nhân (Public and private keys): Đây là một cặp khóa đã được chọn để nếu một trong hai cái được sử dụng để mã hóa, khóa còn lại được sử dụng để giải mã. Các phép biến đổi chính xác được thực hiện bởi thuật toán phục thuộc vào khóa công khai hoặc khóa cá nhân được cung cấp như đầu vào.

Bản mã (Ciphertext): Đây là thông điệp được mã hóa, tạo ra như đầu ra. Nó phụ thuốc vào bản rõ và chìa khóa. Đối với một thông điệp nhất định, hai khóa khác nhau sẽ tạo ra hai bản mã khác nhau.

Thuật toán giải mã (decryption algorithm): Thuật toán này chấp nhận bản mã và kết hợp chìa khóa và tạo ra bản rõ.

2.2.1.2. Các bước cần thiết

1. Mỗi người dùng tạo ra một cặp khóa để sử dụng mã hóa và giải mã tin nhắn.

2. Mỗi người sử dụng đặt một trong hai khóa trong thanh ghi công khai hoặc tập tin truy cập khác. Đây là khóa công khai. Khóa cá nhân được giữ kín. Như sơ đồ hình (2.1. (a)). Mỗi người dùng, duy trì 1 bộ sưu tập các khóa công khai có được từ những người khác.

3. Nếu Bob muốn gửi một tin nhắn bí mật đến Alice, Bob mã hóa tin nhắn bằng khóa công khai của Alice.

4. Khi Alice nhận tin nhắn, cô ấy giải mã nó bằng khóa cá nhân. Không người nhận khác có thể giải mã tin nhắn vì chỉ có Alice biết khóa riêng của Alice.

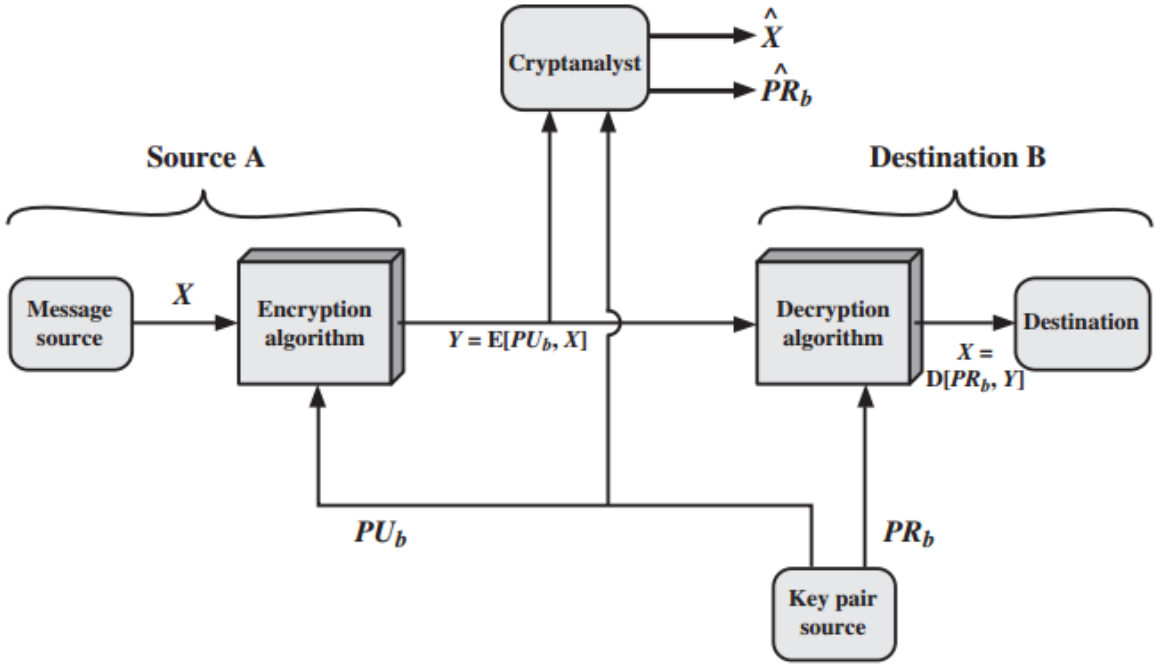
Với cách tiếp cận này, tất cả người tham gia đều có quyền truy cập vào khóa công khai, với các khóa riêng được tạo bởi mỗi người tham gia và do đó không bao giờ được phân phối. Miễn là khóa cá nhân của người dùng vẫn được bảo vệ bí mật, thông tin đến được an toàn. Vào bất kì tời điểm nào, hệ thống có thể thay đổi khóa riêng của nó và xuất bản khóa công khai đi kèm để thay thế khóa công khai cũ của nó.

Mã hóa thông thường	Mã hóa khóa công khai
Cần cho hoạt động: 1. Cùng một thuật toán với một khóa được sử dụng để mã hóa và giải mã 2. Người gửi và người nhận phải chia sẻ thuật toán và khóa	Cần cho hoạt động: 1. Một thuật toán được sử dụng để mã hóa và một thuật toán liên quan để giải mã với một cawoj khóa, một cho mã hóa và một cho giải mã. 2. Người gửi và người nhận phải có một cặp khóa khấp nhau.
Cần cho an ninh: 1. Chìa khóa phải được giữ kín 2. Nó phải không thể hoặc ít nhất là không thực tế (áp dụng) để giải mã một tin nhắn nếu khóa được giữ bí mật. 3. Kiến thức về thuật toán với mật mã hóa không đủ để xác định khóa.	Cần cho an ninh: 1. Một trong hai khóa phải được giữ bí mật. 2. Phải là không thể hoặc ít nhất là không thực tế đối với giải mã một tin nhắn nếu một trong các khóa giữ bí mật.

	3. Kiến thức về thuật toán cộng với một trong các khóa cộng với các mẫu mã hóa phải đủ để xác định khóa khác.

Bảng 2.6. Mã hóa thông thường và mã hóa khóa công khai.

Bảng 2.6. tóm tắt một số khía cạnh quan trọng của mã hóa đối xứng và mã hóa khóa công khai. Để phân biệt giữa hai cái này, chúng ta thấy khóa được sử dụng trong mã hóa đối xứng là một khóa bí mật. Còn hai khóa được sử dụng trong mã hóa bất đối xứng được gọi là khóa công khai và khóa cá nhân. Khóa cá nhân được giữ bí mật, nhưng nó được gọi là khóa cá nhân chứ không phải là một khóa bí mật để tránh nhầm lẫn với mã khóa đối xứng.



Hình 2.2. Hệ mật mã khóa công khai: Bí mật

Ta xem xét kĩ hơn các yếu tố cần thiết của mã khóa khóa công khai (hình 2.2.). Có một số nguồn A, đó là tạo ra các thông điệp trong bản rõ X = [X_1, X_2, ..., X_M]. Các phần tử M của X là các chữ trong số bảng chữ cái hữu hạn. Thông điệp được dùng cho đích B, B tạo ra một cặp khóa liên quan; khóa công khai PU_b và khóa cá nhân PR_b. PR_b chỉ được biết đến ở B, trong khi PU_b công khai có sẵn và do đó có thể truy cập bởi A. Với thông điệp X và khóa mã hóa đầu vào PU_b, A hình thành bản mã Y = [Y_1, Y_2, ..., Y_N]:

$$Y = E(PU_b, X)$$

Người nhận có kế hoạch, sở hữu khoá cá nhân kết hợp, có thể đảo ngược sự chuyển đổi:

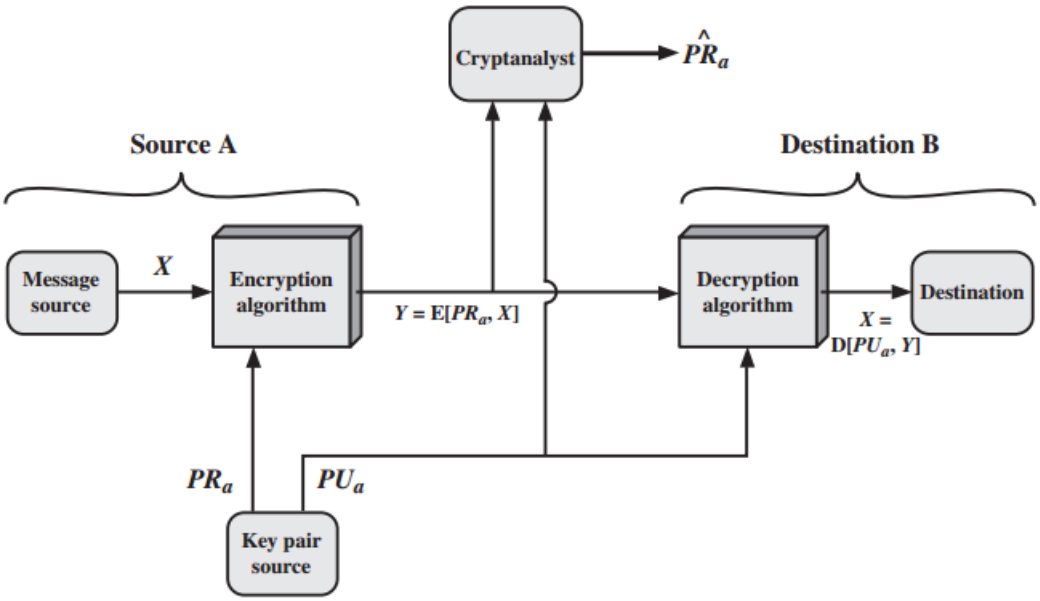
$$X = D(PR_b, Y)$$

Một đối tượng, quan sát Y và có quyền truy cập và PU_b, nhưng không có quyền truy cập vào PR_b hoặc X, phải cố gắng khôi phục X và/ hoặc PR_b. Giả sử đối phương không có kiến thức về các thuật toán mã hóa (E) và giải mã (D). Nếu đối phương chỉ quan tâm đến thông điệp đặc biệt này, thì trọng tâm của nỗ lực là khôi phục X bằng cách tạo ra ước lượng bản rõ $\hat{X}$. Thông thường, đối phương cũng quan tâm đến khả năng đọc được các tin nhắn trong tương lai, trong trường hợp đó, một nỗ lực để khôi phục PR_b bằng cách tạo ra ước tính $\hat{PB}_b$.

Như đề cập trước đó rằng, một trong hai khóa có thể được sử dụng cho mã hóa, và một cách khác được sử dụng để giải mã. Điều này cho phép thực hiện một lược đồ mật mã hóa khác. Hình 2.1. (b) và hình 2.3. cho thấy việc sử dụng khóa công khai mã hóa để cung cấp chứng thực:

$$Y = E(PR_a, X)$$

$$X = D(PU_a, Y)$$



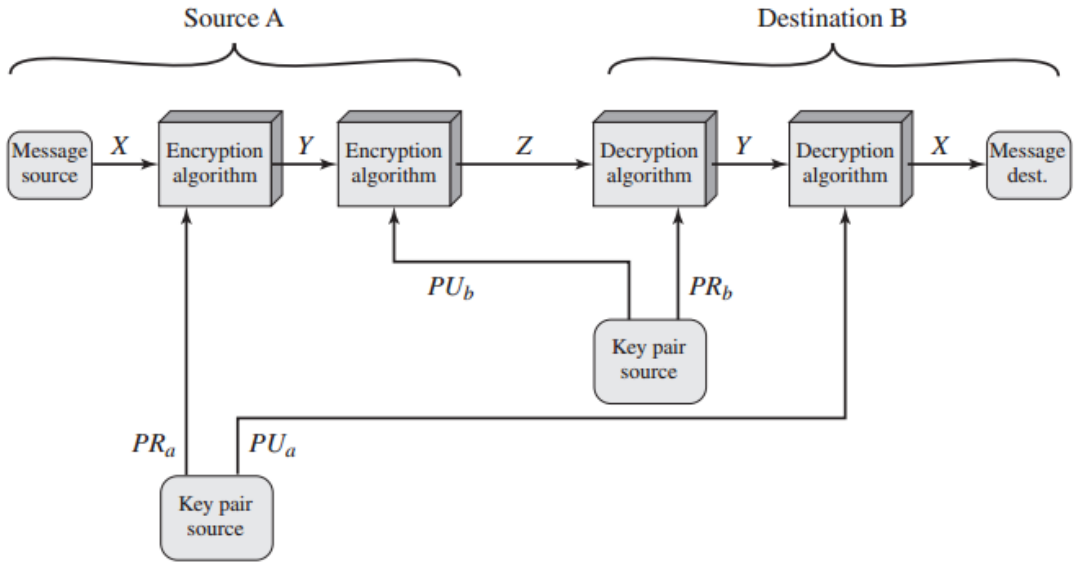
Hình 2.3 Hệ mã khóa công khai: Xác thực

Trong trường hợp này, A chuẩn bị một thông điệp đến B và mã hóa nó bằng cách sử dụng khóa riêng của A trước khi truyền nó. B có thể giải mã thông điệp bằng khóa công khai của A. Bởi vì tin nhắn đã được mã hóa bằng khóa riêng tư của A, chỉ A có thể chuẩn bị thông điệp. Do đó, toàn bộ tin nhắn được mã hóa hoạt động như một chữ kí số. Ngoài ra, không thể thay đổi thông báo mà không cần truy cập vào thông tin cá nhân của A, chính vì vậy thông điệp được xác định cả về nguồn và về dữ liệu chính thực.

Trong sơ đồ trước, toàn bộ tin nhắn được mã hóa, mặc dù cả tác giả và nội dung, đòi hỏi rất nhiều lưu trữ. Mỗi tài liệu phải được giữ nguyên để sử dụng cho mục đích thực tiễn. Một bản sao cũng phải đươc lưu trữ trong bản mã để nguồn gốc và nội dung có thể được xác định trong trường hợp có tranh chấp. Một cách hiệu quả

hơn để đạt được kết quả tương tự là mã hóa một khối nhỏ các bít là một chức năng của tài liệu. Một khối như vậy được gọi là chứng thực, phải có tính chất đó là không thể thay đổi tài liệu và không thay đổi người xác thực. Nếu trình xác thực được mã hóa với khóa cá nhân của người gửi, nó sẽ đóng vai trò như một chữ kí xác minh nguồn gốc, nội dung và trình tự.

Điều quan trọng là phải nhấn mạnh rằng quá trình mã hóa mô tả trong hình 2.1. (b) và hình 2.3. không cung cấp tính bảo mật. Đó là thông điệp đang được gửi đi được an toàn trước sự thay đổi nhưng không phải từ nghe trộm. Điều này rõ ràng trong trường hợp của một chữ kí dựa trên một phần của tin nhắn, bởi vì phần còn lại của tin nhắn được truyền đi trong rõ ràng. Ngay cả trong trường hợp mã hóa hoàn chỉnh như trong hình 2.3, không có bảo vệ bí mật vì bất kì người quan sát có thể giải mã tin nhắn bằng cách sử dụng khóa công khai của người gửi. Tuy nhiên, nó cung cấp cả chức năng xác thực và bảo mật thông qua hai lần khóa công khai như hình 2.4. dưới đây:



Hình 2.4. Hệ mật mã khóa công khai: Xác thực và bí mật.

$Z = E(PU_b, E(PR_a, X))$

$X = D(PU_a, D(PR_b, Z))$

Trong trường hợp này, ta bắt đầu như trước bằng cách mã hóa một tin nhắn, sử dụng khóa riêng tư của người gửi, điều này cung cấp chữ kí số. Tiếp theo, ta lại mã hóa, sử dụng khóa công khai của người nhận. Các bản mã cuối cùng chỉ có thể được giải mã chỉ bởi người nhận chỉ định, một mình người đó có khóa cá nhân kết hợp. Vì vậy, bảo mật được cung cấp của cách tiếp cận này là thuật toán khóa công khai, phức tạp, phải được thực hiện bốn lần thay vì hai lần trong mỗi giao tiếp.

2.2.1.3. Ứng dụng của các hệ thống mật mã khóa công khai

Trước tiên, ta cần nắm một khía cạnh của các hệ thống mật mã khóa công khai, mà có thể dẫn tới sự nhầm lẫn. Các hệ thống khóa công khai được đặc trưng bởi việc sử dụng thuật toán mật mã với hai khóa, một được giữ riêng và một được công khai có sẵn. Tùy thuộc vào ứng dụng, người gửi sử dụng khóa cá nhân của người gửi hoặc khóa công khai của người nhận, hoặc cả hai, để thực hiện một số loại chức năng mật mã. Nói chung, ta có thể phân loại việc sử dụng các hệ thống mật mã khóa công khai thành ba loại:

- Mã hóa/giải mã: Người gửi mã hóa một tin nhắn với khóa công khai của người nhận.

- Chữ kí số: Người gửi "kí" một tin nhắn có khóa riêng của nó. Việc kí kết được thực hiện bằng một thuật toán mật mã được áp dụng cho tin nhắn hoặc một khối dữ liệu nhỏ là chức năng của tin nhắn.

- Trao đổi khóa: Hai bên hợp tác trao đổi khóa phiên. Có thể có một số cách tiếp cận khác nhau, bao gồm các khóa cá nhân của một hoặc cả hai bên.

Một số thuật toán phù hợp cho cả 3 ứng dụng, trong khi một số khác chỉ có thể được sử dụng cho một hoặc 2 ứng dụng này. Bảng 2.7. chỉ ra các ứng dụng được hỗ trợ bởi các thuật toán.

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

Bảng 2.7 Ứng dụng cho các hệ thống mật mã công khai

2.2.1.4. Yêu cầu đối với mật mã khóa công khai

Hệ thống mã hóa được minh họa trong hình 2.2. đến hình 2.4. phụ thuộc vào thuật toán mật mã dựa trên hai khóa liên quan. Difie và Hellman đã đưa ra hệ thống này mà không cần chứng minh rằng các thuật toán như vậy tồn tại. Tuy nhiên, họ đã đưa ra các điều kiện mà các thuật toán như vậy phải đáp ứng.

1. Đó là tính dễ dàng cho một bên B để tạo ra một cặp (khóa công khai PU_b, khóa cá nhân PR_b).

2. Nó dễ tính toán cho người gửi A, biết khóa công khai và thông điệp được mã hóa, M, để tạo ra bản mã tương ứng. $C= E(PU_b, M)$

3. Đó là tính dễ dàng cho người nhận B để giải mã các bản mã kết quả sử dụng khóa riêng để khôi phục lại thông báo ban đầu.

$$M = D(PR_b,C)=D[PR_b,E(PU_b,M)]$$

4. Nó là không thể thực hiện tính toán được đối với kẻ tấn công, biết được khóa công khai PU_b để xác định khóa cá nhân PR_b

5. Nó là không thể thực hiện tính toán được đối với kẻ tấn công, biết được khóa công khai PU_b và 1 bản mã C, để khôi phục thông điệp ban đầu M.

6. Hai khóa có thể được áp dụng theo thứ tự

$$M = D[PU_b,E(PR_b,M)]=D[PR_b,E(PU_b,M)]$$

Đây là những yêu cầu, chứng minh bằng thực thế là chỉ có một vài các thuật toán (RSA, mật mã Ellip Curve, Diffie-Hellman, DSS) đã nhận được sự chấp nhận rộng rãi trong vài thập kỉ kể từ khi khái niệm mã hóa khóa công khai được đề xuất.

Trước khi giải thích lí do tại sao các yêu cầu quá khắt khe, trước tiên ta nên làm lại chúng. Giảm các yêu cầu để cần một cửa chức năng một chiều. Một chức năng một chiều là ánh xạ một miền vào một dải sao cho mỗi giá trị hàm có một sự nghịch đảo duy nhất, với điều kiện tính toán của hàm là dễ dàng, khi tính toán ngược lại là không khả thi.

Y = f(X) dễ dàng $X = f^{-1}(Y)$ không khả thi

Nói chung, dễ dàng được định nghĩa để có nghĩa là một vấn đề có thể được giải quyết trong thời gian đa thức như là một chức năng của chiều dài đầu vào. Vì vậy, nếu chiều dài của đầu vào là n bit, thì thời gian tính toán hàm tương ứng với n^a, trong đó a là hằng số cố định. Như là thuật toán được cho là thuộc về lớp P. Thuật ngữ không khả thi là một khái niệm mở. Nói chung, chúng ta có thể nói rằng một vấn đề là không khả thi nếu nỗ lực để giải quyết nó phát triển nhanh hơn thời gian đa thức như là một chức năng của kích thước đầu vào.

Ví dụ: Nếu chiều dài của đầu vào là n bit và thời gian để tính toán các chức năng là tỉ lệ thuận với 2^n, vấn đề được xem là không thể khả thi. Thật không may, rất khó để xác định nếu một thuật toán thể hiện cụ thể sự phức tạp này. Hơn nữa, các khái niệm truyền thống của sự tính toán phức tạp tập trung vào sự phức tạp của thuật toán hoặc trường hợp xấu nhất. Những biện pháp này không phù hợp với mật mã, điều này đòi hỏi không thể đảo ngược được chức năng của hầu hết các đầu vào, không phải đối với trường hợp xấu nhất hoặc thậm chí là trường hợp trung bình.

Bây giờ ta chuyển sang định nghĩa của một chức năng cửa bẫy, đó là dễ dàng để tính toán theo một hướng và không thể tính toán ở một hướng khác trừ khi một số thông tin bổ sung được biết đến. Với các thông tin bổ sung nghịch đảo có thể được tính bằng thời gian đa thức. Ta có thể tóm tắt như sau: Một chức năng cửa bẫy là một gốc có chức năng đảo ngược f_k, như vậy:

$Y = f_k(X)$ dễ dàng, nếu biết k và X

$X = f_k^{-1}(Y)$ dễ dàng, nếu biết k và Y

$X = f_k^{-1}(Y)$ không khả thi, nếu biết Y và không biết k

Do đó, sự phát triển thực tế của một mã hóa khóa công khai phụ thuộc vào sự phát triển của một chức năng cửa bẫy thích hợp.

2.2.1.5. Phân tích mã khóa công khai

Giống như đối với mã hóa đối xứng, một chương trình mã hóa khóa công khai dễ bị tổn thương bởi brute-force. Các biện pháp đối phó như sau: Sử dụng một khóa lớn. Tuy nhiên, có một sự cân bằng để được xem xét. Các hệ thống khóa công khai phụ thuộc vào việc sử dụng một loại chức năng toán học không thể đảo ngược. Tính phức tạp của tính toán các chức năng này có thể không quy mô tuyến tính với số lượng bit trong khóa nhưng phát triển nhanh hơn. Do đó, kích thước khóa phải đủ lớn đê làm cho cuộc tấn công brute-force không thực tế nhưng đủ nhỏ để thực tế mã hóa và giải mã. Trong thực tế, các kích cỡ quan trọng đã được đề xuất để làm cho cuộc tấn công brute-force không thực tế nhưng kết quả trong mã hóa và giải mã tốc độ quá chậm cho mục đích sử dụng chung. Thay vào đó, mã hóa khóa công khai hiện đang bị giới hạn trong việc quản lý khóa và chữ kí ứng dụng.

Một dạng tấn công khác là tìm một số cách để tính toán các khóa riêng dựa vào khóa công khai được cung cấp. Cho đến nay, nó đã được chứng minh bằng toán học bằng hình thức tấn công này là không thể thực hiện được cho một thuật toán khóa công khai cụ thể. Do đó, bất kì thuật toán nào, bao gồm các thuật toán RSA được sử dụng rộng rãi là nghi ngờ.

Lịch sử của phân tích mã cho thấy một vấn đề dường như không thể giải quyết được từ một quan điểm có thể được tìm thấy có một giải pháp nếu nhìn theo một cách hoàn tòa khác.

Cuối cùng, có một dạng tấn công đặc thù đối với các hệ thống khóa công khai. Đây là về bản chất, một cuộc tấn công có thể xảy ra. Giả sử "Một thông báo được gửi đi bao gồm chỉ một khóa DES 56 bit. Một đối tượng có thế mã hóa tất cả các khóa DES 56 bit có thể sử dụng khóa công khai và có thể khám phá khóa mã hóa bởi kết hợp bản mã truyền. Cuộc tấn công này có thể bị cản trở bằng cách thêm một số bit ngẫu nhiên vào các thông báo.

2.2.2. Thuật toán RSA

Các bài báo đầu tiên của Diffie và Hellman đã đưa ra một cách tiếp cận mới đối với mật mã, và thực tế đã thách thức các nhà mật mã học để đưa ra một thuật toán mật mã đáp ứng các yêu cầu đối với các hệ thống khóa công khai. Một số thuật toán đã được đề xuất cho mật mã khóa công khai. Một số trong số này ban đầu đầy hứa hẹn sau đó nó lại có thể bị bẻ gãy.

Một trong những kết quả thành công đầu tiên với thách thức này đã được phát hiện vào năm 1977 của Ron Rivest, Adi Shamir và Len Adleman tại MIT và xuất bản lần đầu vào năm 1978.

Chương trình Rivest-Shamir-Adleman (RSA) được chấp nhận rộng rãi và áp dụng đối với mã hóa khóa công khai từ đó. RSA là một mật mã, trong đó bản mã là số nguyên giữa 0 và n – 1 cho một số n. Một kích thước điển hình cho n là 1024 bit hoặc 309 chữ số thập phân. Đó là, n nhỏ hơn 2^{1024}. Ta kiểm tra RSA trong phần này một cách chi tiết bắt đầu với sự diễn tả về thuật toán. Sau đó, ta kiểm tra một số các tính toán và mật mã của RSA.

2.2.2.1. Mô tả thuật toán

RSA sử dụng một biểu thức với hàm mũ. Bản rõ được mã hóa trong khối, với mỗi khối có một giá trị nhị phân nhỏ hơn một số n. Tức là kích thước khối phải nhỏ hơn hoặc bằng $\log_2(n)+1$; trong thực tế, kích thước khối là i bit, trong đó: $2^i < n \leq 2^{i+1}$. Mã hóa và giải mã có dạng sau đây, đối với một số khối bản rõ M và khối bản mã C

$C = M^e \bmod n$

$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$

Cả người gửi và người nhận phải biết giá trị của n. Người gửi biết giá trị của e và chỉ định người nhận biết giá trị của d. Do đó, đây là thuật toán khóa công khai của PU = {e,n} và khóa cá nhân PR = {d,n}. Đối với thuật toán này là thỏa đáng cho mã hóa khóa công khai, phải đáp ứng các yêu cầu sau:

1. Có thể tìm được các giá trị của *e*, *d* và *n* sao cho $M^{ed} \bmod n = M$ với $\forall M < n$.
2. Tương đối dễ tính toán $M^e \bmod n$ và $C^d \bmod n$ cho tất cả giá trị của M<n.
3. Không thể xác định *d* cho *e* và *n*

Bây giờ, ta xem xét vấn đề đầu tiên: Ta cần phải tìm một mối qua hệ của biểu thức.

$$M^{ed} \bmod n = M$$

Các mối quan hệ trước đó giữa e và d là số nhân nghịch đảo modulo ∅(n), trong đó ∅(n) là hàm mũ của Euler. Nó được thể hiện ở định lí số học rằng p, q là số nguyên tố, ∅(pq) = (p–1)(q–1). Mối quan hệ giữa e và d có thể được biểu diễn bằng: ed mod ∅(n) = 1 điều này tương đương với :

$$ed \equiv 1 \pmod{\emptyset(n)} \Leftrightarrow d \equiv e^{-1} \pmod{\emptyset(n)}$$

Nghĩa là, e và d là phép nhân nghịch đảo mod ∅(n). Lưu ý rằng theo các quy tắc của mô đun số học, điều này là đúng chỉ khi d (và do đó e) tương đối nguyên tố với ∅(n). Tương đương gcd(∅(n),d) = 1.

Bây giờ ta xác định các thành phần thực tế của RSA:

p, q , z số nguyên tố (riêng tư, đã chọn)

n = pq (công khai, tính)

e với gcd (Ø(n),e) = 1; 1 < e < Ø(n) (công khai, đã chọn)

$d \equiv e^{-1} \pmod{Ø(n)}$ (riêng tư, tính)

Khóa cá nhân bao gồm {d, n} và khóa công khai bao gồm {e, n}. Giả sử người dùng A đã công bố khóa công khai của nó và người dùng B muốn gửi tin nhắn M đến A. Sau đó B tính $C = M^e \bmod n$ và truyền C. Khi nhận được bản mã, người sử dụng A giải mã bằng cách tính $M = C^d \bmod n$. Sau đây là tóm tắt thuật toán RSA, tương ứng với hình 2.1. (a). Alice tạo cặp khóa công khai và cá nhân; Bob mã hóa bằng khóa công khai của Alice và Alice giải mã bằng khóa cá nhân.

Alice tạo cặp khóa :

- Chọn p, q với điều kiện p, q cả hai là số nguyên tố; p≠q
- Tính n = p×q
- Tính Ø(n) = (p–1)(q–1)
- Chọn số nguyên e với gcd (Ø(n),e) = 1; 1 < e < Ø(n)
- Tính d với $d \equiv e^{-1} \pmod{Ø(n)}$
- Khóa công khai PU = {e, n}
- Khóa cá nhân PR = {d, n}

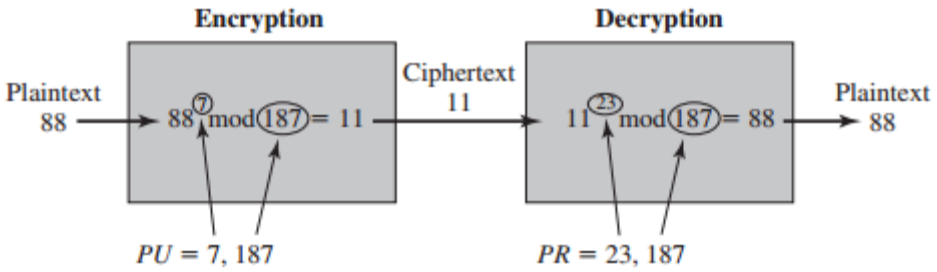
Bob mã hóa tin nhắn với khóa công khai của Alice

- Bản rõ M < n
- Bản mã $C = M^e \bmod n$

Alice giải mã tin nhắn với khóa cá nhân của Alice

- Bản mã C
- Bản rõ $M = C^d \bmod n$

Ví dụ được thể hiện trong hình 2.5. Đối với ví dụ này, các khóa đã được tạo ra như sau:



Hình 2.5 Ví dụ về thuật toán RSA.

Đối với ví dụ này, các khóa được tạo ra như sau:

1. Chọn 2 số nguyên tố, p=17 và q=11
2. Tính n = pq = 17×11=187
3. Tính Ø(n) = (p – 1)×(q – 1) = 16 × 10 = 160

4. Chọn e với điều kiện e tương đối nguyên tố với Ø(n)=160 và nhỏ hơn Ø(n).Ta chọn e =7

5. Xác định d sao cho ed ≡1(mod 160) và d < 160. Giá trị chính xác d = 23, bởi vì 23×4 =161 = (1×160)+1; d có thể được tính bằng cách sử dụng thuật toán Euclid mở rộng.

Kết quả các khóa là khóa công khai PU = {7,187} và khóa cá nhân PR={23,187}. Ví dụ cho thấy việc sử dụng các khóa này cho một đầu vào bản rõ của M = 88. Với mã hóa, chúng ta cần tính $C = 88^7 \bmod 187$. Khai thác tính chất của mô đun số học, chúng ta có thể làm điều này như sau:

$88^7 \bmod 187 = [(88^4 \bmod 187)\times(88^2 \bmod 187)\times(88^1 \bmod 187)] \bmod 187$

$88^1 \bmod 187 = 88$

$88^2 \bmod 187 = 7744 \bmod 187 = 77$

$88^4 \bmod 187 = 59969536 \bmod 187 = 132$

$88^7 \bmod 187 = (88\times77\times132) \bmod 187 = 894432 \bmod 187 = 11$

Đối với giải mã, ta tính $M = 11^{23} \bmod 187$

$11^{23} \bmod 187 = [(11^1 \bmod 187)\times(11^2 \bmod 187)\times(11^4 \bmod 187)\times(11^8 \bmod 187)\times(11^8 \bmod 187)] \bmod 187$

$11^1 \bmod 187 = 11$

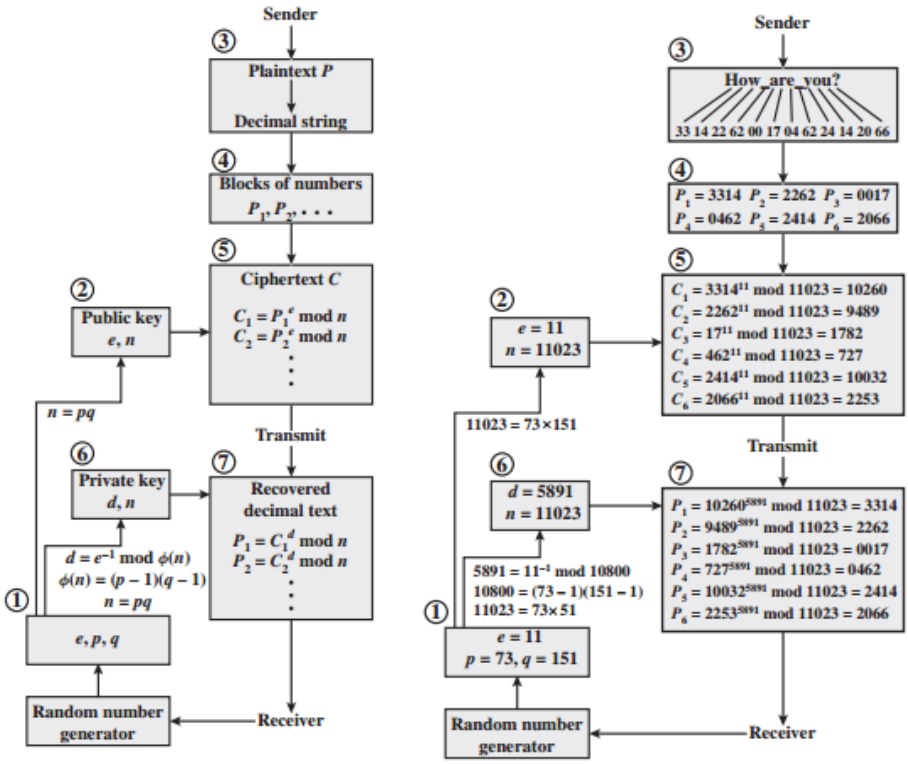
$11^2 \bmod 187 = 121$

$11^4 \bmod 187 = 14641 \bmod 187 = 55$

$11^8 \bmod 187 = 214358881 \bmod 187 = 33$

$11^{23} \bmod 187 = (11\times121\times55\times33\times33) \bmod 187 = 79720245 \bmod 187 = 88$

Bây giờ ta xét 1 ví dụ cho việc sử dụng RSA để xử lý nhiều khối dữ liệu. Trong ví dụ này, bản rõ là một chuỗi chữ số. Mỗi biểu tượng bản rõ được gán một mã duy nhất gồm hai số thập phân (ví dụ: a = 00; A = 26). Một khối bản rõ bao gồm bốn chữ số thập phân, hoặc hai kí tự chữ và số. Hình 2.6. (a) minh họa mã hóa nhiều khối và hình 2.6. (b) đưa ra một ví dụ cụ thể. Số vòng tròn thể hiện thứ tự hoạt động thực hiện.



(a) (b)

Hình 2.6. Xử lý RSA nhiều khối

2.2.2.2. Các khía cạnh tính toán

Bây giờ ta chuyển sang vấn đề cần thiết về tính toán để sử dụng RSA. Thực sự có hai vấn đề để xem xét: mã hóa/giải mã và tạo khóa. Trước tiên ta xét quá trình mã hóa và giải mã.

+ Tính toán lũy thừa trong mô đun số học:

Cả hai mã hóa và giải mã trong RSA liên quan đến việc lũy thừa, một số nguyên cho một số nguyên, mod n. Nếu số lũy thừa được thực hiện trên các số nguyên và sau đó giảm mod n, các giá trị trung gian là vô cùng lớn. Như ví dụ ở trên ta có thể sử dụng một thuộc tính của mô đun số học.

$$[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$$

Như vậy ta có thể giảm các giá trị trung gian theo mod n. Điều này cho phép tính thực tế. Mặt khác xét tính hiệu quả của lũy thừa, đối với RSA, chúng ta đang giải quyết với số mũ lớn. Để xem hiệu suất có thể tăng lên như nào, xét trường hợp chúng ta tính x^{16}. Cách giải quyết bình thường là phải có 15 phép nhân

$$x^{16} = x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x$$

Tuy nhiên ta có thể đạt được kết quả cuối cùng với 4 phép nhân, nếu ta liên tục bình phương của từng kết quả mỗi phần, liên tục hình thành (x^2, x^4, x^8, x^{16}).

Một ví dụ khác, giả sử ta muốn tính $x^{11} \bmod n$, cho một số nguyên x và n:

Ta thấy $x^{11} = x^{1+2+8} = x \times x^2 \times x^8$. Trong trường hợp này, ta tính x mod n, x^2 mod n, x^4 mod n, x^8 mod n và sau đó tính: [(x mod n) ×(x^2 mod n) ×(x^8 mod n)] mod n.

Nhìn chung, giả sử ta muốn tìm giá trị a^b mod n với a, b và n là những số nguyên dương. Nếu ta biểu diễn b như một số nhị phân: $b_k b_{k-1}.....b_0$, sau đó ta có:

$$b=\sum_{b_i \neq 0} 2^i$$

Như vậy:

$$a^b = a^{(\sum_{b_i \neq 0} 2^i)} = \prod_{b_i \neq 0} a^{2^i}$$

$$a^b \text{ mod n} = [\prod_{b_i \neq 0} a^{2^i}] \text{ mod n} = (\prod_{b_i \neq 0} [a^{2^i} \, mod \, n]) \text{ mod n}$$

2.2.2.3. Hoạt động hiệu quả dùng khóa công khai

Để đẩy nhanh hoạt động của thuật toán RSA sử dụng khóa công khai, một lựa chọn đặc biệt của e thường được thực hiện. Lựa chọn phổ biến nhất là 65537 (2^{16}+1). Hai lựa chọn phổ biến khác là 3 và 17. Mỗi lựa chọn này chỉ có hai bít, do đó số lần nhận được yêu cầu để thực hiện lũy thừa được giảm thiểu.

Tuy nhiên, với một khóa công khai rất nhỏ như e = 3, RSA trở lên dễ bị tấn công. Giả sử, chúng ta có ba người dùng RSA khác nhau, tất cả đều dùng giá trị e = 3, nhưng duy nhất giá trị của n. Cụ thể là (n_1, n_2, n_3). Nếu người dùng A gửi một thông điệp mã hóa M đến tất cả ba người, sau đó ba bản mã là :

$C_1 = M^3 \text{ mod } n_1$; $C_2 = M^3 \text{ mod } n_2$; $C_3 = M^3 \text{ mod } n_3$;

Có khả năng n_1; n_2; n_3 là cặp tương đối nguyên tố. Do đó, ta có thể dùng định lý dư lượng của Trung Hoa (CRT) để tính M^3 mod ($n_1 n_2 n_3$). Theo quy tắc của thuật toán RSA, M nhỏ hơn mỗi số của n_i (M < n_1; M < n_2; M < n_3). Như vậy: $M^3 < n_1 n_2 n_3$. Theo đó, kẻ tấn công chỉ cần tính tính toán căn bậc 3 của M^3. Cuộc tấn công này có thể không thực hiện được nếu ta thêm một bít giả cho mỗi trường hợp của M để được mã hóa.

Lưu ý rằng định nghĩa của thuật toán RSA đòi hỏi rằng trong quá trình tạo khóa, người dùng sẽ chọn một giá trị tương đối nguyên tố với Ø(n). Vì vậy, nếu một giá trị của e được chọn đầu tiên và số nguyên tố p và q được tạo ra, nó có thể xảy ra gcd(Ø(n),e) ≠ 1. Trường hợp này ta phải từ chối giá trị p, q và tạo ra một cặp p, q mới.

Chúng ta không thể chọn một giá trị hằng số nhỏ d để hoạt động hiệu quả. Một giá trị nhỏ của dễ bị tấn công bằng cách tấn công brute-force và các hình thức mật mã khác. Tuy nhiên, có một cách để đẩy nhanh tốc độ tính toán sử dụng định lý dư lượng của Trung Hoa. Ta muốn tính giá trị M = C^d mod n. Ta xác định các kết quả trung gian sau đây:

$V_p = C^d \text{ mod p}$; $V_q = C^d \text{ mod q}$

Theo phương trình (1.8): $c_i = M_i \times (M_i^{-1} \bmod m_i)$ với $1 \le i \le k$ của định lý dư lượng Trung Hoa:

$$X_p = q \times (q^{-1} \bmod p); \qquad X_q = p \times (p^{-1} \bmod q)$$

Sau đó dùng phương trình $A \equiv (\sum_{i-1}^{k} a_i c_i) (\text{mod } M)$ biểu diễn:

$$M = (V_p X_p + V_q X_q) \bmod n$$

Hơn nữa, hơn nưa ta có thể đơn giản hóa việc tính V_p và V_q bằng việc sư dụng định lý Fermat, trong đó nêu rằng: $a^{p-1} \equiv 1 \pmod p$; nếu p và a tương đối nguyên tố với nhau. Kết quả tính toán cuối cùng là nhanh hơn gấp bốn lần với $M = C^d \bmod n$.

2.2.2.4. Tạo khóa

Trước khi áp dụng hệ thống mật mã khóa công khai, mỗi người tham gia phải tạo ra một cặp khóa. Điều này bao gồm các vấn đề sau:

- Xác định hai số nguyên tố, p và q.
- Chọn e hoặc d và tính toán khác.

Đầu tiên xem xét việc chọn p và q. Bởi vì giá trị của n = p × q sẽ được biết đến với bất kì kẻ tấn công nào, để ngăn chặn sự phát hiện của p và q bằng các phương pháp đầy đủ, những số nguyên này phải được lựa chọn từ một tập hợp đủ lớn (tức là p và q phải là những con số lớn). Mặt khác, phương pháp này được sử dụng để tìm số nguyên tố lớn phải có hiệu quả hợp lý.

Hiện nay không có các kĩ thuật hữu ích mang lại những con số nguyên tố lớn vì vậy cần phải có một số biện pháp giải quyết vấn đề khác. Phương pháp này nói chung được sử dụng là chọn ngẫu nhiên một số lẻ của vị trí mong muốn của cường độ và kiểm tra số đó là số nguyên tố. Nếu không, chọn số ngẫu nhiên tiếp theo cho đến khi tìm và kiểm tra thấy số nguyên tố.

Nhiều thử nghiệm về tính nguyên tố đã được phát triển và gần như chúng chỉ có tính xác suất. Nghĩa là, việc tử nghiệm sẽ chỉ xác định rằng một số nguyên nhất định có lẽ là số nguyên tố. Mặc cho sự thiếu chắc chắn này, các thử nghiệm này có thể được thực hiện theo cách làm cho xác suất gần bằng 1.0 như mong muốn.

Ví dụ: Một trong những thuật toán phổ biến và hiệu quả là thuật toán Miller-Rabin được giới thiệu ở phần lý thuyết số. Với thuật toán này và hầu hết các thuật toán như vậy, thủ tục để kiểm tra xem một số nguyên n là số nguyên tố là thực hiện một phép tính liên quan đến n và a, chọn ngẫu nhiên số nguyên a. Nếu n không thông qua việc kiểm tra thì n không phải là số nguyên tố. Nếu n thông qua việc kiểm tra, sau đó n có thể là số nguyên tố hoặc không phải là số nguyên tố. Nếu n vượt qua nhiều việc kiểm tra như vậy với nhiều các giá trị được chọn ngẫu nhiên cho a, sau đó ta có thể chắc chắn rằng n là số nguyên tố.

Tóm lại, các bước để chọn một số nguyên tố như sau:

1. Chọn một số nguyên lẻ n ngẫu nhiên (ví dụ: sử dụng bộ tạo số giả ngẫu nhiên).

2. Chọn một số nguyên a < n một cách ngẫu nhiên.

3. Thực hiện phép thử số nguyên tố xác suất, như Miller-Rabin, với a như một tham số. Nếu n không thông qua thử nghiệm, từ chối giá trị n và quay lại bước 1.

4. Nếu n đã vượt qua việc thử nghiệm, chấp nhận n, nếu không quay lại bước 2.

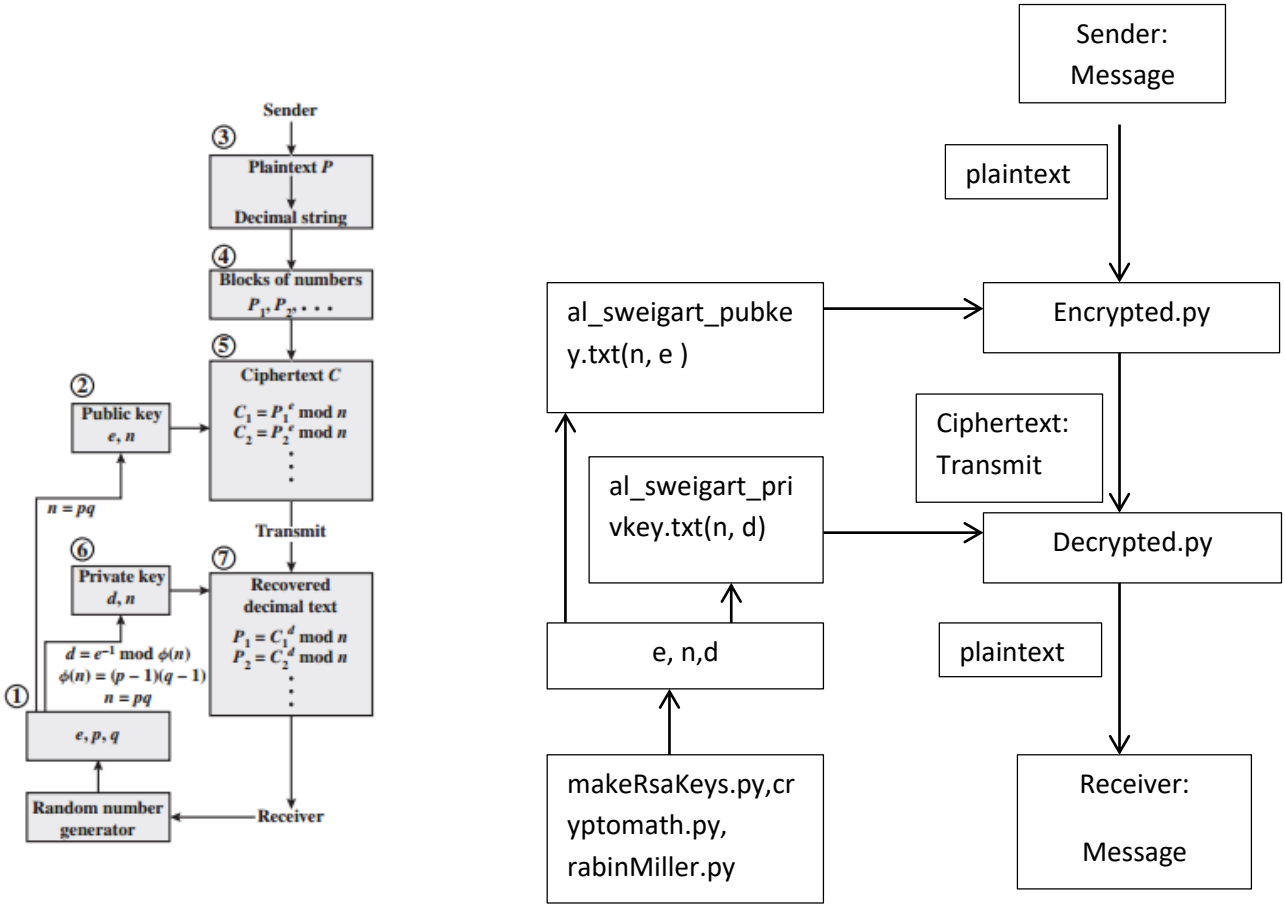
Cần lưu ý rằng có bao nhiêu con số có thể bị từ chối trước khi số nguyên tố được tìm thấy. Kết quả từ lý thuyết số, gọi là định lý số nguyên tố, cho biết các số nguyên tố gần N nằm trên một trong mỗi ln(N) số nguyên. Như vậy, trung bình ta phải kiểm tra theo thứ tự của ln(N) số nguyên trước khi tìm thấy số nguyên tố. Trên thực tế, bởi vì tất cả các số nguyên tố thậm chí có thể ngay lập tức bị từ chối, con số chính xác là ln(N)/2. Ví dụ, nếu một số nguyên trên thứ tự của cường độ 2^{200} đã được tìm kiếm, sau đó $\ln(2^{200})/2 = 70$ thử nghiệm sẽ cần thiết để tìm một số nguyên tố.

Có số nguyên tố p và quá trình tạo ra cặp khóa được hoàn thành bằng cách chọn một giá trị của e và tính d hoặc chọn một giá trí của d và tính e. Giờ ta chọn e sao cho gcd (Ø(n), e) = 1. Sau đó: Tính $d \equiv e^{-1} (\text{mod } (Ø(n))$. Có một thuật toán đơn giản sẽ đồng thời tính toán ước chung lớn nhất của hai số nguyên và nếu gcd bằng 1, xác định nghịch đảo của 1 trong số các số nguyên khác.Thuật toán này được gọi là Euclid mở rộng. Do đó các bước tạo ra một chuỗi số ngẫu nhiên, kiểm tra mỗi số đối với Ø(n) cho đến khi một số nguyên tương đối nguyên tố với Ø(n) được tìm thấy.

Chương 3.

Chương trình mã hóa và giải mã RSA

3.1. Mô tả thuật toán



3.2. Mã chương trình

3.2.1. Cryptomath.py

Sử dụng thuật toán Euclid mở rộng để tính ước chung lớn nhất nhất nếu ước chung lớn nhất bằng 1, xác định nghịch đảo.

```
def gcd(a,b):
    while a!=0:
        a,b =b%a,a
    return b
def findModInverse(a, m):
    if gcd(a, m) != 1:
            return None
```

```
    u1, u2, u3 = 1, 0, a
    v1, v2, v3 = 0, 1, m
    while v3 != 0:
        q = u3 // v3
        v1, v2, v3, u1, u2, u3 = (u1 - q * v1), (u2 - q * v2), (u3 - q * v3),v1, v2, v3
    return u1 % m
```

3.2.2. RabinMiller.py

Sử dụng thuật toán Rabin-Miller để kiểm tra số nguyên tố

```
#Primality Testing with the Rabin-Miller Algorithm
import random

def rabinMiller(num):
    #return True if num is prime number.
    s = num - 1
    t = 0
    while s % 2 ==0:
        s = s // 2
        t += 1

    for trials in range(5):
        a = random.randrange(2, num - 1)
        v = pow(a, s , num)
        if v != 1:
            i = 0
            while v != (num - 1 ):
                if i == t - 1 :
                    return False
                else:
                    i = i + 1
                    v =(v**2)%num
    return True

def isPrime(num):
    if(num<2):
        return False

    lowPrimes = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47,
53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137,
```

```
139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227,
229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313,
317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419,
421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509,
521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617,
619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727,
733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829,
839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947,
953, 967, 971, 977, 983, 991, 997]
    if num in lowPrimes:
        return True
    for prime in lowPrimes:
        if (num % prime == 0 ):
            return False

    return rabinMiller(num)

def generateLargePrime(keysize=1024):
    while True:
        num = random.randrange(2**(keysize-1),2**(keysize))
        if isPrime(num):
            return num
```

3.2.3. MakeRsaKeys.py

Tạo khóa

```
#RSA Key generator
import random, sys, os, rabinMiller, cryptomath


def main():
    print('Making key file...')
    makeKeyFiles('al_sweeigart', 1024)
    print('Key files made.')


def generateKey(keySize):
    #Step 1
    print('Generating p prime...')
```

```
    p = rabinMiller.generateLargePrime(keySize)
    print('Generating q prime...')
    q = rabinMiller.generateLargePrime(keySize)
    n = p*q

    #Step 2
    print('Generating e that is relatively prime to (p-1)*(q-1)...')
    while True:
        e=random.randrange(2**(keySize - 1), 2**(keySize))
        if cryptomath.gcd(e, (p-1)*(q-1))==1:
            break

    #Step 3
    print('Calculating d that is mod inverse of e...')
    d = cryptomath.findModInverse(e, (p-1)*(q-1))
    publicKey = (n, e)
    privateKey = (n, d)

    print('Public key : ', publicKey)
    print('Private key:' , privateKey)

    return (publicKey, privateKey)

def makeKeyFiles(name, keySize):
    if os.path.exists('%s_pubkey.txt' %(name)) or os.path.exists('%s_privkey.txt'%(name)):
        sys.exit('WARNING: The file %s_pubkey.txt or %s_privkey.txt already exists! Use a different name or delete these files and re-run this program.' %(name, name))
    publicKey, privateKey = generateKey(keySize)

    print()
    print('The public key is a %s and a %s digit number.' %(len(str(publicKey[0])), len(str(publicKey[1]))))
    print('Writing public key to file %s_pubkey.txt ...'%(name))
    fo = open('%s_pubkey.txt'%(name), 'w')
    fo.write('%s,%s,%s' %(keySize, publicKey[0], publicKey[1]))
    fo.close()
```

```
    print()
    print('The private key is a %s and a %s digit number.' %(len(str(publicKey[0])),
len(str(publicKey[1]))))
    print('Writing private key to file %s_privkey.txt...' %(name))
    fo = open('%s_privkey.txt' %(name), 'w')
    fo.write('%s,%s,%s' %(keySize, privateKey[0], privateKey[1]))
    fo.close()

if __name__=='__main__':
    main()
```

3.2.4. Encrypted.py

```
#RSA Cipher encrypt


import sys


blockSize = 128 # 128 byte

BYTE_SIZE = 256 # 1 byte have 256 value


filename = 'encrypted_file.txt'

message = input('Input message:')

print ('Message : ',message)

print
('**********************************************************************
*')

pubKeyFilename = 'al_sweigart_pubkey.txt'

print ('Encrypting and writing to %s.....'%(filename))

#encryptedText = encryptAndWriteToFile(filename, pubKeyFilename, message)

#def encryptAndWriteToFile(messageFilename, keyFilename, message,
blockSize= DEFAULT_BLOCK_SIZE)
```

```
#keySize, n , e = readKeyFile(keyFilename)

fo = open(pubKeyFilename)

content = fo.read()

fo.close()

keySize , n , e = content.split(',')

keySize=int(keySize)

n=int(n)

print ('number n is :%s' %n)

print
('**********************************************************************')

e=int(e)

print ('number e is : %s' %e)

print
('**********************************************************************')


if keySize < blockSize*8:# *8 to convert bytes to bits

    sys.exit('ERROR: block size is %s bits and key size is %s bits. The RSA cipher
requires the block size to be equal to or less than the key size. Either increase the
block size or use different keys.' % (blockSize * 8,keySize))

#encryptedBlocks = encryptMessage(message,(n,e),blockSize)

#def encryptMessage(message, key, blockSize=DEFAULT_BLOCK_SIZE):

encryptedBlocks = []

#for block in getBlocksFromText(message, blockSize):

messageBytes = message.encode('ascii')

blockInts = []

for blockStart in range(0, len(messageBytes), blockSize):

    blockInt = 0
```

```
    for i in range(blockStart, min(blockStart + blockSize, len(messageBytes))):
        blockInt += messageBytes[i]* (BYTE_SIZE**(i%blockSize))
    blockInts.append(blockInt)
print('blockInts is: ',blockInts)
print('************************************************************')
encryptedBlocks = []
for block in blockInts:
    #ciphettext = plaintext ^ e mod n
    encryptedBlocks.append(pow(block, e, n))
print ('Encryptedblocks : ',encryptedBlocks)
print('************************************************************')

for i in range(len(encryptedBlocks)):
    encryptedBlocks[i]=str(encryptedBlocks[i])
encryptedContent=','.join(encryptedBlocks)
encryptedContent = '%s_%s_%s'%(len(message), blockSize, encryptedContent)
fo = open(filename,'w')
fo.write(encryptedContent)
fo.close()
print ('Len(message)__ blockSize__content is: ',encryptedContent)
print
('************************************************************')
```

3.2.5. Decrypted.py

```
#RSA cipher decrypt

import sys

blockSize = 128 # 128 byte
```

```
BYTE_SIZE = 256 # 1 byte have 256 value

filename = 'encrypted_file.txt'
privKeyFilename = 'al_sweigart_privkey.txt'
print ('Reading from %s and decrypting ...'%(filename))
#decryptedText = readFromFileAndDecrypt(filename, privKeyFilename)
#def readFromFileAndDecrypt(messageFilename, keyFilename):
#keySize , n , d = readKeyFilename(keyFilename)
#def readKeyFilename(keyFilename):
fo = open(privKeyFilename)
content = fo.read()
fo.close()
keySize, n , d = content.split(',')
keySize = int(keySize)
print ('keySize is : ', keySize)
print('*******************************')
n = int(n)
print ('number n is : ',n)
print('*******************************')
d =int(d)
print ('number d is :', d)
print ('*******************************')
fo =open(filename)
content = fo.read()
messageLength, blockSize, encryptedMessage = content.split('_')
messageLength = int(messageLength)
print('mesageLength is :' ,messageLength)
print
('************************************************************')
blockSize = int(blockSize)
print ('blockSize is : ',blockSize)
print('************************************************************
**')
print('encryptedMessage is :',encryptedMessage)
print('************************************************************
***')
if keySize < blockSize*8: #*8 to convert byte to bits
```

```
        sys.exit('Error :The RSA cipher requires the block size to be equal to or less
    than the key size. Did you specify the correct key file and encrypted file?')
    encryptedBlocks = []
    for block in encryptedMessage.split(','):
        encryptedBlocks.append(int(block))
    print('encryptedBlocks is :',encryptedBlocks)
    #return decryptMessage(encryptedBlocks, messageLength,(n,d), blockSize)
    decryptedBlocks = []
    for block in encryptedBlocks:
        #plaintext = ciphertext ^ d mod n
        decryptedBlocks.append(pow(block,d,n))
    print('decryptedBlocks is:',decryptedBlocks)
    print('**********************************************************
    ******8')
    #   return getTextFromBlocks(decryptedBlocks, messageLength, blockSize)
    #def getTextFromBlocks (blockInts, messageLength, blockSize)
    message = []
    for blockInt in decryptedBlocks:
        blockMessage = []
        for i in range(blockSize - 1 ,-1, -1):
            if len(message) + 1 < messageLength:
                asciiNumber = blockInt //(BYTE_SIZE **i)
                blockInt = blockInt%(BYTE_SIZE**i)
                blockMessage.insert(0, chr(asciiNumber))
        print('blockMesage', blockMessage)
        print(''.join(blockMessage))
        message.append(blockMessage)
print('message.extend',message)
```

KẾT LUẬN

Trên đây em đã trình bày toàn bộ nội dung về xây dựng chương trình mã hóa và giải mã mật mã khóa công khai RSA. Để tìm hiểu về cách hoạt động của mật mã khóa công khai, em đã tìm hiểu về cơ sở toán học, các thuật toán để xây dựng về mã hóa RSA. Mã hóa RSA hoạt động dựa trên cơ sở toán học như các định lý Fermat, định lý Euler, định lý còn lại của Trung Hoa và các các thuật toán Euler, thuật toán Miller-Rabin. Chương trình mã hóa và giải mã được xây dựng trên ngôn ngữ Python3.

Do điệu kiện thời gian có hạn cùng với khả năng còn hạn chế nên không tránh khỏi thiếu sót. Vậy mong được thầy cô chỉ bảo để quyển đồ án được hoàn thiện hơn.

Em xin chân thành cám ơn thầy Nguyễn Văn Dương, thầy giáo hướng dẫn trực tiếp, và các thầy cô khác trong bộ môn tận tình giúp đỡ và tạo điều kiện để em hoàn thành quyển đồ án này.

TÀI LIỆU THAM KHẢO

1. Cryptography and Network Security – Principles and Practice - Sixth Edittion (William Stallings).

2. Volume 1/Fundamental Algorithms – The Art Computer Programming –Third Edition.

3. Core Python Programming – Wesley J.Chun

4. Applied Cryptography and Network Securty – Ioana Boureanu Philippe Owesarski Serge Vaudenay (Eds.)- 12th International Conference, ACNS 2014 Lausanne, Switzerland, June 10-13, 2014 Proceeedings.

5. Topics in Cryptology-CT-RSA 2014-Josh Benaloh (Ed.) – The Cryptographer's Track at the RSA conference 2014 San Francisco, CA, USA, Frebruary 25-28, 2014 Proceeedings