MỤC LỤC

GIỚI THIỆU

Con người luôn có nhu cầu trao đổi thông tin với nhau. Nhu cầu đó tăng cao khi các công nghệ mới ra đời đáp ứng cho việc trao đổi thông tin ngày càng nhanh. Chúng ta vẫn không quên việc chiếc máy điện thoại ra đời đã là bước tiến vượt bậc trong việc rút ngắn khoảng cách đáng kể cả về thời gian và không gian giữa hai bên muốn trao đổi thông tin. Những bức thư hay điện tín được gửi đi nhanh hơn khi các phương tiện truyền thông phát triển. Đặc biệt hơn là từ khi Internet xuất hiện, dường như yêu cầu trao đổi thông tin của chúng ta được đáp ứng ngay khi ấn phím "send". Sẽ còn rất nhiều tiện ích mà các công nghệ mới đã đem lại cho chúng ta trong mọi lĩnh vực Kinh tế-Văn hóa-Giáo dục-Y tế...

Ích lợi của Internet mang lại đối với xã hội là vô cùng, nhưng cũng không thể không kể đến những mặt trái của nó khi con người sử dụng nó với mục đích không tốt. Vì vậy mà đối với những thông tin quan trọng khi truyền trên mạng như những bản hợp đồng ký kết, các văn kiện mang tính bảo mật... thì vấn đề quan tâm nhất đó là có truyền được an toàn hay không?

Do vậy để chống lại sự tấn công hay giả mạo, thì nảy sinh yêu cầu là cần phải làm thế nào cho văn bản khi được gửi đi sẽ "không được nhìn thấy", hoặc không thể giả mạo văn bản, dù có xâm nhập được vào văn bản. Nhu cầu đó ngày nay đã được đáp ứng khi công nghệ mã hóa và chữ ký số ra đời. Với công nghệ này, thì đã trợ giúp con người giải quyết được bài toán nan giải về bảo mật khi trao đổi thông tin.

Cùng với sự phát triển của mật mã khóa công khai, người ta đã nghiên cứu và đưa ra nhiều phương pháp, nhiều kỹ thuật ký bằng chữ ký số ứng dụng trong các hoạt động kinh tế, xã hội. Chẳng hạn như các ứng dụng trong thương mại điện tử, các giao dịch của các chủ tài khoản trong ngân hàng, các ứng dụng trong chính phủ điện tử đòi hỏi việc xác nhận danh tính phải được đảm bảo.

Ngày nay chữ ký số được sử dụng trong nhiều lĩnh vực như trong kinh tế với việc trao đổi các hợp đồng giữa các đối tác kinh doanh, trong xã hội là các cuộc bỏ phiếu kín khi tiến hành bầu cử từ xa, hay trong các cuộc thi phạm vi rộng lớn.

Một số chữ ký đã được xây dựng là: chữ ký RSA, chữ ký ELGAMAL, chữ ký DSS, chữ ký RABIN... Mặc dù các chữ ký số còn nhiều hạn chế như là về kích thước chữ ký, hay khả năng chống giả mạo chưa cao... nhưng những khả năng mà nó đem lại là rất hữu ích.

RSA (Rivest-Shamir-Adleman): năm 1977, R.1. Rivest, A. Shamir và L.M. Adleman đề xuất một hệ mật mã khóa công khai mà độ an toàn của hệ dựa vào bài toán khó "phân tích số nguyên thành thừa số nguyên tố", hệ này trở thành một hệ nổi tiếng và mang tên là hệ RSA.

ELGAMAL: hệ mật mã ElGamal được T. ElGamal đề xuất năm 1985, độ an toàn của hệ dựa vào độ phức tạp của bài toán tính logarit rời rạc.

DSS (Digital Signature Standard) được đề xuất từ năm 1991 và được chấp nhận vào cuối năm 1994 để sử dụng trong một số lĩnh vực giao dịch điện tử tại Hoa Kỳ. DSS dựa vào sơ đồ chữ ký ElGamal với một vài sửa đổi.

RABIN: hệ mã hóa khóa công khai được M.O. Rabin đề xuất năm 1977, độ an toàn của hệ dựa vào bài toán khó "phân tích số nguyên thành thừa số nguyên tố".

Khi nói đến chữ ký điện tử, chúng ta luôn lấy mục tiêu an toàn lên hàng đầu. Một chữ ký điện tử chỉ thực sự được áp dụng trong thực tế nếu như nó được chứng minh là không thể giả mạo. Mục tiêu lớn nhất của kẻ tấn công các sơ đồ chữ ký chính là giả mạo chữ ký, điều này có nghĩa kẻ tấn công sẽ sinh ra được chữ ký của người ký lên thông điệp, mà chữ ký này sẽ được chấp nhận bởi người xác nhận. Trong thực tế các hành vi tấn công chữ ký điện tử là hết sức đa dạng. Đó cũng là vấn đề chính được nghiên cứu trong luận văn "Nghiên cứu một số loại tấn công chữ ký số". Nội dung chính của luận văn này bao gồm 2 chương:

Chương 1: Một số khái niệm cơ bản .

Chương 2: Tấn công chữ ký số.

Chương 1. MỘT SỐ KHÁI NIỆM CƠ BẢN

1.1. CÁC KHÁI NIỆM TRONG TOÁN HỌC

1.1.1. Một số khái niệm trong số học

1.1.1.1. Số nguyên tố

1/. Khái niệm

Số nguyên tố là số tự nhiên lớn hơn 1 và chỉ có hai ước là 1 và chính nó.

2/. Ví dụ:

Các số 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37 là số nguyên tố.

Số 2 là số nguyên tố ***chẵn*** duy nhất.

Số nguyên tố có vai trò và ý nghĩa to lớn trong số học và lý thuyết mật mã. Bài toán kiểm tra tính nguyên tố của một số nguyên dương n và phân tích một số n ra thừa số nguyên tố là các bài toán rất được quan tâm.

Ví dụ: ***10 số nguyên tố lớn đã được tìm thấy*** [33]

rank	Prime	Digits	Who	when	reference
1	$2^{32582657} - 1$	9808358	G9	2006	Mersenne 44??
2	$2^{30402457} - 1$	9152052	G9	2005	Mersenne 43??
3	$2^{25964951} - 1$	7816230	G8	2005	Mersenne 42??
4	$2^{24036583} - 1$	7235733	G7	2004	Mersenne 41??
5	$2^{20996011} - 1$	6320430	G6	2003	Mersenne 40??
6	$2^{13466917} - 1$	4053946	G5	2001	Mersenne 39??
7	$19249 \cdot 2^{13018586} + 1$	3918900	SB10	2007	
8	$27653 \cdot 2^{9167433} + 1$	2759677	SB8	2005	
9	$28433 \cdot 2^{7830457} + 1$	2357207	SB7	2004	
10	$33661 \cdot 2^{7031232} + 1$	2116617	SB11	2007	

1.1.1.2. Ước số và bội số

1/. Khái niệm

Cho hai số nguyên a và b, b ≠ 0. Nếu có một số nguyên q sao cho a = b*q, thì ta nói rằng a ***chia hết*** cho b, kí hiệu **b\a.** Ta nói b là ***ước*** của a, và a là ***bội*** của b.

2/. Ví dụ:

Cho a = 6, b = 2, ta có 6 = 2*3, ký hiệu 2\6. Ở đây 2 là ước của 6 và 6 là bội của 2.

Cho các số nguyên a, b ≠ 0, tồn tại cặp số nguyên (q, r) (0 ≤ r < /b/) duy nhất sao cho **a = b*q + r**. Khi đó q gọi là ***thương nguyên***, r gọi là số ***dư*** của phép chia a cho b. Nếu r = 0 thì ta có phép chia hết.

Ví dụ:

Cho a = 13, b = 5, ta có 13 = 5*2 + 3. Ở đây thương là q = 2, số dư là r = 3.

1.1.1.3. Ước số chung và bội số chung

1/. Khái niệm

Số nguyên **d** được gọi là ***ước chung*** của các số nguyên $a_1, a_2, ..., a_n$, nếu nó là ***ước*** của tất cả các số đó.

Số nguyên m được gọi là ***bội chung*** của các số nguyên $a_1, a_2, ..., a_n$, nếu nó là ***bội*** của tất cả các số đó.

Một ước chung d > 0 của các số nguyên $a_1, a_2, ..., a_n$, trong đó mọi ước chung của $a_1, a_2, ..., a_n$ đều là ước của d, thì d được gọi là ***ước chung lớn nhất*** (UCLN) của $a_1, a_2, ..., a_n$. Ký hiệu d = gcd ($a_1, a_2, ..., a_n$) hay d = **UCLN**($a_1, a_2, ..., a_n$).

Một bội chung **m** > 0 của các số nguyên $a_1, a_2, ..., a_n$, trong đó mọi bội chung của $a_1, a_2, ..., a_n$ đều là bội của **m**, thì **m** được gọi là ***bội chung nhỏ nhất*** (BCNN) của $a_1, a_2, ..., a_n$. Ký hiệu **m = lcm**($a_1, a_2, ..., a_n$) hay **m = BCNN**($a_1, a_2, ..., a_n$).

2/. Ví dụ:

Cho a = 12, b = 15, **gcd**(12, 15) = 3, **lcm**(12, 15) = 60.

1.1.1.4. Số nguyên tố cùng nhau

1/. Khái niệm

Nếu **gcd**($a_1, a_2, ..., a_n$) = 1, thì các số $a_1, a_2, ..., a_n$ gọi là ***nguyên tố cùng nhau.***

2/. Ví dụ:

Hai số 8 và 13 là ***nguyên tố cùng nhau***, vì **gcd**(8, 13) = 1.

1.1.1.5. Khái niệm Đồng dư

1/. Khái niệm

Cho hai số nguyên a, b, m (m > 0). Ta nói rằng a và b ***"đồng dư"*** với nhau theo **modulo m**, nếu chia a và b cho **m**, ta nhận được cùng một số dư.

Ký hiệu: **a ≡ b** (mod m).

2/. Ví dụ:

17 ≡ 5 (mod 3) vì chia 17 và 5 cho 3, được cùng số dư là 2.

1.1.2. Một số khái niệm trong đại số

1.1.2.1. Nhóm

1/. Khái niệm

Nhóm là một bội (G, *), trong đó G ≠ ∅, * là ***phép toán hai ngôi*** trên G thỏa mãn ba tính chất sau:

+ Phép toán có tính kết hợp: (x*y)*z = x*(y*z) với mọi x, y, z ∈ G.

+ Có phần tử ***trung lập*** e ∈ G: x*e = e*x = x với mọi x ∈ G.

+ Với mọi x ∈ G, có phần tử nghịch đảo x' ∈ G: x*x' = x'*x = e.

Cấp của nhóm G được hiểu là số phần tử của nhóm, ký hiệu là |G|.

Cấp của nhóm có thể là ∞ nếu G có vô hạn phần tử.

Nhóm Abel là nhóm (G, *), trong đó phép toán hai ngôi * có tính giao hoán.

Tính chất: Nếu a*b = a*c, thì b = c.

Nếu a*c = b*c, thì a = b.

2/. Ví dụ:

* Tập hợp các số nguyên Z cùng với phép cộng (+) thông thường là nhóm giao hoán, có phần tử đơn vị là số 0. Gọi là ***nhóm cộng*** các số nguyên.

* Tập Q^* các số hữu tỷ khác 0 (hay tập R^* các số thực khác 0), cùng với phép nhân (*) thông thường là nhóm giao hoán. Gọi là ***nhóm nhân*** các số hữu tỷ (số thực) khác 0.

* Tập các vectơ trong không gian với phép toán cộng vectơ là nhóm giao hoán.

*1.1.2.2. Nhóm con của nhóm (G, *)*

1/. Khái niệm

Nhóm con của G là tập $S \subset G$, $S \neq \phi$, và thỏa mãn các tính chất sau:

+ Phần tử trung lập e của G nằm trong S.

+ S khép kín đối với phép tính (*) trong G, tức là $x*y \in S$ với mọi $x, y \in S$.

+ S khép kín đối với phép lấy nghịch đảo trong G, tức $x^{-1} \in S$ với mọi $x \in S$.

1.1.2.3. Nhóm Cyclic

1/. Khái niệm

Nhóm (**G**, *) được gọi là ***Nhóm Cyclic*** nếu nó được sinh ra bởi một trong các phần tử của nó.

Tức là có phần tử $\mathbf{g} \in \mathbf{G}$ mà với mỗi $\mathbf{a} \in \mathbf{G}$, đều tồn tại $\mathbf{n} \in \mathbf{N}$ để

g^n **=g*g*...*g = a.** (Chú ý **g*g*...*g** là **g*g** với **n** lần).

Nói cách khác: **G** được gọi là Nhóm Cyclic nếu tồn tại $\mathbf{g} \in \mathbf{G}$ sao cho mọi phần tử trong **G** đều là một ***lũy thừa nguyên*** nào đó của g.

2/. Ví dụ:

Nhóm ($\mathbf{Z}^+$, +) gồm các số nguyên dương là Cyclic với phần tử sinh **g** = 1.

1.1.2.4. Tập thặng dư thu gọn theo modulo

1/. Khái niệm

Kí hiệu $\mathbf{Z}_n$ = {0, 1, 2, ..., n-1} là tập các số nguyên không âm < n.

$\mathbf{Z}_n$ và phép cộng (+) lập thành ***nhóm Cyclic*** có phần tử sinh là **1**, phần tử trung lập **e** = 0. ($\mathbf{Z}_n$, +) gọi là nhóm cộng, đó là nhóm hữu hạn có cấp n.

Kí hiệu $\mathbf{Z}_n^*$ = {**x** ∈ $\mathbf{Z}_n$, **x** là nguyên tố cùng nhau với **n**}. Tức là **x** phải ≠ 0.

$\mathbf{Z}_n^*$ được gọi là ***Tập thặng dư thu gọn theo mod n***, có số phần tử là **ϕ(n)**.

$\mathbf{Z}_n^*$ với phép nhân mod n lập thành một nhóm (nhóm nhân), phần tử trung lập e = 1.

Tổng quát ($\mathbf{Z}_n^*$, phép nhân mod n) không phải là nhóm Cyclic.

Nhóm nhân $\mathbf{Z}_n^*$ là Cyclic chỉ khi **n** có dạng: 2, 4, p^k hay $2p^k$ với p là nguyên tố lẻ.

2/. Ví dụ: Cho n = 21, Z_n^* = {1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20}.

1.1.2.5. Phần tử nghịch đảo đối với phép nhân

1/. Khái niệm

Cho **a** ∈ $\mathbf{Z}_n$, nếu tồn tại **b** ∈ $\mathbf{Z}_n$ sao cho **a b** ≡ 1 (mod n), ta nói **b** là ***phần tử nghịch đảo*** của **a** trong $\mathbf{Z}_n$ và ký hiệu $\mathbf{a}^{-1}$.

Một phần tử có phần tử nghịch đảo, gọi là khả nghịch.

2/. Ví dụ: Tìm phần tử nghịch đảo của 3 trong $\mathbf{Z}_7$

Tức là phải giải phương trình 3 **x** ≡ 1 (mod **7**), x sẽ là phần tử nghịch đảo của 3.

I	g_i	u_i	v_i	y
1	7	1	0	
1	3	0	1	2
2	1	1	-2	3
3	0			

Vì t = V_2 = -2 < 0 do đó x = a^{-1} := 1 + n = -2 + 7 = 5.

Vậy 5 là phần tử nghịch đảo của 3 trong Z_7.

1.1.3. Độ phức tạp của thuật toán

1.1.3.1. Khái niệm bài toán

Bài toán được diễn đạt bằng hai phần:

Input: Các dữ liệu vào của bài toán.

Ouput: Các dữ liệu ra của bài toán (kết quả).

Không mất tính chất tổng quát, giả thiết các dữ liệu trong bài toán đều là số nguyên.

1.1.3.2. Khái niệm Thuật toán

"Thuật toán" được hiểu đơn giản là cách thức để giải một bài toán. Cũng có thể được hiểu bằng hai quan niệm: Trực giác hay Hình thức như sau:

1/. Quan niệm trực giác về "Thuật toán".

Một cách trực giác, Thuật toán được hiểu là một dãy hữu hạn các qui tắc (chỉ thị, mệnh lệnh) mô tả một quá trình tính toán, để từ dữ liệu đã cho (Input) ta nhận được kết quả (Output) của bài toán.

2/. Quan niệm toán học về "Thuật toán".

Một cách hình thức, người ta quan niệm thuật toán là một máy Turing.

Thuật toán được chia thành hai loại: Đơn định và không đơn định.

Thuật toán đơn định (Deterministic):

Là thuật toán mà kết quả của mọi phép toán đều được xác định duy nhất.

Thuật toán không đơn định (NoDeterministic):

Là thuật toán có ít nhất một phép toán mà kết quả của nó là không duy nhất.

1.1.3.3. Khái niệm Độ phức tạp của thuật toán

1/. Chi phí của thuật toán (Tính theo một bộ dữ liệu vào):

Chi phí phải trả cho quá trình tính toán gồm chi phí về thời gian và bộ nhớ.

Chi phí thời gian của một quá trình tính toán là thời gian cần thiết để thực hiện một quá trình tính toán. Với thuật toán tựa Algol: Chi phí thời gian là số các phép tính cơ bản thực hiện trong quá trình tính toán.

Chi phí bộ nhớ của một quá trình tính toán là số ô nhớ cần thiết để thực hiện một quá trình tính toán. Gọi A là thuật toán, e là dữ liệu vào của bài toán đã được mã hóa bằng cách nào đó. Thuật toán A tính trên dữ liệu vào e phải trả một giá nhất định. Ta ký hiệu: $\mathbf{t}_A(e)$ là giá thời gian và $\mathbf{I}_A$ **(e)** là giá bộ nhớ.

2/. Độ phức tạp về bộ nhớ (Trong trường hợp xấu nhất):

$L_A(n) = \max\{ I_A(e)$, với $|e| \le n\}$, n là "kích thước" đầu vào của thuật toán.

3/. Độ phức tạp thời gian (Trong trường hợp xấu nhất):

$T_A(n) = \max\{ t_A(e)$, với $|e| \le n\}$.

4/. Độ phức tạp tiệm cận: Độ phức tạp PT(n) được gọi là tiệm cận tới hàm $f(n)$ ký hiệu O(f(n)) nếu $\exists$ các số n_0, c mà $PT(n) \le c.f(n)$, $\forall n \ge n_0$.

5/. Độ phức tạp đa thức:

Độ phức tạp PT(n) được gọi ***đa thức***, nếu nó ***tiệm cận tới đa thức p(n).***

6/. Thuật toán đa thức: Thuật toán được gọi là ***đa thức***, nếu độ phức tạp về thời gian (trong trường hợp xấu nhất) của nó là ***đa thức***.

Nói cách khác:

+ Thuật toán ***thời gian đa thức*** là thuật toán có độ phức tạp thời gian $\mathbf{O(n^t)}$, trong đó t là hằng số.

+ Thuật toán ***thời gian hàm mũ*** là thuật toán có độ phức tạp thời gian $\mathbf{O(t^{f(n)})}$, trong đó t là hằng số và f(n) là đa thức của n.

**** Thời gian chạy của các lớp thuật toán khác nhau:***

Độ phức tạp	Số phép tính ($n = 10^6$)	Thời gian (10^6 phép tính/s)
O(1)	1	1 micro giây
O(n)	10^6	1 giây
$O(n^2)$	10^{12}	11,6 ngày
$O(n^3)$	10^{18}	32 000 năm
$O(2^n)$	10^{301030}	10^{301006} tuổi của vũ trụ

Chú ý

- Có người cho rằng ngày nay máy tính với tốc độ rất lớn, không cần quan tâm nhiều tới thuật toán nhanh, chúng tôi xin dẫn một ví dụ đã được kiểm chứng.
- Bài toán xử lý n đối tượng, có ba thuật toán với 3 mức phức tạp khác nhau sẽ chịu 3 hậu quả như sau: ***Sau 1 giờ:***

Thuật toán A có độ phức tạp O(n) : xử lý được 3,6 triệu đối tượng.

Thuật toán B có độ phức tạp O(n log n) : xử lý được 0,2 triệu đối tượng.

Thuật toán C có độ phức tạp $O(2^n)$: xử lý được 21 đối tượng.

1.1.3.4. Khái niệm "dẫn về được"

Bài toán **B** được gọi là ***"Dẫn về được"*** bài toán **A** một cách ***đa thức*** ,

ký hiệu: **B** $\propto$ **A**, nếu có thuật toán đơn định đa thức để giải bài toán **A**, thì cũng có thuật toán đơn định để giải bài toán **B.**

Nghĩa là: Bài toán **A** "khó hơn" bài toán **B**, hay **B** "dễ" hơn **A, B** được diễn đạt bằng ngôn ngữ của bài toán **A**, hay có thể hiểu **B** là trường hợp riêng của **A.**

Vậy nếu giải được bài toán **A** thì cũng sẽ giải được bài toán **B.**

Quan hệ $\propto$ có tính chất bắc cầu: Nếu C $\propto$ B và B $\propto$ A thì C $\propto$ A.

1.1.3.5. Khái niệm "khó tương đương"

Bài toán A gọi là "khó tương đương" bài toán B, ký hiệu **A ~ B,**

nếu: A $\propto$ B và B $\propto$ A

1.1.3.6. Lớp bài toán P, NP

Ký hiệu:

P là lớp bài toán giải được bằng thuật toán đơn định, đa thức (Polynomial).

NP là lớp bài toán giải được bằng thuật toán không đơn định, đa thức.

Theo định nghĩa ta có P $\subset$ NP.

Hiện nay người ta chưa biết được **P** $\neq$ **NP ?**

1.1.3.7. Lớp bài toán NP – Hard

Bài toán **A** được gọi là ***NP - Hard*** (NP - khó) nếu $\forall$**L** $\in$ **NP** đều là **L** $\propto$ **A.**

Lớp bài toán NP - Hard bao gồm tất cả những bài toán NP - Hard.

Bài toán NP - Hard có thể nằm ***trong*** hoặc ***ngoài*** lớp NP

1.1.3.8. Lớp bài toán NP – Complete

Bài toán A được gọi là NP - Complete (NP-đầy đủ) nếu A là ***NP - Hard*** và A $\in$ **NP.**

Bài toán NP - Complete là bài toán NP - Hard nằm trong lớp NP.

Lớp bài toán NP - Complete bao gồm tất cả những bài toán NP - Complete .

Lớp NP – Complete là có thực, vì Cook và Karp đã chỉ ra BT đầu tiên thuộc lớp này, đó là bài toán "thỏa được": SATISFYABILITY.

1.1.3.9. Hàm một phía và hàm cửa sập một phía

1/. Hàm f(x) được gọi là ***hàm một phía*** nếu tính ***"xuôi"*** y = f(x) thì ***"dễ"***, nhưng tính ***"ngược"*** x = $f^{-1}(y)$ lại rất ***"khó".***

Ví dụ:

Hàm f(x) = g^x (mod p), với p là số nguyên tố lớn, (g là phần tử nguyên thủy mod p) là hàm một phía.

2/. Hàm f(x) được gọi là ***hàm cửa sập một phía*** nếu tính y = f(x) thì ***"dễ"***, tính x = $f^{-1}(y)$ lại rất ***"khó"*** . Tuy nhiên có cửa sổ sập **z** để tính x = $f^{-1}(y)$ là ***"dễ".***

Ví dụ:

Hàm f(x) = x^a (mod n) (với n là tích của hai số nguyên tố lớn n = p*q) là hàm một phía. Nếu chỉ biết a và n thì tính x = $f^{-1}(y)$ rất ***"khó"*** , nhưng nếu biết ***cửa sập*** p và q, thì tính được $f^{-1}(y)$ là khá ***"dễ".***

1.2. VẤN ĐỀ MÃ HÓA DỮ LIỆU

1.2.1. Khái niệm Mã hóa

Để bảo đảm **An toàn thông tin (ATTT)** lưu trữ trong máy tính (giữ gìn thông tin cố định) hay bảo đảm An toàn thông tin trên đường truyền tin (trên mạng máy tính), người ta phải ***"Che Giấu"*** các thông tin này.

"Che" thông tin (dữ liệu) hay ***"Mã hóa"*** thông tin là ***thay đổi hình dạng*** thông tin gốc, và người khác ***"khó"*** nhận ra.

"Giấu" thông tin (dữ liệu) là ***cất giấu*** thông tin trong bản tin khác, và người khác cũng ***"khó"*** nhận ra.

Trong phần này chúng ta bàn về ***"Mã hóa"*** thông tin.

1/. Hệ mã hóa:

Việc mã hóa phải theo quy tắc nhất định, quy tắc đó gọi là ***Hệ mã hóa***.

Hệ mã hóa được định nghĩa là bộ năm **(P, C, K, E, D)**, trong đó:

P là tập hữu hạn các ***bản rõ*** có thể.

C Là tập hữu hạn các ***bản mã*** có thể.

K là tập hữu hạn các ***khóa*** có thể.

E là tập hữu hạn các hàm lập mã.

D là tập các hàm giải mã.

Với khóa lập mã **ke** $\in$ **K**, có hàm lập mã $e_{ke} \in$ **E**, e_{ke} **:** $\mathbf{P} \rightarrow \mathbf{C}$,

Với khóa giải mã **kd** $\in$ **K**, có hàm giải mã $d_{kd} \in$ **D,** d_{kd} **:** $\mathbf{C} \rightarrow \mathbf{P}$**,**

sao cho $d_{kd}(e_{ke}(\mathbf{x})) = \mathbf{x}$, $\forall\ \mathbf{x} \in \mathbf{P}$.

Ở đây **x** được gọi là ***bản rõ***, e_{ke}**(x)** được gọi là ***bản mã***.

2/. Mã hóa và Giải mã:

Người gửi G	$\rightarrow\ \rightarrow$ e_{ke}**(T)**	$\rightarrow\ \rightarrow$ Người nhận N
(có khóa lập mã **ke**)		(có khóa giải mã **kd**)
	↑	
	Tên tặc có thể trộm bản mã e_{ke}**(T)**	

Người gửi G muốn gửi bản tin T cho người nhận N. Để bảo đảm bí mật, G mã hóa bản tin bằng khóa lập mã ke, nhận được bản mã e_{ke} **(T),** sau đó gửi cho N. Tên tặc có thể trộm bản mã e_{ke} **(T),** nhưng cũng ***"khó"*** hiểu được bản tin gốc T nếu không có khóa giải mã **kd**.

Người N nhận được bản mã, họ dùng khóa giải mã **kd**, để giải mã e_{ke} **(T),** sẽ nhận được bản tin gốc **T** = d_{kd} **(**e_{ke} **(T)).**

1.2.2. Phân loại mã hóa

Hiện có 2 loại mã hóa chính: mã hóa khóa đối xứng và mã hóa khóa công khai.

Hệ mã hóa khóa đối xứng có khóa lập mã và khóa giải mã "giống nhau", theo nghĩa biết được khóa này thì ***"dễ"*** tính được khóa kia. Vì vậy phải giữ bí mật cả 2 khóa.

Hệ mã hóa khói công khai có khóa lập mã khác khóa giải mã (**ke** ≠ **kd**), biết được khóa này cũng ***"khó"*** tính được khóa kia. Vì vậy cần bí mật khóa giải mã, còn công khai khóa lập mã.

1.2.2.1. Hệ mã hóa khóa đối xứng

Mã hóa khóa đối xứng là Hệ mã hóa mà biết được khóa lập mã thì có thể "dễ" tính được khóa giải mã và ngược lại. Đặc biệt một số Hệ mã hóa có khóa lập mã và khóa giải mã trùng nhau (**ke** = **kd**), như Hệ mã hóa "dịch chuyển" hay DES.

Hệ mã hóa khóa đối xứng còn gọi là ***Hệ mã hóa khóa bí mật,*** hay ***khóa riêng,*** vì phải giữ bí mật cả 2 khóa. Trước khi dùng Hệ mã hóa khóa đối xứng, người gửi và người nhận phải thỏa thuận thuật toán mã hóa và khóa chung (lập mã hay giải mã), khóa phải giữ bí mật. Độ an toàn của Hệ mã hóa loại này ***phụ thuộc vào khóa.***

Ví dụ:

+ ***Hệ mã hóa cổ điển*** là Mã hóa khóa đối xứng: dễ hiểu, dễ thực thi, nhưng có độ an toàn không cao. Vì giới hạn tính toán chỉ trong phạm vi bảng chữ cái, sử dụng trong bản tin cần mã, ví dụ là Z_{26} nếu dùng các chữ cái tiếng Anh. Với hệ mã hóa cổ điển, nếu biết khóa lập mã hay thuật toán lập mã, có thể "dễ" xác định được bản rõ, vì "dễ" tìm được khóa giải mã.

+ ***Hệ mã hóa DES*** (1973) là Mã hóa khóa đối xứng ***hiện đại***, có độ an toàn cao.

1/. Đặc điểm của Hệ mã hóa khóa đối xứng.

Ưu điểm:

Hệ mã hóa khóa đối xứng mã hóa và giải mã nhanh hơn Hệ mã hóa khóa công khai.

Hạn chế:

+ Mã hóa khóa đối xứng chưa thật an toàn với lý do sau:

Người nhận mã hóa và người giải mã phải có ***"chung"*** một khóa. Khóa phải được giữ bí mật tuyệt đối, vì biết khóa này ***"dễ"*** xác định được khóa kia và ngược lại.

+ Vấn đề thỏa thuận khóa và quản lý khóa chung là khó khăn và phức tạp. Người gửi và người nhận phải luôn thống nhất với nhau về khóa. Việc thay đổi khóa là rất khó và dễ bị lộ. Khóa chung phải được gửi cho nhau trên kênh an toàn.

Mặt khác khi hay người (lập mã, giải mã) cũng biết "chung" một bí mật, thì càng khó giữ được bí mật!

2/. Nơi sử dụng Hệ mã hóa khóa đống xứng.

Hệ mã hóa khóa đối xứng thường được sử dụng trong môi trường mà khóa chung có thể dễ dàng trao chuyển bí mật, chẳng hạn trong cùng một mạng nội bộ. Hệ mã hóa khóa đối xứng thường dùng để mã hóa những bản tin lớn, vì tốc độ mã hóa và giải mã nhanh hơn Hệ mã hóa khóa công khai.

1.2.2.2. Hệ mã hóa khóa công khai

Hệ mã hóa khóa phi đối xứng là Hệ mã hóa có khóa lập mã và khóa giải mã khác nhau (**ke ≠ kd**), biết được khóa này cũng ***"khó"*** tính được khóa kia.

Hệ mã hóa này còn được gọi là **Hệ mã hóa khóa công khai**, vì:

Khóa lập mã cho ***công khai***, gọi là ***khóa công khai (Public key)***.

Khóa giải mã giữ bí mật, còn gọi là ***khóa riêng (Private key)*** hay ***khóa bí mật***.

Một người bất kỳ có thể dùng khóa công khai để mã hóa bản tin, nhưng chỉ người nào có đúng khóa giải mã thì mới có khả năng đọc được bản rõ.

Hệ mã hóa khóa công khai hay ***Hệ mã hóa phi đối xứng*** do Diffie và Hellman phát minh vào những năm 1970.

1/. Đặc điểm của Hệ mã hóa khóa công khai.

Ưu điểm:

+ Hệ mã hóa khóa công khai có ưu điểm chủ yếu sau:

Thuật toán được viết một lần, công khai cho nhiều lần dùng, cho nhiều người dùng, họ chỉ cần giữ bí mật khóa riêng của mình.

+ Khi biết các tham số ban đầu của hệ mã hóa, việc tính ra cặp khóa công khai và bí mật là "dễ", tức là trong thời gian đa thức.

Người gửi có bản rõ P và khóa công khai, thì "dễ" tạo ra bản mã C.

Người nhận có bản mã C và khóa bí mật, thì "dễ" giải được thành bản rõ P.

+ Người mã hóa dùng khóa công khai, người giải mã giữ khóa bí mật. Khả năng lộ khóa bí mật khó hơn vì chỉ có một người giữ gìn.

Nếu thám mã biết khóa công khai, cố gắng tìm khóa bí mật, thì chúng phải đương đầu với bài toán "khó".

+ Nếu thám mã biết khóa công khai và bản mã C, thì việc tìm ra bản rõ P cũng là bài toán "khó", số phép thử là vô cùng lớn, không khả thi.

Hạn chế:

Hệ mã hóa khóa công khai: mã hóa và giải mã ***chậm hơn*** hệ mã hóa khóa đối xứng.

2/. Nơi sử dụng Hệ mã hóa khóa công khai.

Hệ mã hóa khóa công khai thường được sử dụng chủ yếu trên các mạng công khai như Internet, khi mà việc trao chuyển khóa bí mật tương đối khó khăn.

Đặc trưng nổi bật của hệ mã hóa công khai là khóa công khai (public key) và bản mã (ciphertext) đều có thể gửi đi trên một kênh truyền tin ***không an toàn***.

Có biết cả khóa công khai và bản mã, thì thám mã cũng không dễ khám phá được bản rõ.

Nhưng vì có tốc độ mã hóa và giải mã ***chậm***, nên hệ mã hóa khóa công khai chỉ dùng để mã hóa những bản tin ngắn, vì dụ như mã hóa khóa bí mật gửi đi.

Hệ mã hóa khóa công khai thường được sử dụng cho cặp người dùng thỏa thuận khóa bí mật của Hệ mã hóa khóa riêng.

1.3. VẤN ĐỀ CHỮ KÝ SỐ

1.3.1. Khái niệm "chữ ký số"

1.3.1.1. Giới thiệu "chữ ký số"

Để chứng thực nguồn gốc hay hiệu lực của một tài liệu (ví dụ: đơn xin học, giấy báo nhập học, ...), lâu nay người ta dùng chữ ký ***"tay"***, ghi vào phía dưới của mỗi tài liệu. Như vậy người ký phải ***trực tiếp "ký tay"*** vào tài liệu.

Ngày nay các tài liệu được số hóa, người ta cũng có nhu cầu chứng thực nguồn gốc hay hiệu lực của các tài liệu này. Rõ ràng không thể "ký tay" vào tài liệu, vì chúng không được in ấn trên giấy. Tài liệu "số" (hay tài liệu "điện tử") là một xâu các bít (0 hay 1), xâu bít có thể rất dài (nếu in trên giấy có thể hàng nghìn trang). "Chữ ký" để chứng thực một xâu bít tài liệu cũng không thể là một xâu bít nhỏ đặt phía dưới xâu bít tài liệu. Một "chữ ký" như vậy chắc chắn sẽ bị kẻ gian sao chép để đặt dưới một tài liệu khác bất hợp pháp.

Những năm 80 của thế kỷ 20, các nhà khoa học đã phát minh ra ***"chữ ký số"*** để chứng thực một ***"tài liệu số"***. Đó chính là ***"bản mã"*** của xâu bít tài liệu.

Người ta tạo ra ***"chữ ký số"*** (chữ ký điện tử) trên ***"tài liệu số"*** giống như tạo ra ***"bản mã"*** của tài liệu với "khóa lập mã".

Như vậy ***"ký số"*** trên ***"tài liệu số"*** là ***"ký"*** trên từng bít tài liệu. Kẻ gian khó thể giả mạo "chữ ký số" nếu nó không biết "khóa lập mã".

Để kiểm tra một ***"chữ ký số"*** thuộc về một ***"tài liệu số"***, người ta giải mã ***"chữ ký số"*** bằng "khóa giải mã", và so sánh với tài liệu gốc.

Ngoài ý nghĩa để chứng thực nguồn gốc hay hiệu lực của các tài liệu số hóa. Mặt mạnh của ***"chữ ký số"*** hơn ***"chữ ký tay"*** là ở chỗ người ta có thể ***"ký"*** vào tài liệu từ rất xa trên mạng công khai. Hơn thế nữa, có thể ***"ký"*** bằng các thiết bị cầm tay (ví dụ điện thoại di động) tại khắp mọi nơi (Ubikytous) và di động (Mobile), miễn là kết nối được vào mạng. Đỡ tốn bao thời gian, sức lực, chi phí, ...

"Ký số" thực hiện trên từng bít tài liệu, nên độ dài của "***chữ ký số "*** ít nhất cũng bằng độ dài tài liệu. Do đó thay vì ký trên tài liệu dài, người ta thường dùng ***"hàm băm"*** để tạo ***"đại diện"*** cho tài liệu, sau đó mới "Ký số" lên ***"đại diện"*** này.

1.3.1.2. Sơ đồ "chữ ký số"

Sơ đồ chữ ký là bộ năm **(P, A, K, S, V**), trong đó:

P là tập hữu hạn các văn bản có thể.

A là tập hữu hạn các chữ ký có thể.

K là tập hữu hạn các khóa có thể.

S là tập các thuật toán ký.

V là tập các thuật toán kiểm thử.

Với mỗi khóa $k \in \mathbf{K}$, có thuật toán ký $\mathbf{Sig}_k \in \mathbf{S}$, $\mathbf{Sig}_k : \mathbf{P} \rightarrow \mathbf{A}$,

có thuật toán kiểm tra chữ ký $\mathbf{Ver}_k \in \mathbf{V}$, $\mathbf{Ver}_k : \mathbf{P \times A} \rightarrow$ {đúng, sai},

thỏa mãn điều kiện sau với mọi $\mathbf{x} \in \mathbf{P}$, $\mathbf{y} \in \mathbf{A}$:

$$\mathbf{Ver}_k(\mathbf{x}, \mathbf{y}) = \begin{cases} \text{Đúng, nếu } \mathbf{y} = \mathbf{Sig}_k(\mathbf{x}) \\ \text{Sai, nếu } \mathbf{y} \neq \mathbf{Sig}_k(\mathbf{x}) \end{cases}$$

Chú ý:

Người ta thường dùng hệ mã hóa khóa công khai để lập ***"Sơ đồ chữ ký số"***. Ở đây khóa bí mật **a** dùng làm khóa ***"ký"***, khóa công khai **b** dùng làm khóa kiểm tra ***"chữ ký".***

Ngược lại với việc mã hóa, dùng khóa công khai **b** để lập mã, dùng khóa bí mật **a** để giải mã.

Điều này là hoàn toàn tự nhiên, vì ***"ký"*** cần giữ bí mật nên phải dùng khóa bí mật **a** để ***"ký"***. Còn ***"chữ ký"*** là công khai cho mọi người biết, nên họ dùng khóa công khai **b** để kiểm tra.

1.3.2. Phân loại "chữ ký số"

Có nhiều loại chữ ký tùy theo cách phân loại, sau đây xin giới thiệu một số cách.

1.3.2.1. Phân loại chữ ký theo đặc trưng kiểm tra chữ ký

1/. Chữ ký khôi phục thông điệp:

Là loại chữ ký, trong đó người gửi chỉ cần gửi ***"chữ ký"*** , người nhận có thể khôi phục lại được thông điệp, đã được ***"ký"*** bởi ***"chữ ký"*** này.

Ví dụ: Chữ ký RSA là chữ ký khôi phục thông điệp, sẽ trình bày trong mục sau.

2/. Chữ ký đi kèm thông điệp:

Là loại chữ ký, trong đó người gửi chỉ cần gửi ***"chữ ký"*** , phải gửi kèm cả thông điệp đã được ***"ký"*** bởi ***"chữ ký"*** này. Ngược lại, người nhận sẽ không có được thông điệp gốc.

Ví dụ: Chữ ký Elgamal là chữ ký đi kèm thông điệp, sẽ trình bày trong mục sau.

1.3.2.2. Phân loại chữ ký theo mức an toàn

1/. Chữ ký "không thể phủ nhận":

Nhằm tránh việc nhân bản chữ ký để sử dụng nhiều lần, tốt nhất là người gửi tham gia trực tiếp vào việc kiểm thử chữ ký. Điều đó được thực hiện bằng một giao thức kiểm thử, dưới dạng một giao thức mời hỏi và trả lời.

Ví dụ: Chữ ký không phủ định (Chaum- van Antverpen), trình bày trong mục sau.

2/. Chữ ký "một lần":

Để bảo đảm an toàn, "Khóa ký" chỉ dùng 1 lần (one - time) trên 1 tài liệu.

Ví dụ: Chữ ký một lần Lamport. Chữ ký Fail - Stop (Van Heyst & Pedersen).

1.3.2.3. Phân loại chữ ký theo ứng dụng đặc trưng

Chữ ký "mù" (Blind Signature).

Chữ ký "nhóm" (Group Signature).

Chữ ký "bội" (Multy Signature).

Chữ ký "mù nhóm" (Blind Group Signature).

Chữ ký "mù bội" (Blind Multy Signature).

1.4. MỘT SỐ BÀI TOÁN QUAN TRỌNG TRONG MẬT MÃ

Trong phần này sẽ xét ba bài toán có vai trò quan trọng trong lý thuyết mật mã, đó là ba bài toán: Kiểm tra số nguyên tố, phân tích một số nguyên thành tích của các thừa số nguyên tố, tính logarit rời rạc của một số theo modulo nguyên tố. Ở đây ta mặc định rằng các số nguyên tố là rất lớn.

1.4.1. Bài toán kiểm tra số nguyên tố lớn

Cho n là số nguyên bất kỳ. Làm thế nào để biết n là số nguyên tố hay không? Bài toán được đặt ra từ những buổi đầu của số học, và trải qua hơn 2000 năm đến nay vẫn là một bài toán chưa có được những cách giải dễ dàng. Bằng những phương pháp đơn giản như phương pháp sàng Eurratosthène, từ rất sớm người ta đã xây dựng được các bảng số nguyên tố đầu tiên, rồi tiếp tục bằng nhiều phương pháp khác tìm thêm được nhiều số nguyên tố lớn.

Tuy nhiên chỉ đến giai đoạn hiện nay của lý thuyết mật mã hiện đại, nhu cầu sử dụng các nguyên tố và thử tính nguyên tố của các số mới trở thành một nhu cầu to lớn và phổ biến, đòi hỏi nhiều phương pháp mới có hiệu quả hơn.

Trong mục này sẽ lược qua vài tính chất của số nguyên tố và một vài phương pháp thử tính nguyên tố của một số nguyên bất kỳ.

1/. Tiêu chuẩn Euler-Solovay-Strassen:

a) Nếu n là số nguyên tố, thì với mọi số nguyên dương a ≤ n-1:

$$\left(\frac{a}{b}\right) \equiv a^{(n-1)/2} \bmod n.$$

b) Nếu n là hợp số, thì:

$$\left|\left\{a : 1 \le a \le n-1, \left(\frac{a}{b}\right) \equiv a^{(n-1)/2} \bmod n\right\}\right| \le \frac{n-1}{2}.$$

2/. Tiêu chuẩn Solovay-Strassen-Lehmann:

a) Nếu n là số nguyên tố, thì với mọi số nguyên dương a ≤ n-1:

$$a^{(n-1)/2} \equiv \pm 1 \bmod n$$

b) Nếu n là hợp số thì

$$\left|\{a : 1 \le a \le n-1, a^{(n-1)/2} \equiv \pm 1 \bmod n\}\right| \le \frac{n-1}{2}$$

3). Tiêu chuẩn Miler-Rabin:

a) Cho n là số nguyên lẻ, ta viết (n-1) = 2^e.u, với u là số lẻ. Nếu n là số nguyên tố, thì với mọi số nguyên dương a ≤ n-1:

$$a^u \equiv a \bmod n \vee \exists k < e(a^{2^k.u} \equiv -1 \bmod n).$$

b) Nếu n là hợp số, thì

$$\left|\{a : 1 \le a \le n-1, (a^u \equiv 1 \bmod n) \vee \exists k < e(a^{2^k.u} \equiv -1 \bmod n)\}\right| \le \frac{n-1}{2}$$

Các tiêu chuẩn kể trên là cơ sở để ta xây dựng các thuật toán xác suất kiểu Monte-Carlo thử tính nguyên tố (hay hợp số) của các số nguyên.

Thuật toán Euler-Solovay-Strassen.

Dữ liệu vào: số nguyên dương n và t số ngẫu nhiên $a_1, ..., a_t$

$(1 \le a_i \le n-1)$,

1. **for** i = 1 **to** t **do**
2. **if** $(a_i / n \equiv a_i^{(n-1)/2} \bmod n)$, then
3. **answer** "n là số nguyên tố"
4. **else**
5. **answer** "n là hợp số" and **quit**

Nếu thuật toán cho trả lời "n là hợp số" thì đúng n là hợp số.

Nếu thuật toán cho trả lời "n là số nguyên tố", thì trả lời đó có thể sai với xác suất Monte-Carlo thiên về có, nếu xem nó là thuật toán thử tính là hợp số. Thuật toán xác suất thiên về không, nếu nó là thuật toán thử tính nguyên tố của các số nguyên.

Tương tự, dựa vào các tiêu chuẩn 2 và 3, người ta đã xây dựng các thuật toán xác suất Solovay-Strassen-Lehmann và Miler-Rabin kiểu Monte-Carlo để thử tính nguyên tố (hay hợp số) của các số nguyên.

Hai thuật toán đó chỉ khác thuật toán Euler-Solovay-Strassen ở chỗ công thức trong hàng lệnh 2 cần được thay tương ứng bởi

$$a^{(n-1)/2} \equiv \pm 1 \bmod n$$

hay

$$\left|\{(a^u \equiv 1 \bmod n) \vee \exists k < e(a^{2^k.u} \equiv -1 \bmod n)\}\right|$$

trong đó u và e được xác định bởi: (n-1) = 2^e.u, u là số lẻ.

Xác suất sai lầm ε khi nhận được kết quả "n là số nguyên tố" trong các thuật toán trên được tính như sau:

Giả sử n là số lẻ trong khoảng N và 2N, tức N<n<2N. Gọi A là sự kiện "n là số nguyên tố", và B là sự kiện "thuật toán cho kết quả trả lời n là số nguyên tố".
Ta phải tính xác suất ε = p(A|B).

Theo tính chất (b) của tiêu chuẩn Euler-Solovay-Strassen, nếu n là hợp số, thì sự kiện $\left(\frac{a}{b}\right) \equiv a^{(n-1)/2} \bmod n$

đối với mỗi a ngẫu nhiên (1 ≤ a ≤ n-1) có xác suất ≤ 1/2, vì vậy ta có

$$\text{p(B/A)} \le \frac{1}{2^t}.$$

Theo công thức Bayes ta có

$$p(A/B) = \frac{p(B/A).p(A)}{p(B)} = \frac{p(A/B).p(A)}{p(B/A).p(A).p(B/\overline{A}).p(\overline{A})}$$

Theo định lý về số nguyên tố, số các số nguyên tố giữa N và 2N xấp xỉ
N/ lnN ≈ n/ ln n, số các số lẻ là N/2 ≈ n/2, do đó p($\overline{A}$) ≈ 2/ ln n và p($\overline{A}$) ≈ 1-2/ln n. Dĩ nhiên ta có p(B/$\overline{A}$) = 1. Thay các giá trị đó vào công thức trên, ta được:

$$p(A/B) \le \frac{2^{-t}(1-\frac{2}{\ln n})}{2^{-t}(1-\frac{2}{\ln n})+\frac{2}{\ln n}} = \frac{\ln n - 2}{\ln n - 2 + 2^{t+1}} \qquad (1.1)$$

Chú ý:

Đánh giá đó cũng đúng đối với thuật toán Solovay-Strassen-Lehmann. Đối với thuật toán Miler-Rabin, ta được một đánh giá tốt hơn, cụ thể là:

$$p(A/B) = \frac{\ln n - 2}{\ln n - 2 + 2^{t+1}} \quad (1.2)$$

Chú ý rằng khi t = 50 thì đại lượng ở vế phải của (1.1) $\approx 10^{-13}$, và vế phải của (1.2) $\approx 10^{-28}$; do đó nếu chọn cho dữ liệu vào năm mươi số ngẫu nhiên a_i thì các thuật toán Euler-Solovay-Strassen và Solovay-Lehmann sẽ thử cho ta một số nguyên tố với xác suất sai lầm $\leq 10^{-13}$ và thuật toán Miler-Rabin với xác suất sai lầm là $\leq 10^{-28}$.

Có thể tính được độ phức tạp tính toán về thời gian của các thuật toán xác suất kể trên vào cỡ $\log_n$, tức là đa thức theo độ dài biểu diễn của dữ liệu vào (số n).

Tuy nhiên các thuật toán đó chỉ cho ta tính thử nguyên tố của một số với một xác suất sai lầm ε nào đó, dù ε là rất bé. Trong nhiều ứng dụng ta muốn có được số nguyên tố với độ chắc chắn 100% là số nguyên tố. Khi đó ta có thể dùng các thuật toán xác suất như trên và sau đó tìm kiếm những thuật toán tất định để thử tính nguyên tố với độ chính xác tuyệt đối.

Adleman, Pomerance và Rumely đã đề xuất một số thuật toán kiểu như vậy, trong đó nổi bật là thuật toán thử tổng Jacobi, sau đó được đơn giản hóa bởi Cohen và Lenstra. Gold Wasser, Kilian, Adleman và Hoang đề xuất thuật toán thử bằng đường cong Elliptic, và được tiếp tục hoàn thiện bởi Atkin và Morain.
Các thuật toán này đã được dùng để tìm nhiều số nguyên tố lớn.

4/. Thuật toán Agrawal-Kayal-Saxene

Tháng 8-2002, các nhà toán học Ấn độ Agrawal, Kayal và Sexena đưa ra thuật toán tất định, thử tính nguyên tố, có độ phức tạp thời gian đa thức.

Thuật toán Agrawal-Kayal-Saxena.

Input: integer n > 1

1. If (n is of the form a^b, b > 1) output COMPOSITE;
2. r = 2;
3. While (r < n) {
4. if (gcd(n, r) ≠ 1) output COMPOSITE;
5. if (r is prime)
6. let q be the largest prime factor of r -1 ;
7. if (q ≥ 4 $\sqrt{r}$ log n) and ($n^{\frac{r-1}{q}} \neq 1 (\mathrm{mod}\, r)$)
8. break;
9. r ← r + 1;
10. }
11. for a = 1 to 2 $\sqrt{r}$ log n
12. if $(x-a)^n = (x^n - a)(\mathrm{mod}\, x^r - 1, n)$ output COMPOSITE;
13. output PRIME;

Thuật toán này đã được một số nhà toán học kiểm nghiệm, đánh giá cao và xem là thuật toán tốt, có thể dùng cho việc kiểm thử tính nguyên tố của các số nguyên. Trong thực tiễn xây dựng các giải pháp mật mã, có nhu cầu các số nguyên tố rất lớn. Để tìm được số như vậy, người ta chọn ngẫu nhiên một số n rất lớn, dùng một thuật toán xác suất, chẳng hạn như thuật toán Miller-Rabin. Nếu thuật toán cho kết quả "n là số nguyên tố" với một xác suất sai ε nào đó, thì dùng tiếp một thuật toán tất định (chẳng hạn thuật toán Thuật toán Agrawal-Kayal-Saxena) để đảm bảo chắc chắn 100% rằng số n là nguyên tố. Thuật toán Agrawal-Kayal-Saxena được chứng tỏ là có độ phức tạp thời gian đa thức cỡ $O((\log n)^{12})$ khi thử trên số n.

1.4.2. Bài toán phân tích thành thừa số nguyên tố

Bài toán phân tích một số nguyên thành thừa số nguyên tố cũng được xem là bài toán khó, thường được sử dụng trong lý thuyết mật mã. Biết số n là hợp số thì việc phân tích n thành các thừa số, mới là có nghĩa; do đó để phân tích n thành các thừa số, ta thử trước n có phải là hợp số hay không.

Bài toán phân tích n thành các thừa số có thể dẫn về bài toán ***tìm một ước số của n***. Vì biết một ước số d của n, thì tiến trình phân tích n được tiếp tục thực hiện bằng cách phân tích d và n/d.

Bài toán phân tích thành các thừa số, hay bài toán tìm ước số của một số nguyên cho trước, đã được nghiên cứu nhiều, nhưng cũng chưa có thuật toán hiệu quả nào để giải nó trong trường hợp tổng quát. Do đó người ta có khuynh hướng tìm thuật toán giải nó trong những trường hợp đặc biệt, chẳng hạn khi n có một ước số nguyên tố p với p – 1 là ***B-min***, hoặc khi n kà số Blum, tức là số có dạng tích của hai số nguyên tố lớn nào đó n = p.q.

Một số nguyên n được gọi là ***B-min*** nếu tất cả các ước số nguyên tố của nó đều ≤ B với một cận B > 0 nào đó.

1/. Trường hợp 1.

Giải sử n là ***B-min***. Ký hiệu Q là bội chung bé nhất của các lũy thừa của các số nguyên tố ≤ B mà bản thân chúng ≤ n. Nếu $q^l \le n$ thì l ln(q) ≤ ln n, tức l≤ $\left\lfloor \frac{\ln n}{\ln q} \right\rfloor$

(⌊x⌋ là số nguyên bé nhất lớn hơn x).

Ta có $Q = \prod_{q \le B} q^{\lfloor \ln n / \ln q \rfloor}$

trong đó tích lấy theo tất cả các số nguyên tố khác nhau q ≤ B.

Nếu p là thừa số nguyên tố của n sao cho p-1 là ***B-min,*** thì p-1|Q, và do đó với mọi a bất kỳ thỏa mãn gcd(a, p) = 1. Theo định lý Fermat ta có $a^Q \equiv q \bmod p$

Vì vậy nếu lấy d = gcd(a^Q-1, n) thì p|d.

Nếu d = n thì coi như thuật toán không cho ta điều mong muốn, tuy nhiên điều đó chắc không xảy ra nếu n có ít nhất hai thừa số nguyên tố khác nhau.

(p-1)-Thuật toán Pollard phân tích thành thừa số:

INPUT: Một hợp số n không phải là lũy thừa của một số nguyên tố.

OUTPUT: Một thừa số không tầm thường của n.

1. Chọn một cận cho độ mịn B.
2. Chọn ngẫu nhiên một số nguyên a, $2 \le a \le n-1$, và tính d = gcd(a, n).

Nếu $d \ge$ thì cho ra kết quả (d).

3. Với mỗi số nguyên tố $q \le B$ thực hiện:

3.1. Tính $l = \left\lfloor \frac{\ln n}{\ln q} \right\rfloor$.

3.2. Tính a $\leftarrow a^{q^l} \bmod n$

4. Tính d = gcd(a-1, n).
5. Nếu $1 < d < n$ thì cho kết quả (d).

Nếu ngược lại thì coi như không có kết quả.

2/. Trường hợp 2.

Xét trường hợp số nguyên Blume, tức là số có dạng n = p.q, tích của hai số nguyên tố lớn.

Chú ý rằng nếu biết hai số nguyên khác nhau x và y sao cho $x^2 \equiv y^2 (\bmod n)$ thì dễ tìm được một thừa số của n.

Thực vậy, từ $x^2 \equiv y^2 (\bmod n)$ ta có thể suy ra rằng $x^2 - y^2 (x+y)(x-y)$ chia hết cho n, do n không là ước số của x + y hoặc x – y, nên gcd(x-y, n) phải là một ước số của n, tức bằng p hoặc q.

Ta biết nếu n = p.q là số Blume, thì phương trình đồng dư $x^2 \equiv a^2 (\bmod n)$ có 4 nghiệm, hai nghiệm tầm thường là x = a và x = -a. Hai nghiệm không tầm thường khác là ±b, chúng là nghiệm của hai hệ phương trình đồng dư bậc nhất sau:

$$\left\{ \begin{matrix} x = a(\bmod p) \\ x = -a(\bmod q) \end{matrix} \right\} \qquad \left\{ \begin{matrix} x = -a(\bmod p) \\ x = a(\bmod q) \end{matrix} \right\}$$

Bằng lập luận như trên, ta thấy rằng n là số Blume, a là số nguyên tố với n, và ta biết một nghiệm không tầm thường của phương trình $x^2 \equiv a^2 (\mathrm{mod} n)$, tức là biết $x \neq \pm a$ sao cho $x^2 \equiv a^2 (\mathrm{mod} n)$ thì gcd(x-a, n) sẽ là một ước số của n.

Từ những điều rút ra ở trên, người ta đã tìm ra một số phương pháp tìm ước số nguyên tố của một số nguyên dạng Blume. Các phương pháp đó dựa vào việc tìm một nghiệm không tầm thường của phương trình $x^2 \equiv 1 (\mathrm{mod} n)$.

Ta giả thiết $a.b = 2^s.r$ với r là số lẻ.

Ta phát triển một thuật toán xác suất kiểu Las Vegas như sau:

Chọn một số ngẫu nhiên v ($1 \leq v \leq n-1$). Nếu v may mắn là bội số của p hay q, thì ta được ngay một ước số của n là gcd(v, n).

Nếu v nguyên tố với n, thì ta tính các bình phương liên tiếp kể từ v^r, được v^r, v^{2r}, v^{4r}, ... cho đến khi được $v^{2^t.r} \equiv 1 (\mathrm{mod} n)$ với một t nào đó.

Số t như vậy bao giờ cũng đạt được, vì có $2^s.r \equiv 0 (\mathrm{mod} \Phi(n))$

nên có $v^{2^t.r} \equiv 1 (\mathrm{mod} n)$.

Như vậy ta đã tìm được số $x = v^{2^{t-1}.r}$ sao cho $x^2 \equiv 1 (\mathrm{mod} n)$. Tất nhiên có $x \neq 1 \mod n$.

Nếu cũng có $x \neq -1 \mod n$ thì x là nghiệm không tầm thường của $x^2 \equiv 1 (\mathrm{mod} n)$, từ đó ta có thể tìm ước số của n.

Nếu không thì thuật toán cho kết quả không đúng.

Người ta có thể ước lượng xác suất cho kết quả không đúng với một lần thử với một số v là < ½, do đó nếu thiết kế thuật toán với m số ngẫu nhiên $v_1, v_2, ..., v_m$, thì sẽ đạt được xác suất kết quả không đúng là $< 1/2^m$.

1.4.3. Bài toán tính logarit rời rạc theo modulo

Cho p là số nguyên tố và α là phần tử nguyên thủy theo mod p. Bài toán tính logarit rời rạc theo mod p là bài toán tìm, với mỗi số $\beta \in Z_p^*$, một số a ($1 \le a \le p-1$) sao cho $\beta = \alpha^a \bmod p$, tức là $a = \log_\alpha \beta (\bmod p-1)$.

Một thuật toán tầm thường để giải bài toán này là duyệt toàn bộ các số a từ q đến p-1, cho đến khi tìm được a thỏa mãn $\beta = \alpha^a \bmod p$.

Tuy nhiên thuật toán này sẽ không hiệu quả nếu p là số rất lớn. Một biến dạng của thuật toán đó với ít nhiều hiệu quả hơn là thuật toán Shanks.

1/. Thuật toán Shanks.

Đặt $m = \lfloor \sqrt{p-1} \rfloor$. Ta tìm a dưới dạng a = mj + i, $0 \le i, j \le m-1$.

Rõ ràng $\beta = \alpha^a \bmod p$ khi và chỉ khi $\alpha^{mj} = \beta\alpha^i \bmod p$.

Ta lập hai danh sách gồm có các cặp (j, α^{mj}) và $(i, \beta\alpha^{-i})$ với i, j chạy từ 0 đến m-1. Khi phát hiện hai cặp từ hai danh sách đó có phần tử thứ hai bằng nhau là ta được kết quả a = mj + i, đó chính là giá trị $\log_\alpha \beta$ mà ta cần tìm. Thuật toán Shanks có độ phức tạp cỡ O(m) phép toán nhân và O(m) bộ nhớ (chứ kể $O(m^2)$) phép so sánh).

2/. Thuật toán Polig-Hellman.

Được dùng có hiệu quả trong trường hợp p-1 chỉ có các thừa số nguyên tố bé. Giả thiết rằng p-1 có dạng phân tích chính tắc là:

$$p-1 = \prod_{i=1}^{k} p_i^{c_i}$$

Để tìm a = $\log_\alpha \beta (\bmod p-1)$, ta tìm các số a_i sao cho $a_i \equiv$ a mod $P_i^{c_i}$ với i = 1,...,k.

Sau khi tìm được các a_i, thì hệ phương trình $x \equiv a_i \bmod P_i^{c_i}$ (i = 1,..., k), được giải theo định lý số dư Trung quốc, sẽ cho lời giải x = a (mod p-1) cần tìm.

Vấn đề là xác định các số $a_i \bmod P_i^{c_i}$ (i = 1,..., k). Vấn đề này phát biểu như sau:

Giả sử q là một ước số nguyên tố của p-1, q^c và | p-1, nhưng không còn q^{c+1}|p-1. Ta cần tìm x = $\bmod q^c$.

Ta biểu diễn x dưới dạng sau:

$$X = \sum_{i=0}^{c-1} X_i q_i \qquad (0 \le x_i \le \text{q-1})$$

Vì x = mod q^c nên a viết dưới dạng a = x + q^c .s và vì $\alpha^{p-1} \equiv 1 (\text{mod } p)$, nên ta có

$\beta^{(p-1)/q} \equiv \alpha^{\alpha(p-1)/q} \equiv (\alpha^{p-1})^{a/q} \equiv \alpha^{(p-1)x_0/q} (\text{mod } p)$

Ta đặt $\gamma = \alpha^{(p-1)/q}$, và tính lần lượt $\gamma^0, \gamma^1, \gamma^2, ...,$ đồng thời so sánh với $\beta^{(p-1)/q}$ mod p.

Ta lấy số i đó là x_0, tức x_0 = i.

Nếu c = 1 thì x = x_0, ta tìm xong x. Nếu c > 1 thì đặt $\beta^i = \beta\alpha^{-x}$

Tương tự như trên, tính lần lượt $\gamma^0, \gamma^1, \gamma^2, ...,$ đồng thời so sánh với $\beta^{(p-1)/q^2}$, ta tìm được x_1.

Cứ làm như vậy, ta tìm được dần các giá trị x_i với i = 0, 1, ..., c-1, tức tính được x.

Sau khi tìm được tất cả các giá trị của x ứng với mọi số nguyên tố q của p, thì theo một số nhận xét ở trên, chỉ cần giải tiếp một hệ phương trình đồng dư bậc nhất theo các modulo từng cặp nguyên tố với nhau (bằng phương pháp số dư Trung quốc), ta tìm được số a cần tìm, a = $\log_\alpha \beta$ theo mod p.

Thuật toán Polig-Hellman cho ta cách tính logarit rời rạc khá hiệu quả, nhưng chỉ khi p-1 chỉ có các thừa số nguyên tố bé. Nếu p-1 có ít nhất một thừa số nguyên tố lớn, thì thuật toán đó khó hiệu quả, trong trường hợp đó bài toán tính logarit rời rạc theo mod p vẫn là bài toán khó.

Một lớp các số nguyên tố p mà p-1 có ít nhất một thừa số nguyên tố lớn và lớp các số nguyên tố dạng p = 2q+1, trong đó q là số nguyên tố. Đó gọi là số nguyên tố dạng Sophie Germain, có vai trò quan trọng trong việc xây dựng các hệ mật mã khóa công khai.

Người ta đã nghiên cứu phát triển khá nhiều thuật toán khác, cả thuật toán tất định, cả thuật toán xác suất, để tính logarit rời rạc, nhưng chưa có thuật toán nào được chứng tỏ là có độ phức tạp thời gian đa thức.

Chương 2 . TẤN CÔNG CHỮ KÝ SỐ

2.1. TẤN CÔNG CHỮ KÝ RSA

2.1.1. Chữ ký RSA

2.1.1.1. Sơ đồ chữ ký

1/. Sơ đồ (đề xuất năm 1978)

**** Tạo cặp khóa (bí mật, công khai)* (a, b):**

Chọn bí mật số nguyên tố lớn p, q, tính n = p * q, công khai n, đặt P = C = Z_n

Tính bí mật $\phi(n) = (p-1).(q-1)$. Chọn khóa công khai $b < \phi(n)$, nguyên tố với $\phi(n)$.

Khóa bí mật a là phần tử nghịch đảo của b theo mod $\phi(n)$: $a*b \equiv 1 \pmod{\phi(n)}$.

Tập khóa (bí mật, công khai) $K = \{(a, b)/\ a,b \in Z_n, a*b \equiv 1 \pmod{\phi(n)}\}$.

**** Ký số:*** Chữ ký trên $x \in P$ là $y = Sig_k(x) = x^a \pmod n$, $y \in A$. (R1)

**** Kiểm tra chữ ký:*** $Ver_k(x, y) =$ đúng $\Leftrightarrow x \equiv y^b \pmod n$. (R2)

2/. Chú ý:

- So sánh giữa sơ đồ chữ ký RSA và sơ đồ mã hóa RSA ta thấy có sự tương ứng.
- Việc ký chẳng qua là mã hóa, việc kiểm thử lại chính là việc giải mã:

Việc "ký số" vào x tương ứng với việc "mã hóa" tài liệu x.

Kiểm thử chữ ký chính là việc giải mã "chữ ký", để kiểm tra xem tài liệu đã giải mã có đúng là tài liệu trước khi ký không. Thuật toán và khóa kiểm thử "chữ ký" là công khai, ai cũng có thể kiểm thử chữ ký được.

- Chữ ký RSA thuộc loại chữ ký khôi phục thông điệp.

2.1.1.2. Ví dụ Chữ ký trên x = 2

* Tạo cặp khóa (bí mật, công khai) = (a, b):

Chọn bí mật số nguyên tố p = 3, q = 5, tính n = p*q = 3*5 = 15, công khai n.

Đặt P = C = Z_n. Tính bí mật $\phi(n) = (p-1).(q-1) = 2*4 = 8$.

Chọn khóa công khai $b = 3 < \phi(n)$, nguyên tố với $\phi(n) = 8$.

Khóa bí mật a = 3, là phần tử nghịch đảo của b theo mod $\phi(n)$: $a*b \equiv 1 \pmod{\phi(n)}$.

* Ký số: Chữ ký trên $x = 2 \in P$ là $y = Sig_k(x) = x^a \pmod n = 2^3 \pmod{15} = 8$, $y \in A$.

* Kiểm tra chữ ký: $Ver_k(x, y) =$ đúng $\Leftrightarrow x \equiv y^b \pmod n \Leftrightarrow 2 \equiv 8^b \pmod{15}$.

2.1.2. Các dạng tấn công vào chữ ký RSA

2.1.2.1. Tấn công dạng 1: Tìm cách xác định khóa bí mật

1/. Kẻ tấn công chỉ biết khóa công khai của người ký.

Khi biết được n, thám mã tìm cách tính ra hai số nguyên tố p và q. Sau đó tính được Φ(n)=(p-1)(q-1) từ đó tính khóa bí mật a theo công thức $a.b \equiv 1 (\text{mod} \Phi(n))$.

→ Giải pháp phòng tránh:

Chọn số nguyên tố p, q lớn, để việc phân tích n thành tích 2 thừa số nguyên tố là khó có thể thực hiện được trong thời gian thực. Người ta thường sinh ra các số lớn (khoảng 100 chữ số), sau đó kiểm tra tính nguyên tố của nó.

2/. Kẻ tấn công biết được Φ(N).

Khi biết được Φ(n), kẻ tấn công có thể tính p, q theo hệ phương trình:

$$\begin{cases} p.q = n \\ (p-1)(q-1) = \Phi(n) \end{cases} \leftrightarrow \begin{cases} p.q = n \\ p+q = n+1-\Phi(n) \end{cases}$$

Do đó p và q là nghiệm của phương trình bậc hai :

$$x^2 - (n - \Phi(n) + 1)x + n = 0$$

Khi đã tính được Φ(n) chúng sẽ tính được khóa bí mật a theo công thức:

$a.b \equiv 1 (\text{mod } \Phi(n))$.

Biết được khóa bí mật thì kể tấn công sẽ giả mạo chữ ký của người dùng.

→ Giải pháp phòng tránh:

Chọn số nguyên tố p, q lớn để Φ(n) là một số lớn. Từ đó nếu thám mã có biết được Φ(n), thì việc tính ra khóa mật a cũng rất khó khăn.

3/. Nhiều người sử dụng chung modulo n.

Trong hệ thống có k người đăng ký sử dụng chữ ký RSA, trung tâm phân phối khóa (TT) sinh ra 2 số nguyên tố p, q, tính số modul n = p.q;

sinh ra các cặp khóa mã hóa/ giải mã (e_i, d_i). TT cấp cho người đăng ký thứ i khóa bí mật d_i tương ứng, cùng các thông tin bao gồm số n và danh sách đầy đủ khóa công khai e_i (i=1...k).

Bất kỳ người nào có thông tin công khai trên đều có thể:

Mã hóa văn bản M để gửi cho người đăng ký thứ i, bằng cách sử dụng thuật toán mã hóa RSA với khóa mã hóa e_i: Y = M^{e_i} mod n.

Người đăng ký thứ i có thể ký văn bản M bằng cách tính chữ ký $S_i = M^{d_i} \bmod n$. Bất cứ ai cũng có thể xác thực rằng M được ký bởi người đăng ký thứ i bằng cách tính $S_i^{e_i}$ mod n và so sánh với M. Sử dụng modul chung dẫn đến:

Trường hợp 1: Một thành viên có thể sử dụng khóa công khai và bí mật của mình để sinh ra khóa bí mật của người dùng khác.

Tức là căn cứ vào khóa công khai e_1, người giữ cặp khóa mã hóa/giải mã (e_2, d_2) có thể tìm được số nguyên d_1' sao cho $e_1 d_1' = 1 \bmod \Phi(n)$ mà không cần biết $\Phi(n)$.

Để tìm được số d_1' này, cần phải tìm một số nguyên t, nguyên tố cùng nhau với e_1 và là bội của $\Phi(n)$. Điều này thực hiện được bởi $(e, \Phi(n)) = 1$.

Khi đó, do t và e_1 nguyên tố cùng nhau nên tồn tại r và s sao cho rt + se_1 = 1. Vì t là bội của $\Phi(n)$ nên $s.e_1 \equiv 1 \bmod \Phi(n)$, và khi đó d_1' = s.

Thủ tục tìm số dư d_1' như sau : (trong đó chỉ cần đến các giá trị e_1, e_2, d_2).

1. Đặt $t = e_2 d_2 - 1$;
2. Sử dụng thuật toán Euclid mở rộng để tìm ước số chung lớn nhất f của e_1 và t. Đồng thời cũng phải tìm được hai số r và s thỏa r.t + s.e_1 = f.
3. Nếu f = 1 thì đặt d_1' = s và kết thúc ;
4. Nếu f $\neq$ 1 thì đặt t :=t/f, quay lại bước 2.

Hiển nhiên t nguyên tố cùng nhau với e_1. Theo các định nghĩa trên, ta biết rằng e_1 nguyên tố cùng nhau với $\Phi(n)$. Thủ tục trên đưa ra khóa giải mã e_1.

Vì độ phức tạp tính toán của thủ tục này là $O[(\log n)^2]$, nên đó là một khả năng đe dọa hệ thống. Một lần nữa, những thông tin có sẵn cho người dùng hợp pháp trong hệ thống thừa sức bẻ được hệ thống mật mã. Tất nhiên, người dùng này không thực hiện nguyên xi theo yêu cầu của nhà thiết kế giao thức dành cho người dùng, nhưng những thông tin cần thiết vẫn có thể lấy được mà người dùng không vi phạm quy định của giao thức.

Trường hợp 2: Nếu một văn bản được gửi tới hơn một người trong nhóm trên, thì đối phương có thể giải mã được văn bản mà không cần biết khóa giải mã.

Để chứng minh điều này, hãy xem kết quả của việc mã hóa văn bản M gửi cho hai người có khóa công khai tương ứng e_i và e_j:

$$Y_i = M^{e_i} \bmod n$$

$$Y_j = M^{e_j} \bmod n$$

Vì e_i và e_j là hai số nguyên tố cùng nhau, nên có thể tìm được các số nguyên r và s bằng thuật toán Euclid, thỏa: $re_i + se_j = 1$.

Rõ ràng, hoặc r hoặc s phải là số âm và trong trường hợp này ta giả sử r < 0 và viết r = -1.$|r|$.

Nếu Y_i hay Y_j không nguyên tố cùng nhau với n, ta hãy sử dụng thuật toán Euclid để tìm nghịch đảo của Y_i mod n.

Phép tính sau chỉ ra cách văn bản bị khám phá:

$$\left[Y_i^{-1}\right]^r \cdot \left[Y_j\right]^s = \left[M^{e_i}\right]^{-1r} \cdot \left[M^{e_i}\right]^s = M^{re_i+se} = M \bmod n .$$

Bởi vậy giao thức này thất bại trong việc bảo đảm bí mật văn bản M gửi tới các thành viên có khóa công khai là những số nguyên tố cùng nhau.

Chú ý rằng điều vừa trình bày ở trên không thể phá vỡ được hệ mật vì khả năng đọc được một văn bản M không thể dẫn tới khả năng đọc các văn bản tùy ý được mã hóa với cùng hệ thống đó.

Trường hợp 3: Việc sử dụng số modul chung cũng làm cho giao thức RSA dễ bị tấn công, trong đó một người đăng ký có thể bẻ được hệ mật.

Hệ mật bị sập, tất nhiên kênh bí mật bị lộ và người đăng ký này có thể giải mã các văn bản của người dùng khác, kênh chữ ký cũng hỏng vì anh ta có thể giả mạo chữ ký của người dùng mà không bị phát hiện.

Đó là sử dụng phương pháp xác suất để phân tích ra thừa số modul, hoặc sử dụng thuật toán tất định để tính toán số mũ giải mã mà không cần số modul.

Ý tưởng cơ bản của kiểu tấn công thứ hai là phân tích số modul n bằng cách tìm căn bậc hai không tầm thường của 1 mod n. Nghĩa là tìm một số b thỏa mãn:

$b^2 = 1 \bmod n$, b $\neq \pm$ mod n, 1 < b< n-1

Nếu tìm được số b như thế, thì số modul n có thể được phân tích theo cách sau.

Vì $b^2 = 1 \bmod n$ nên $b^2 - 1 = 0 \bmod n$.

(b+1)(b-1) = 0 mod n. Hay (b+1)(b-1) = sn = spq, s là số nguyên tùy ý.

Nghĩa là (b+1)(b-1) chia hết cho cả p và q.

Tuy nhiên, 1 < b < n-1, vì vậy 0 < b-1 < b+1 < n = p.q. Ta thấy:

Nếu b-1 chia hết cho p, thì không chia hết cho q. Tương tự với b+1.

Vì thế, ước số chung lớn nhất của b+1 và n phải là p hoặc q.

Áp dụng thuật toán Euclid, sẽ phân tích ra thừa số của n.

Vì vậy, cách tấn công vào hệ thống này tập trung vào cách để tìm căn bậc hai không tầm thường của 1 mod n.

Đặt e_1 và d_1 là khóa mã hóa và giải mã của người dùng hệ thống.

Theo định nghĩa, $e_1.d_1 = 1 \bmod \Phi(n)$. Vì vậy, $e_1 d_1 - 1$ phải là số nguyên nào đó, là bội số của $\Phi(n)$, và có thể tìm được các số nguyên không âm φ và k mà

$e_1 d_1 = c.\Phi(n) = 2^k \varphi$, với φ là số lẻ.

Thủ tục tìm căn bậc hai không tầm thường của 1 mod n:

1. Chọn số nguyên a sao cho (a, n) =1 và 1 < a < n-1.
2. Tìm số nguyên dương j nhỏ nhất thỏa mãn $a^{2^j\varphi} = 1 \bmod n$.

(Vì $2^k\varphi$ là bội số của $\Phi(n)$, nên chắc chắn tồn tại số này).

3. Đặt $b = a^{2^{j-1}\varphi} \bmod n$.
4. Nếu b $\neq$ -1 mod n, thì nó là căn bậc hai không tầm thường của 1.
5. Nếu b = -1 mod n, quay trở lại bước 1.

De Laurentis đã chứng minh rằng với a ngẫu nhiên, 1 < a < n-1:

$$\textbf{Prob}\ ((a^{\varphi} \equiv 1 \bmod n) \vee (\exists j \le k)(a^{2^j\varphi} \equiv -1 \bmod n)) \le 1/2$$

Hay $$\textbf{Prob}\ (\exists j(1 \le j \le k)(a^{2^{j-1}\varphi} \neq \pm 1 \bmod n \wedge a^{2^j\varphi} = 1 \bmod n)) \ge 1/2$$

Do đó nếu ta xây dựng thuật toán xác suất , thử lần lượt với m giá trị ngẫu nhiên a theo tính chất :

$$(\exists j(1 \le j \le k)(a^{2^{j-1}k} \neq \pm 1 \bmod n \wedge a^{2^j\varphi} = 1 \bmod n)).$$

Thuật toán dừng nếu tính chất đó được nghiệm đúng ở một lúc nào đó và cho kết quả $b = a^{2^{j-1}\varphi} \bmod n$. Ngược lại, thuật toán cũng dừng, nhưng không cho kết quả.

Như vậy, thuật toán khi dùng m giá trị ngẫu nhiên a (1 < a < n-1) sẽ cho kết quả với xác suất thành công $\ge 1 - 1/2^m$. Và khi đó, ta tìm được phân tích p, q của n.

Vì vậy, một người trong cuộc có thể bẻ mật mã trong giao thức này với xác suất rất cao, bằng cách sử dụng thông tin mà mình có. Cách tấn công vừa đề cập tới là rất quan trọng, vì nó chỉ ra rằng những thông tin về cặp khóa mã hóa/ giải mã có thể cho phép tìm ra các thừa số của modul n.

→ ***Giải pháp phòng tránh :*** dùng *modul n* khác nhau cho mỗi người tham gia.

4/. Dùng khóa công khai nhỏ.

Ta chỉ ra rằng nếu tất cả các khóa công khai bằng e, thì Oscar có thể khôi phục được m nếu k ≥ e, với e nhỏ.

Để giảm thời gian mã hóa, người ta sử dụng số mũ công khai e nhỏ, giá trị e nhỏ nhất có thể là 3.

Giả sử A muốn gửi văn bản mã m tới một số người nhận U_1, U_2,..., U_k. Mỗi người có khóa RSA riêng (n, e_i). Ta giả thiết m nhỏ hơn tất cả mọi n_i.

Bình thường để gửi m, A mã hóa nó bằng cách sử dụng mỗi khóa công khai và gửi bản mã thứ i tới U_i. Oscar có thể chặn bắt và thu được k bản mã đã được truyền trên kênh là $c_1, c_2,..., c_k$.

Để cho đơn giản, giả sử mọi số mũ công khai đều bằng 3, khi đó Oscar có thể khôi phục được m nếu k ≥ 3. Thật vậy, Oscar nhận được các bản mã c_1, c_2, c_3, với :

$$\begin{cases} c_1 = m^3 \bmod n_1 \\ c_2 = m^3 \bmod n_2 \\ c_3 = m^3 \bmod n_3 \end{cases} \qquad (2.1)$$

Ta giả thiết rằng UCLN(n_i, n_j) = 1 với mọi i ≠ j, vì trong trường hợp ngược lại thì Oscar có thể phân tích được số n_i nào đó.

Khi đó, ta dùng định lý phần dư Trung Quốc tìm được nghiệm của hệ (2.1) là c', với $c' = m^3 \bmod n_1 n_2 n_3$, vì m nhỏ hơn các n_i nên $m^3 < n_1 n_2 n_3$, nên có $c' = m^3$.

Như vậy, Oscar có thể tìm được m bằng cách tính căn bậc 3 của c'.

Một cách tổng quát, nếu tất cả các khóa công khai bằng e, thì Oscar có thể khôi phục được m nếu k ≥ e. Cách tấn công này chỉ có thể sử dụng khi e nhỏ.

Giá trị thường dùng hiện nay là 65537 vì được xem là đủ lớn và cũng không quá lớn ảnh hưởng tới việc thực hiện hàm mũ.

→ Giải pháp phòng tránh:

Chọn khóa công khai là những số nguyên lớn, có kích cỡ lớn gần như bản thân số n.

5/. Dùng khóa bí mật nhỏ

Để giảm thời gian giải mã, mọi người có thể sử dụng giá trị d nhỏ thay cho d ngẫu nhiên. Bởi vì phép tính modul tốn thời gian tuyến tính theo $\log_2 d$, với d nhỏ có thể tăng tốc độ giải mã. Không may, cách tấn công của M. Wiener đã chỉ ra rằng, với d nhỏ sẽ dẫn đến việc phá hoàn toàn hệ mật.

Định lý M. Wiener :

Giả sử n = p.q với q<p<2q và d< $n^{1/4}/3$.

Cho (n, e) với ed =1 mod $\Phi(n)$, thì Oscar có thể tính được d một cách hiệu quả.

Dan Boneh chỉ ra rằng với d < $n^{0,292}$, thì có thể tính được d có hiệu quả từ (n, e).

Kết quả này chứng tỏ cận của Wiener là chưa chặt.

→ ***Giải pháp phòng tránh:*** Chọn khóa bí mật là những số nguyên lớn, có kích cỡ lớn gần như bản thân số n.

6/. Dùng các tham số p-1 và q-1 có các ước nguyên tố nhỏ

Khi xây dựng sơ đồ chữ ký RSA, nếu ta bất cẩn trong việc chọn các tham số p và q để p-1 hoặc q-1 có các ước nguyên tố nhỏ, thì sơ đồ chữ ký trở nên mất an toàn. Khi p-1 hoặc q-1 có các ước nguyên tố nhỏ thì ta có thể dùng thuật toán của Pollar đưa ra vào năm 1974 phân tích được n một cách hiệu quả.

Thuật toán Pollar.

Input : n và cận b.

output: trả lời

- Thành công và đưa ra thừa số của n.
- Không thành công.

Method:

B1: a:=2;

B2: For j:=2 to b do a:= a^j mod n

B3: d:= UCLN(a-1, n);

B4: If 1<d<n then

Write('Thành công, các thừa số của n là: ', d, n/d);

else

write('Không thành công');

Giả sử p là một ước nguyên tố của n, và p-1 có phân tích ra các mũ nguyên tố sau:

$$p-1=\prod_{i=1}^{n} p_i^{\alpha_i}$$

Đặt q = max($p_i^{\alpha_i}$, i=1,..., n). Nếu q $\leq$ b, khi đó (p-1) | b !

Ở cuối bước 2 ta có: a = $2^{b!} \bmod n$, nên $a = 2^{b!} \bmod p$ vì p | n.

Theo định lý Fermat, ta có: $2^{p-1} \bmod p$. Vì (p-1) | b nên a = 1 mod p.

Vì vậy ở bước 4 ta có p | (a-1) và p | n, do đó p | d = UCLN(a-1, n).

Số nguyên d sẽ là ước không tầm thường của n trừ khi a = 1 ở bước 3.

Ví dụ:

Giả sử n = 15770708441. Nếu áp dụng thuật toán p-1 với b = 180 thì sẽ thấy rằng a = 11620221425 ở bước 3, còn d = 235979 và n/d = 115979.

Trên thực tế, phân tích n thành các thừa số nguyên tố là:

15770708441 = 135979 x 115979

Trong trường hợp này, phép phân tích sẽ thành công do:

p-1 = 135979-1 = 135978 = 2 x3 x131 x173

Nếu lấy b $\geq$ 173 thì chắc chắn rằng 135978 | b!

Trong thuật toán có (b-1) lũy thừa theo modul, mỗi lũy thừa cần nhiều nhất là $2\log_2 b$ phép nhân modul, dùng thuật toán bình phương và nhân. Việc tính UCLN có thể được thực hiện trong thời gian $O((\log_2 n)^3)$ bằng thuật toán Euclide. Bởi vậy, độ phức tạp của thuật toán là $O(b\log_2 b(\log_2 n)^2 + (\log_2 n)^3)$.

Nếu b là $O((\log_2 n)^i)$, với một số nguyên i xác định nào đó, thì thuật toán thực sự là thuật toán thời gian đa thức. Tuy nhiên, vối phép chọn b như vậy, xác suất thành công sẽ rất nhỏ.

Mặt khác, nếu tăng kích thước của b lên thật lớn, chẳng hạn tới $n^{1/2}$, thì thuật toán sẽ thành công, nhưng khi đó nó sẽ không thực hiện nhanh hơn phép chia thử. Như vậy, điểm bất lợi của thuật toán này là nó yêu cầu n phải có ước nguyên tố, sao cho p-1 chỉ có các thừa số nguyên tố bé.

$\rightarrow$ ***Giải pháp phòng tránh:*** Các số p-1 và q-1 phải có ước nguyên tố lớn.

2.1.2.2. Tấn công dạng 2: Giả mạo chữ ký (không tính trực tiếp khóa bí mật)

1/. Ký trước, Mã hóa sau.

Người gửi G gửi tài liệu **x** cùng chữ ký **y** đến người nhận N, G ký trước vào **x** bằng chữ ký **y =** $Sig_G(x)$**,** sau đó mã hóa x và y nhận được $Z = e_G(x, y)$. G gửi **z** cho N. H lấy trộm được thông tin trên được truyền từ G đến N. Để tấn công x, H sẽ tìm cách giải mã thông tin lấy được. Để tấn công vào chữ ký thay bằng chữ ký (giải mạo), H tìm cách giải mã Z, mới nhận được y. Sau đó H thay y bằng chữ ký giả mạo y', gửi đến N. Tuy nhiên trường hợp này H phải giải mã trước, sau đó mới có thể giả mạo chữ ký.

→ ***Giải pháp phòng tránh:*** chọn các số lập mã và giải mã là những số nguyên lớn, có kích cỡ lớn gần như bản thân số n.

2/. Mã hóa trước, Ký sau.

Người gửi G gửi tài liệu x cùng chữ ký y đến người nhận N, G mã hóa trước **x** bằng $u = e_G(x)$, sau đó ký vào **u** bằng chữ ký $v = Sig_G(u)$. G gửi **(u, v)** cho N. H lấy trộm được thông tin trên đường truyền từ G đến N. Để tấn công x, H sẽ tìm cách giải mã thông tin lấy được. Để tấn công chữ ký v, H đã sẵn có v', H chỉ việc thay v bằng v'. H thay chữ ký **v** trên **u**, bằng chữ ký (của H) là **v'** = $Sig_H(u)$, gửi **(u, v')** đến N.

Khi nhận được **v'**, N kiểm thử thấy sai, gửi phản hồi lại G. G có thể chứng minh chữ ký đó là giả mạo. G gửi chữ ký đúng **v** cho N, nhưng quá trình truyền tin sẽ bị chậm lại. Như vậy trong trường hợp này, H có thể giả mạo chữ ký mà không cần giải mã.

→ ***Giải pháp phòng tránh:***

Hãy ký trước, sau đó mã hóa cả chữ ký. Chọn các số lập mã và giải mã là những số nguyên lớn, có kích cỡ lớn gần như bản thân số n.

3/. Kẻ tấn công có khả năng kiểm tra các chữ ký khác nhau có phù hợp với một thông điệp có trước hay không.

Đây là kiểu tấn công rất thông dụng trong thực tế nó thường chia làm 3 lớp:

- Kẻ tấn công có chữ ký cho một lớp các thông điệp.
- Kẻ tấn công dành được các chữ ký đúng cho một danh sách các thông điệp trước khi tiến hành hoạt động phá hủy chữ ký, cách tấn công này là non-adaptive (không mang tính phù hợp), bởi vì thông điệp được chọn trước khi bất kỳ một chữ ký nào được gửi đi.
- Kẻ tấn công được phép sử dụng người ký như là một bên đáng tin cậy, kẻ tấn công có thể yêu cầu chữ ký cho các thông điệp, mà các thông điệp này phụ thuộc vào khóa công khai của người ký. Như vậy kẻ tấn công có thể yêu cầu chữ ký của các thông điệp phụ thuộc vào chữ ký và thông điệp dành trước đây và qua đó tính toán được chữ ký.

→ Giải pháp phòng tránh:

Sử dụng các giải pháp phòng tránh đã trình bày với các dạng tấn công chữ ký. Đây là kiểu tấn công của thám mã chuyên nghiệp.

2.2. TẤN CÔNG CHỮ KÝ ELGAMAL

2.2.1. Chữ ký Elgamal

2.2.1.1. Sơ đồ chữ ký

1/. Sơ đồ (Elgamal đề xuất năm 1985)

**** Tạo cặp khóa (bí mật, công khai)* (a, h):**

Chọn số nguyên tố p sao cho bài toán logrit rời rạc trong Z_p là "khó" giải.

Chọn phần tử nguyên thủy g $\in$ Z_p^*. Đặt P = Z_p^*, A = Z_p^* x Z_{p-1}.

Chọn khóa bí mật là a $\in$ Z_p^*. Tính khóa công khai h $\equiv$ g^a mod p.

Định nghĩa tập khóa: K = {(p, g, a, h): h $\equiv$ g^a mod p}.

Các giá trị p, g, h được công khai, phải giữ bí mật a.

**** Ký số:*** Dùng 2 khóa ký: khóa a và khóa ngẫu nhiên bí mật r $\in$ Z_{p-1}^*.

(Vì r $\in$ Z_{p-1}^*, nên nguyên tố cùng p -1, do đó tồn tại r^{-1} mod (p-1)).

Chữ ký trên x $\in$ P là y = $Sig_k(x,r)$ = (γ, δ), y $\in$ A (E1)

Trong đó γ $\in$ Z_p^*, δ $\in$ Z_{p-1}:

γ = g^r mod p và δ = (x – a*γ)* r^{-1} mod (p-1)

**** Kiểm tra chữ ký:***

$Ver_k(x,\gamma,\delta)$ = đúng $\Leftrightarrow$ $h^\gamma * \gamma^\delta \equiv g^x \bmod p$. (E2)

2/. Chú ý: Nếu chữ ký được tính đúng, kiểm thử sẽ thành công vì

$h^\gamma * \gamma^\delta \equiv g^{a\gamma} * g^{r*\delta} \bmod p \equiv g^{(a\gamma + r*\delta)} \bmod p \equiv g^x \bmod p$.

Do δ = (x – a * γ) * r^{-1} mod (p-1) nên (a * γ + r * δ) $\equiv$ x mod (p-1).

- Chữ ký Elgamal thuộc loại chữ ký đi kèm thông điệp. Tức là người gửi chuyển ***"chữ ký"***, phải gửi kèm cả thông điệp đã được ***"ký"*** bởi ***"chữ ký"*** này. Ngược lại, người nhận sẽ không có được thông điệp gốc.

2.2.1.2. Ví dụ

Chữ ký Elgamal trên dữ liệu x = 112.

**** Tạo cặp khóa (bí mật, công khai)*** **(a, h):**

Chọn số nguyên tố p = 463. Đặt P = Z_p^*, A = $Z_p^* \text{ x } Z_{p-1}$.

Chọn phần tử nguyên thủy g = 2 $\in$ Z_p^*.

Chọn khóa bí mật là a = 211 $\in$ Z_p^*.

Tính khóa công khai h $\equiv$ g^a mod p = 2^{211} mod 463 = 249.

Định nghĩa tập khóa: K = {(p, g, a, h): h $\equiv$ g^a mod p}.

Các giá trị p, g, h được công khai, phải giữ bí mật a.

**** Ký số:*** Chọn ngẫu nhiên bí mật r = 235 $\in$ Z_{p-1}^*. Khóa ký là (a, r).

Vì r $\in$ Z_{p-1}^*, nên nguyên tố cùng p-1, do đó tồn tại r^{-1} mod (p-1). Cụ thể:

UCLN(r, p-1) = UCLN(235, 462) = 1, nên r^{-1} mod (p-1) = 235^{-1} mod 462 = 289.

Chữ ký trên dữ liệu x = 112 là (γ, δ) = (16, 108), trong đó:

γ = g^r mod p = 2^{235} mod 463 = 16

δ = (x – a*γ)* r^{-1} mod (p-1) = (112 – 211 * 16) * 289 mod 462 = 108

**** Kiểm tra chữ ký:*** $Ver_k(x, \gamma, \delta)$ = đúng $\Leftrightarrow$ $h^\gamma * \gamma^\delta \equiv g^x \bmod p$.

$h^\gamma * \gamma^\delta = 249^{16} * 16^{108} \bmod 463 = 132$

$g^x \bmod p = 2^{112} \bmod 463 = 132$.

Hai giá trị đó bằng nhau, như vậy chữ ký là đúng.

2.2.2. Các dạng tấn công vào chữ ký Elgamal

2.2.2.1. Xác định khóa (tìm cách xác định khóa bí mật)

1/. Số ngẫu nhiên r bị lộ:

Nếu r bị lộ, thám mã sẽ tính được khóa mật a = (x – r δ) $\gamma^{-1} \bmod(p-1)$.

→ ***Giải pháp phòng tránh:*** Cần thận trọng trong việc sử dụng số ngẫu nhiên k, không để lộ số k được dùng

2/. Dùng r cho hai lần ký khác nhau:

Giả sử dùng r cho hai lần ký trên x_1 và x_2.

(γ, δ_1) là chữ ký trên x_1, (γ, δ_2) là chữ ký trên x_2,

Khi đó thám mã có thể tính a như sau:

$\beta^{\gamma} * \gamma^{\delta_1} \equiv \alpha^{x_1} (\bmod p)$. $\beta^{\gamma} * \gamma^{\delta_2} \equiv \alpha^{x_2} (\bmod p)$

Do đó ta có $\alpha^{x_1 - x_2} \equiv \gamma^{\delta_1 - \delta_2} (\bmod p)$

Đặt $\gamma = \alpha^r$, ta có $\alpha^{x_1 - x_2} \equiv \gamma^{k*(\delta_1 - \delta_2)} (\bmod p)$

tương đương với $x_1 - x_2 \equiv r(\delta_1 - \delta_2) \bmod(p-1)$ (1)

Đặt d = ($\delta_1 - \delta_2$, p-1). Khi đó d | (p-1), d | ($\delta_1 - \delta_2$) $\Rightarrow$ d | ($x_1 - x_2$).

$$x' = \frac{x_1 - x_2}{d}$$

$$\delta' = \frac{\delta_1 - \delta_2}{d}$$

$$p' = \frac{p-1}{d}$$

Khi đó đồng dư thức (1) trở thành: $x' \equiv r * \delta' (\bmod p')$

Vì (δ', p') = 1 nên tính ε = (δ'-1) mod p' và tính r = x'*ε mod p'

$\Rightarrow$ r = x'*ε + i*p' mod (p-1), với i là giá trị nào đó, 0 ≤ i ≤ d-1.

Thử với giá trị nào đó, ta tìm được r (điều kiện thử để xác định r là $\gamma = \alpha^r \bmod p$).

Tiếp theo sẽ tính được a như trường hợp 1.

→ ***Giải pháp phòng tránh:*** mỗi lần ký sử dụng một số k khác nhau.

3/. Khóa mật a quá nhỏ.

Nếu khóa mật a quá nhỏ, thì bằng phương pháp dò tìm đơn giản, người ta có thể tính được nó.

→ ***Giải pháp phòng tránh:*** chọn khóa bí mật a là những số nguyên lớn, có kích cỡ lớn gần như bản thân số n.

4/. Số ngẫu nhiên r quá nhỏ.

Tương tự như đối với khóa mật a, số ngẫu nhiên r cũng phải bí mật. Trong trường hợp các tham số này quá nhỏ, thì hiển nhiên bằng phương pháp dò tìm đơn giản người ta cũng có thể tìm được chúng.

Khi đó sơ đồ chữ ký sẽ mất an toàn. Nếu r bị lộ, thám mã sẽ tính được khóa mật $a = (x - r\,\delta)\ \gamma^{-1} \bmod (p-1)$.

→ ***Giải pháp phòng tránh:*** chọn số ngẫu nhiên r là những số nguyên lớn, có kích cỡ lớn gần như bản thân số n.

2.2.2.2. Giả mạo chữ ký (không tính trực tiếp khóa bí mật)

1/. Trường hợp 1: Giả mạo chữ ký không cùng với tài liệu được ký.

+ H cố gắng giả mạo chữ ký trên x, mà không biết khóa bí mật a.

Như vậy, H phải tính được γ và δ.

* Nếu chọn trước γ, H phải tính δ qua đẳng thức $h^{\gamma} * \gamma^{\delta} \equiv g^{x} \bmod p$ (E2)

Tức là $\gamma^{\delta} \equiv g^{x} h^{-\gamma} \bmod p$ hay $\delta \equiv \log_{\gamma} g^{x} h^{-\gamma} \bmod p$.

* Nếu chọn trước δ, H phải tính γ qua phương trình: $h^{\gamma} * \gamma^{\delta} \equiv g^{x} \bmod p$.

Hiện nay chưa có cách hữu hiệu 2 trường hợp trên, nhưng phỏng đoán là khó hơn bài toán logarit rời rạc.

Có thể có cách tính γ, δ đồng thời với (γ, δ) là chữ ký? Chưa có trả lời rõ!

* Nếu chọn trước γ, δ, sau đó tính x, H phải đối dấu với bài toán logarit rời rạc.

Ta có $h^{\gamma} * \gamma^{\delta} \equiv g^{x} \bmod p$ (E2).

Như vậy $x \equiv \log_{g} g^{x} \equiv \log_{g} h^{\gamma} * \gamma^{\delta}$

2/. Trường hợp 2: Giả mạo chữ ký cùng với tài liệu được ký.

H có thể ký trên tài liệu ngẫu nhiên bằng cách chọn trước đồng thời x, γ, δ.

Cách 1

* Chọn x, γ, δ thỏa mãn điều kiện kiểm thử như sau:

Chọn các số nguyên i, j sao cho $0 \le i, j \le p-2$, (j, p-1) = 1 và tính:

$$\gamma = g^i h^j \bmod p, \qquad \delta = -\gamma j^{-1} \bmod(p-1), \qquad x = -\gamma i j^{-1} \bmod(p-1).$$

Trong đó j^{-1} được tính theo mod (p-1) (nghĩa là j nguyên tố với p-1).

* Chứng minh (γ, δ) là chữ ký trên x, bằng cách kiểm tra điều kiện kiểm thử:

$$h^\gamma \gamma^\delta \equiv h^\gamma (g^i h^j)^{-\gamma . j-1} \bmod p \equiv h^\gamma g^{-i.\gamma . j-1} h^{-\gamma} \bmod p \equiv g^x \bmod p$$

Cách 2

* Nếu (γ, δ) là chữ ký trên tài liệu x có từ trước, thì có thể giả mạo chữ ký trên tài liệu x' khác.

+ Chọn số ngẫu nhiên k, i, j thỏa mãn $0 \le k, i, j \le p-2$, (k γ - j δ, p-1) = 1 và tính:

$$\lambda = \gamma^k g^i h^j \bmod p, \qquad \mu = \delta\lambda(k\gamma - j\delta)^{-1} \bmod(p-1),$$

$$x' = \lambda(kx + i\delta)(k\gamma - j\delta)^{-1} \bmod(p-1)$$

* (λ, μ) là chữ ký trên x', và thỏa mãn điều kiện kiểm thử:

$$h^\lambda \lambda^\mu \equiv g^{x'} \bmod p$$

Chú ý

Cả hai cách giả mạo nói trên đều cho chữ ký đúng trên tài liệu tương đương, nhưng đó không phải là tài liệu được chọn theo ý của người giả mạo. Tài liệu đó đều được tính sau khi tính chữ ký, vì vậy giả mạo loại này trong thực tế cũng không có ý nghĩa nhiều.

2.3. TẤN CÔNG CHỮ KÝ DSS

2.3.1. Chữ ký DSS

2.3.1.1. Sơ đồ chữ ký DSS

1/. Giới thiệu chuẩn chữ ký số DSS

Chuẩn chữ ký số (DSS: Digital Signature Standard) được đề xuất năm 1991, là cải biên của sơ đồ chữ ký Elgamal, và được chấp nhận là chuẩn vào năm 1994 để dùng trong một số lĩnh vực giao dịch ở USA.

Thông thường tài liệu số được mã hóa và giải mã 1lần. Nhưng chữ ký lại liên quan đến ***pháp luật, chữ ký*** có thể phải kiểm thử sau nhiều năm đã ký. Do đó ***chữ ký*** phải được bảo vệ cẩn thận. Như vậy số nguyên tố p phải đủ lớn (chẳng hạn dài cỡ 512 bit) để bảo đảm an toàn, nhiều người đề nghị nó phải dài 1024 bit.
Tuy nhiên, độ dài chữ ký theo sơ đồ Elgamal là gấp đối số bít của p, do đó nếu p dài 512 bít thì độ dài chữ ký là 1024 bit.

Trong ứng dụng dùng thẻ thông minh (Smart card) lại mong muốn có chữ ký ngắn, nên giải pháp sửa đổi là một mặt dùng p với độ dài từ 512 bit đến 1024 bit (bội của 64), mặt khác trong chữ ký (γ, δ), các số γ, δ có độ dài biểu diễn ngắn, ví dụ 160 bit. Khi đó chữ ký là 320 bit.

Điều này được thực hiện bằng cách dùng nhóm con cyclic Z_q^* của Z_p^* thay cho Z_p^*, do đó mọi tính toán được thực hiện trong Z_p^*, nhưng thành phần chữ ký lại thuộc Z_q^*.

Thay đổi công thức tính δ trong sơ đồ chữ ký Elgamal thành $\delta = (x + a * \gamma) r^{-1} \bmod q$.
Điều kiện kiểm thử là $h^{\gamma} \gamma^{\delta} \equiv g^{x} \bmod p$ được sửa thành

$a^{x*\delta^{-1}} * \beta^{\gamma*\delta^{-1}} \equiv \gamma (\bmod p)$.

Nếu (x + g * γ, p-1) = 1 thì $\delta^{-1} \bmod p$ tồn tại.

2/. Sơ đồ chữ ký DSS

Sơ đồ

* Tạo cặp khóa (bí mật, công khai) (a, h):

+ Chọn số nguyên tố p sao cho bài toán logarit rời rạc trong Z_p là "khó" giải. Chọn q là ước nguyên tố của p-1. Tức là p-1 = t * q hay p = t * q + 1.
(Số nguyên tố p cỡ 512 bit, q cỡ 160 bit).

+ Chọn g $\in Z_p^*$ là căn bậc q của 1 mod p, (g là phần tử sinh của Z_p^*).

Tính $\alpha = g^t$, chọn khóa bí mật a $\in Z_p^*$, tính khóa công khai $h \equiv \alpha^a \bmod p$.

+ Đặt P = Z_q^*, A = Z_q^* x Z_q^*, K = {(p, q, α, a, h)/ a $\in Z_p^*$, $h \equiv \alpha^a \bmod p$ }.

+ Với mỗi khóa (p, q, α, a, h), k' = a bí mật, k" = (p, q, α, h) công khai.

* Ký số: Dùng 2 khóa ký: khóa a và khóa ngẫu nhiên bí mật r $\in Z_q^*$.

Chữ ký trên x $\in Z_p^*$ là $Sig_{k'}(x,r) = (\gamma, \delta)$, trong đó

$\gamma = (\alpha^r \bmod p) \bmod q$, $\delta = (x + a * \gamma) * r^{-1} \bmod q$.

(Chú ý r $\in Z_q^*$, để bảo đảm tồn tại r^{-1} mod q).

* Kiểm tra chữ ký: Với $e_1 = x * \delta^{-1} \bmod q$, $e_2 = \gamma * \delta^{-1} \bmod q$.

$Ver_{k"}(x, \gamma, \delta) = đúng \Leftrightarrow (\alpha^{e_1} * h^{e_2} \bmod p) \bmod q = \gamma$

2.3.1.2. Ví dụ

**** Tạo cặp khóa (bí mật, công khai)* (a, b):**

Chọn p = 7649, q = 239 là ước nguyên tố của p – 1, t = 32.

Tức là p-1 = t * q hay p = t * q + 1 = 32 * q + 1 = 32*239 + 1 = 7649.

Chọn g = 3 $\in Z_{7649}$ là phần tử sinh. $\alpha = g^t \bmod p = 3^{32} \bmod 7649 = 7098$.

Chọn khóa mật a = 85, khóa công khai $h = \alpha^a \bmod p = 7098^{85} \bmod 7649 = 5387$.

**** Ký số:*** Dùng 2 khóa ký: a và khóa ngẫu nhiên r = 58 $\in Z_q^*$, $r^{-1} \bmod q = 136$.

+ Chữ ký trên x = 1246 là $Sig_{k'}(x,r) = (\gamma, \delta)$ = (115, 87), trong đó

$\gamma = (\alpha^r \bmod p) \bmod q$ = ($7098^{58} \bmod 7649$) mod 239 = 593 mod 239 = 115.

$\delta = (x + a * \gamma) * r^{-1} \bmod q$ = (1246 + 85 * 115) * 136 mod 239 = 87.

**** Kiểm tra chữ ký:*** (γ, δ) = (115, 87) là chữ ký đúng trên x = 1246.

$e_1 = x * \delta^{-1} \bmod q$ = 1246 * 11 mod q = 83, $e_2 = \gamma * \delta^{-1} \bmod q$ = 115 * 11 mod q = 70.

Điều kiện kiểm thử đúng ? $(\alpha^{e_1} * h^{e_2} \bmod p) \bmod q = \gamma$, với δ^{-1} = 11.

$$(7098^{83} * 5387^{70} \bmod 7649) \bmod 239 = 593 \bmod 239 = 115 = \gamma .$$

Chú ý:

1). Liên quan tới các tính toán cụ thể trong sơ đồ:

+ Chú ý rằng phải có $\delta \neq 0 \pmod q$ để bảo đảm có δ^{-1} mod q trong điều kiện kiểm thử (tương đương UCLN(δ, p-1) = 1). Vì vậy nếu chọn r mà không được điều kiện trên, thì phải chọn r khác để có $\delta \neq 0 \pmod q$.

Tuy nhiên khả năng $\delta \equiv 0 \pmod q$ là 2^{-160}, điều đó hầu như không xảy ra.

+ Một chú ý là thay vì tính p trước rồi mới tính q, ta sẽ tính q trước rồi tìm p.

2). Liên quan chung tới DSS (1991):

+ Độ dài cố định của p là 512 bit. Nhiều người muốn p có thể thay đổi lớn hơn. Vì thế NIST sửa đổi là p có độ dài thay đổi, là bội của 64: từ 512 đến 1024 bit.

+ Nếu dùng chữ ký RSA với thành phần kiểm thử chữ ký nhỏ, thì việc kiểm thử nhanh hơn việc ký. Đối với DSS, ngược lại, việc ký nhanh hơn kiểm thử.

Điều này dẫn đến vấn đề:

Một tài liệu chỉ được ký một lần, nhưng nó lại được kiểm thử nhiều lần, nên người ta muốn thuật toán kiểm thử nhanh hơn.

Máy tính ký và kiểm thử như thế nào? Nhiều ứng dụng dùng thẻ thông minh với khả năng có hạn, kết nối với 1 máy tính mạnh hơn, vì vậy nên xây dựng sơ đồ chữ ký liên quan đến thẻ.

Nhưng tình huống đặt ra là một thẻ thông minh có thể sinh ra chữ ký và cũng có thể kiểm thử chữ ký, do vậy rất khó kết luận?

NIST trả lời rằng thời gian kiểm thử và sinh chữ ký, cái nào nhanh hơn không quan trọng, miễn là đủ nhanh.

Chữ ký DSS thuộc loại chữ ký đi kèm thông điệp. Đó là cải tiến của chữ ký Elgamal. Các dạng tấn công vào DSS tương tự như với chữ ký Elgamal.

KẾT LUẬN

Cùng với sự phát triển chung của loài người, công nghệ thông tin đã và đang là một trong những lĩnh vực đem lại nhiều lợi ích cho xã hội, và sẽ trở thành yếu tố không thể thiếu trong nền kinh tế hội nhập và toàn cầu hóa của xã hội loài người.

Chính vì vậy an toàn và bảo mật thông tin sẽ là một trong những yếu tố rất quan trọng, đảm bảo an toàn cho việc áp dụng nhiều ứng dụng trong thực tiễn, cho các giao dịch điện tử. Các giải pháp về chính quyền điện tử, về thương mại điện tử sẽ không bao giờ thực hiện được nếu không có cơ sở an toàn thông tin vững chắc

Một trong những nhiệm vụ của bảo đảm an toàn thông tin là bảo vệ chữ ký (công cụ xác thực quan trọng), vì vậy đề tài đã nghiên cứu về chữ ký số. Cụ thể là nghiên cứu khả năng tấn công chữ ký, từ đó đưa ra các giải pháp khắc phục, tránh các sự cố giả mạo chữ ký.

Kết quả chính của Đồ án tốt nghiệp là tìm hiểu và nghiên cứu qua tài liệu để hệ thống lại các vẫn đề sau:

1/. Trình bày một số khái niệm cơ bản về mã hóa dữ liệu, về chữ ký số.

2/. Trình bày một số khả năng tấn công chữ ký số của thám mã và giải pháp phòng tránh.

Để hoàn thành được luận văn, em đã nhận được sự chỉ bảo, hướng dẫn tận tình của thầy giáo PGS.TS. Trịnh Nhật Tiến. Tuy nhiên, luận văn không tránh khỏi thiếu sót, rất mong sự góp ý của các Thầy, Cô giáo và các bạn.

BẢNG CHỮ VIẾT TẮT

RSA (Rivest-Shamir-Adleman)	
ELGAMAL (T. ElGamal)	
DSS (Digital Signature Standard)	
DES (Data Encryption Standard)	
USA (United States of America)	
NIST	Viện các tiêu chuẩn và công nghệ quốc gia
UCLN	Ước chung lớn nhất
BCNN	Bội chung nhỏ nhất
ATTT	An toàn thông tin
TT	Trung tâm phân phối khóa
Smart card	Thẻ thông minh
PT	Độ phức tạp

TÀI LIỆU THAM KHẢO

1. Phan Đình Diệu. Lý thuyết mật mã và An toàn thông tin, 2004.

2. TS. Nguyễn Ngọc Cương (1999), Bài giảng An toàn hệ thống thông tin.

3. PGS.TS. Trịnh Nhật Tiến. Bài giảng môn An toàn dữ liệu, 2005.

4. Phạm Huy Điển, Hà Duy Khoái (2003), Mã hóa thông tin: Cơ sở toán học và ứng dụng, nhà xuất bản Đại Học Quốc Gia Hà Nội.

5. Jalal Feghhi, Jalil Feghhi, Peter Williams. Digital Certificates.
Applied Internet Security, 1999.

6. S. Castano, M. Fugina, G. Martella, P. Samarati. Database Security, 1994.

7. Danley Harrisson. “An Introduction to Steganography”, 2002.