ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

LÊ CÔNG TUẤN ANH

CÁC PHƯƠNG PHÁP TẤN CÔNG CHỮ KÝ SỐ: RSA,ELGAMAL,DSS

LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN

Hà Nội - 2016

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

LÊ CÔNG TUẤN ANH

CÁC PHƯƠNG PHÁP TẤN CÔNG CHỮ KÝ SỐ: RSA,ELGAMAL,DSS

Ngành: Công nghệ Thông tin
Chuyên ngành: Kỹ thuật phần mềm
Mã số: 60480103

LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN

NGƯỜI HƯỚNG DẪN KHOA HỌC: PGS.TS TRỊNH NHẬT TIẾN

Hà Nội - 2016

LỜI CẢM ƠN

Tôi xin được gửi lời cảm ơn sâu sắc tới ***PGS.TS Trịnh Nhật Tiến***, Trường Đại học Công nghệ - Đại học Quốc gia Hà Nội, người thầy đã dành nhiều thời gian tận tình chỉ bảo, hướng dẫn, giúp đỡ tôi trong suốt quá trình tìm hiểu và nghiên cứu.Thầy cũng là người định hướng và đưa ra nhiều góp ý quý báu trong suốt quá trình tôi thực hiện luận văn.

Tôi xin chân thành cảm ơn các thầy, cô ở khoa Công nghệ thông tin – Trường Đại học Công nghệ - ĐHQGHN đã cung cấp cho tôi những kiến thức và tạo cho tôi những điều kiện thuận lợi trong suốt quá trình tôi học tập tại trường.

Tôi xin cảm ơn gia đình, người thân và bạn bè luôn động viên và tạo mọi điều kiện tốt nhất cho tôi.

Tôi xin chân thành cảm ơn!

Hà Nội, tháng 10 năm 2016
Họ và tên

Lê Công Tuấn Anh

LỜI CAM ĐOAN

Tôi xin cam đoan đây là đề tài do tôi nghiên cứu, thực hiện dưới sự hướng dẫn của ***PGS.TS Trịnh Nhật Tiến***.

Trong toàn bộ nội dung nghiên cứu của luận văn, các vấn đề được trình bày đều là những tìm hiểu và nghiên cứu của chính cá nhân tôi hoặc là được trích dẫn từ các nguồn tài liệu có ghi tham khảo rõ ràng, hợp pháp.

Hà Nội, tháng 10 năm 2016
Họ và tên

Lê Công Tuấn Anh

MỤC LỤC

DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT

STT	Từ viết tắt	Ý nghĩa
1	BCNN	Bội chung nhỏ nhất
2	CA	Certificate Authority
3	DSS	Digital Signature Standard
4	NIST	National Institute of Standards and Technology
5	PT	Độ phức tạp
6	RSA	Ron Rivest, Adi Shamir, Len Adleman
7	Sig_k	Thao tác ký số
8	UCLN	Ước chung lớn nhất
9	USA	United States of America
10	Ver_k	Thao tác kiểm tra chữ ký

DANH MỤC CÁC BẢNG

DANH MỤC CÁC HÌNH VẼ, ĐỒ THỊ

MỞ ĐẦU

Ngày nay, chữ ký số được sử dụng trong rất nhiều lĩnh vực, ví dụ: trong kinh tế với các cuộc trao đổi 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 có phạm vi rộng lớn.

Một vài chữ ký số đã được xây dựng và phát triển là: *RSA,ELGAMAL,DSS*. Mặc dù bản thân chúng vẫn còn tồn tại nhiều hạn chế như là về kích thước chữ ký, khả năng chống giả mạo chưa cao, tuy nhiên, những khả năng mà nó đem lại cho chúng ta là rất hữu ích.

Khi áp dụng chữ ký số, vấn đề an ninh luôn được chúng ta quan tâm hàng đầu. Một chữ ký số chỉ thực sự được áp dụng trong thực tế nếu như nó được chứng minh là không thể hoặc rất khó giả mạo. Mục tiêu của những kẻ tấn công các sơ đồ chữ ký chính là việc giả mạo chữ ký, điều này có nghĩa là 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 vào chữ ký số hết sức đa dạng. Đây cũng chính là vấn đề được nghiên cứu trong luận văn này.

Nội dung của luận văn gồm các chương:

Chương 1. Trình bày một số khái niệm cơ bản

Chương 2. Tìm hiểu các phương pháp tấn công chữ ký số

Chương 3. Xây dựng thư viện tính toán số lớn

Chương 4. Thử nghiệm chương trình tấn công

Chương 1. MỘT SỐ KHÁI NIỆM CƠ BẢN

1.1. Một số khái niệm trong số học

1.1.1. Ước chung lớn nhất và bội chung nhỏ nhất

1/. Khái niệm [1]

Cho hai số nguyên a và b, $b \neq 0$. 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.

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.

2/. Ước chung lớn nhất, bội chung nhỏ nhất [1]

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

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

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

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

Ví dụ: Cho a =12, b =15 thì: gcd(12,15) = 3 và lcm(12,15) = 60.

1.1.2. Quan hệ đồng dư

1/. Khái niệm [1]

Cho các 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: $a \equiv b \pmod m$

Ví dụ: $17 \equiv 5 \pmod 3$ vì chia 17 và 5 cho 3, được cùng số dư là 2.

Nhận xét:

Các mệnh đề sau đây là tương đương:

a). $a \equiv b \pmod m$

b). $m \backslash (a - b)$

c). Tồn tại số nguyên t sao cho $a = b + mt$

2/. Các tính chất [1]

a). Quan hệ "đồng dư" là quan hệ tương đương trong Z:

Với mọi số nguyên dương m ta có:

a ≡ a (mod m) với mọi a∈Z; *(tính chất phản xạ).*

a ≡ b (mod m) thì b ≡ a (mod m); *(tính chất đối xứng).*

a ≡ b (mod m) và b ≡ c (mod m) thì a ≡ c (mod m); *(tính chất bắc cầu).*

b). Tổng hay hiệu các "đồng dư":

(a+b) (mod n) ≡ [(a mod n) + (b mod n)] (mod n)

(a- b) (mod n) ≡ [(a mod n) - (b mod n)] (mod n)

c). Tích các "đồng dư":

(a*b) (mod n) ≡ [(a mod n) * (b mod n)] (mod n)

1.1.3. Số nguyên tố

1/. Khái niệm: Số nguyên tố là số tự nhiên lớn hơn 1, chỉ chia hết cho 1 và chính nó.

Ví dụ: 2, 3, 5, 7, 11, 17,…, 1299827,… là các số nguyên tố.

Số 2 là số nguyên tố chẵn duy nhất và đồng thời là số nguyên tố nhỏ nhất.

Người ta đã chứng minh rằng số lượng các số nguyên tố là vô hạn. Số nguyên tố có vai trò vô cùng quan trọng trong lý thuyết mật mã. Các nhà khoa học đang ngày đêm tìm kiếm các số nguyên tố lớn để ứng dụng vào trong mật mã.

Bảng 10 số nguyên tố lớn nhất được tìm ra cho tới thời điểm này (tháng 10/2016):

Bảng 1.1 Bảng 10 số nguyên tố lớn nhất [13]

Rank	Prime	Digits	Who	When	Reference
1	$2^{74207281}-1$	22338618	G14	2016	Mersenne 49??
2	$2^{57885161}-1$	17425170	G13	2013	Mersenne 48??
3	$2^{43112609}-1$	12978189	G10	2008	Mersenne 47??
4	$2^{42643801}-1$	12837064	G12	2009	Mersenne 46??
5	$2^{37156667}-1$	11185272	G11	2008	Mersenne 45??
6	$2^{32582657}-1$	9808358	G9	2006	Mersenne 44??
7	$2^{30402457}-1$	9152052	G9	2005	Mersenne 43??
8	$2^{25964951}-1$	7816230	G8	2005	Mersenne 42??
9	$2^{24036583}-1$	7235733	G7	2004	Mersenne 41??
10	$2^{20996011}-1$	6320430	G6	2003	Mersenne 40??

Bảng 10 số nguyên tố sinh đôi lớn nhất được tìm thấy (tháng 10/2016):

Bảng 1.2 Bảng 10 số nguyên tố sinh đôi lớn nhất [13]

Rank	Prime	Digits	Who	When	Reference
1	$2996863034895 \cdot 2^{1290000}+1$	388342	L2035	2016	Twin (p+2)
2	$2996863034895 \cdot 2^{1290000}-1$	388342	L2035	2016	Twin (p)
3	$3756801695685 \cdot 2^{666669}+1$	200700	L1921	2011	Twin (p+2)
4	$3756801695685 \cdot 2^{666669}-1$	200700	L1921	2011	Twin (p)
5	$65516468355 \cdot 2^{333333}+1$	100355	L923	2009	Twin (p+2)
6	$65516468355 \cdot 2^{333333}-1$	100355	L923	2009	Twin (p)
7	$70965694293 \cdot 2^{200006}+1$	60219	L95	2016	Twin (p+2)
8	$70965694293 \cdot 2^{200006}-1$	60219	L95	2016	Twin (p)
9	$66444866235 \cdot 2^{200003}+1$	60218	L95	2016	Twin (p+2)
10	$66444866235 \cdot 2^{200003}-1$	60218	L95	2016	Twin (p)

Trong mật mã, người ta thường sử dụng các số nguyên tố có vài trăm chữ số trở lên.

Hai số m và n được gọi là nguyên tố cùng nhau, nếu ước số chung lớn nhất của chúng bằng 1. Ký hiệu: *gcd (m, n) = 1.*

Ví dụ: 9 và 14 là hai số nguyên tố cùng nhau. [6]

2/. Các định lý về số nguyên tố

a). Định lý về số nguyên dương >1

Mọi số nguyên dương n >1 đều có thể biểu diễn được *duy nhất* dưới dạng:

$n = P_1^{n_1}.P_2^{n_2}...P_k^{n_k}$, trong đó: k,n_i (*i =1,2,..,k*) là các số tự nhiên, P_i là các số nguyên tố, từng đôi một khác nhau. [1]

b). Định lý Mersenne [1]

Cho $p = 2^k - 1$, nếu p là số nguyên tố, thì k phải là số nguyên tố.

c). Định lý Fermat và số nguyên tố Fermat [6]

- Định lý: Nếu p là số nguyên tố, a là số nguyên thì $a^p \equiv a \pmod{p}$.

Một cách phát biểu khác của định lý như sau: Nếu p là số nguyên tố và a là số nguyên tố cùng nhau với p thì: $a^{p-1} \equiv 1 \pmod{p}$.

Ví dụ: $4^7 \equiv 4 \pmod 7$; $4^{7-1} \equiv 1 \pmod 7$.

- Số nguyên tố Fermat: là một số nguyên dương có dạng: $F_n = 2^{2^n} + 1$

Rất nhiều số Fermat là số nguyên tố, cho nên có một thời gian người ta cho rằng tất cả các số có dạng đó đều là số nguyên tố.

Với n là số không âm, các số Fermat đầu tiên bao gồm:

$F_0 = 2^1 + 1 = 3$ $F_1 = 2^2 + 1 = 5$

$F_2 = 2^4 + 1 = 17$ $F_3 = 2^8 + 1 = 257$

$F_4 = 2^{16} + 1 = 65.537$ $F_5 = 2^{32} + 1 = 4.294.967.297$

$F_6 = 2^{64} + 1 = 18.446.744.073.709.551.617$

$F_7 = 2^{128} + 1 = 340.282.366.920.938.463.463.374.607.431.768.211.457$

d). Hàm Euler [1]

Cho số nguyên dương n, *số lượng* các số nguyên dương bé hơn n và *nguyên tố cùng nhau* với n được ký hiệu ϕ(n) và gọi là hàm *Euler*.

Nhận xét: Nếu p là số nguyên tố, thì ϕ(p) = p-1.

Ví dụ: Tập các số nguyên không âm nhỏ hơn 7 là Z_7 ={0, 1, 2, 3, 4, 5, 6}.

Do 7 là *số nguyên tố*, nên tập các số nguyên dương nhỏ hơn 7 và *nguyên tố cùng nhau* với 7 là Z_7^* ={1, 2, 3, 4, 5, 6}. Khi đó /Z/=ϕ(p) = p-1= 7-1= 6.

Định lý: Nếu n là tích của hai số nguyên tố n=p.q, thì ϕ(n) =ϕ(p).ϕ(q) = (p-1).(q-1).

1.2. Một số khái niệm trong đại số

1.2.1. Cấu trúc nhóm [1]

1/. Khái niệm: Nhóm là một bộ (G, *), trong đó G ≠ ∅, * là *phép toán hai ngôi* trên G thoả 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: $|G|$.

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.

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.

2/. Nhóm con của nhóm (G,*)

Nhóm con của G là tập $S \subset G$, $S \neq \varnothing$, 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 $x \in S$.

1.2.2. Nhóm Cyclic [1]

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 $g \in G$ mà với mỗi $a \in G$, đều tồn tại số $n \in N$ để $g^n = g*g*\ldots*g = a$. (*g*g*...*g là g*g với n lần*). Khi đó g được gọi là *phần tử sinh* hay *phần tử nguyên thuỷ* của nhóm G.

2/. Cấp của nhóm Cyclic: Cho (G,*) là *nhóm Cyclic* với phần tử sinh g và phần tử trung lập e. Nếu tồn tại số tự nhiên *nhỏ nhất* n mà $g^n = e$, thì G sẽ chỉ gồm có n phần tử khác nhau: e, g, g^2, g^3,...,g^{n-1}. Khi đó, G được gọi là *nhóm Cyclic* hữu hạn *cấp* n. Nếu không tồn tại số tự nhiên n để $g^n = e$, thì G có *cấp* ∞.

Ví dụ: $(Z^+, +)$ gồm các số nguyên dương là *Cyclic* với phần tử sinh g = 1, e = 0. Đó là nhóm Cyclic vô hạn, vì không tồn tại số tự nhiên n để $g^n = e$.

3/. Cấp của một phần tử trong Nhóm Cyclic: Phần tử $\alpha \in G$ gọi là có *cấp d*, nếu d là số nguyên dương *nhỏ nhất* sao cho $\alpha^d = e$, trong đó e là phần tử trung lập của G. Như vậy, phần tử α có *cấp 1*, nếu $\alpha = e$.

1.2.3. Nhóm Z_n^*

1/. Tập thặng dư thu gọn theo modulo [1]

* Kí hiệu:

$Z_n = \{0, 1, 2,..., n-1\}$ là tập các số nguyên không âm < n.

Z_n và phép cộng (+) lập thành *nhóm Cyclic* có phần tử sinh là 1 và phần tử trung lập là e = 0.

$(Z_n,+)$ gọi là nhóm cộng, đó là nhóm hữu hạn có cấp n.

* Kí hiệu:

Z_n^* = {x∈Z_n, x là nguyên tố cùng nhau với n}. Tức là x phải ≠ 0.

Z_n^* được gọi là *tập thặng dư thu gọn theo mod n*, có số phần tử là ϕ(n).

Z_n^* với *phép nhân mod n* lập thành một nhóm nhân, phần tử trung lập e = 1.

Tổng quát (Z_n^*, *phép nhân mod n*) không phải là nhóm Cyclic.

Nhóm nhân Z_n^* là Cyclic chỉ khi n có dạng: 2, 4, p^k, hay $2p^k$ với p là số nguyên tố lẻ và k ≥ 1. Còn nếu *n* là số nguyên tố thì Z_n^* là nhóm Cyclic.

2/. Phần tử nghịch đảo đối với phép nhân [1]

a). Định nghĩa: Cho a∈Z_n, nếu tồn tại b∈Z_n sao cho a.b ≡ 1(mod n), ta nói b là *phần tử nghịch đảo* của a trong Z_n và ký hiệu a^{-1}. Một phần tử có phần tử nghịch đảo, gọi là khả nghịch.

b). Định lý: UCLN(a, n) = 1 ⇔ Phần tử a∈Z_n có phần tử nghịch đảo.

c). Thuật toán Euclid tìm phần tử nghịch đảo

Input: a∈Z_n, n

Output: Phần tử nghịch đảo của a.

```
Procedure Nghịch đảo(a,n)
Begin
   g0:=n; g1:=a;
   u0:=1; u1:=0;
   v0:=0; v1:=1;
   i:=1;
   while (gi ≠ 0) do
        begin
          y := gi-1 div gi;
          gi+1 := gi+1 - y.gi;
          ui+1 := ui+1 - y.ui;
          vi+1 := vi+1 - y.vi;
          i := i+1;
        end;
   t := vi+1; if (t > 0) then a-1 := t else a-1:= t+n;
End.
```

3/. Khái niệm logarit rời rạc [1]

Cho *p* là số nguyên tố, *g* là phần tử nguyên thuỷ $\in Z^*_p$ và $\beta \in Z^*_p$

"*Logarit rời rạc*" chính là việc giải phương trình $x = log_g{}^{\beta} \pmod p$ với ẩn x.

Hay phải tìm số x duy nhất sao cho: $g^x \equiv \beta \pmod p$.

4/. Thặng dư bậc hai [5]

a). Định nghĩa: Phần tử $a \in Z^*_n$ được gọi là thặng dư bậc 2 theo modulo n nếu $\exists x \in Z^*_n$: $x^2 \equiv a \bmod n$. Gọi Q_N là tập các thặng dư bậc 2 và Q_N là tập các thặng dư không bậc 2.

b). Định lý (về số lượng thặng dư bậc 2):

- Với p là số nguyên tố, α là phần tử sinh $\in Z^*_p$, khi đó $a = \alpha^i \bmod p$ là thặng dư bậc 2 khi và chỉ khi *i* chẵn. Số lượng thặng dư bậc 2 được tính bằng công thức:

$$|Q_p| = \frac{p-1}{2} = |\overline{Q_p}|.$$

- Với $n = p.q$ trong đó p,q là các số nguyên tố. Khi đó, $a \in Q_N$ khi và chỉ khi $a \in Q_p$ và $a \in Q_q$.

Vậy số lượng thặng dư bậc 2 là:

$$|Q_N| = \frac{1}{4}(p-1).(q-1)$$

số lượng thặng dư không bậc 2 là:

$$|\overline{Q_N}| = \frac{3}{4}(p-1).(q-1).$$

1.3. Độ phức tạp của thuật toán

1.3.1. Khái niệm độ phức tạp của thuật toán [1]

1/. Chi phí của thuật toán

Chi phí phải trả cho một quá trình tính toán gồm chi phí về thời gian và chi phí về bộ nhớ. Gọi A là một thuật toán, e là dữ liệu vào của bài toán đã được mã hoá bằng cách nào đó. Thuật toán A tính trên dữ liệu vào e phải trả một giá nhất định.

Ta ký hiệu: $t_A(e)$ là giá thời gian và $l_A(e)$ là giá bộ nhớ.

2/. Độ phức tạp về bộ nhớ

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

3/. Độ phức tạp về thời gian

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

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 $\exists(n_0,c)$ 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 $O(n^t)$, trong đó t là hằng số.

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

Bảng 1.3 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 có tốc độ rất cao cho nên không cần phải quan tâm nhiều tới thuật toán nhanh, 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.

1.3.2. Phân lớp bài toán theo độ phức tạp [1]

1/. 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 đa thức để giải bài toán B.

Nghĩa là: Bài toán A "khó hơn" bài toán B, hay B "dễ" hơn A, bài toán 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.

2/. 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.

3/. Lớp bài toán P, NP

Ký hiệu:

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

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 ?

4/. 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.

5/. 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 bài toán đầu tiên thuộc lớp này, đó là bài toán "thỏa được" (*Satisfy ability*).

1.3.3. Hàm một phía và hàm cửa sập một phía [1]

1/. Hàm một phía

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

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

2/. Hàm cửa sập một phía

Hàm f(x) được gọi là *hàm cửa sập một phía* nếu tính "*xuôi*" y = f(x) thì "*dễ*", nhưng tính "*ngược*" x = f^{-1} (y) thì lại rất "*khó*". Tuy nhiên, lại có *cửa sập z* sao cho việc 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 việc tính x = f^{-1}(y) là rất "khó", nhưng nếu biết cửa sập p và q, thì việc tính được f^{-1}(y) là khá "dễ".

1.4. Các bài toán quan trọng trong mật mã

Trong phần này, chúng ta sẽ tìm hiểu ba bài toán có vai trò cực kỳ quan trọng trong lý thuyết mật mã, đó là các bài toán: kiểm tra tính nguyên tố của một số nguyên; 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 là rất lớn.

1.4.1. Bài toán kiểm tra số nguyên tố lớn

Cho n là số nguyên dương bất kỳ. Làm thế nào để kiểm tra được n có phải 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. Năm 1975, *Pratt* đã chứng minh nó thuộc lớp *NP* và thuộc lớp *co-NP*$\cap$*NP*, đây là bài toán "khó". Bằng những phương pháp đơn giản như sàng *Eratosthenes*, từ rất sớm người ta đã xây dựng được bảng các số nguyên tố đầu tiên, rồi tiếp tục bằng nhiều phương pháp khác tìm thêm được nhiều số nguyên tố lớn hơn.

Tuy nhiên, chỉ đến giai đoạn hiện nay của lý thuyết mật mã hiện đại thì nhu cầu sử dụng các số 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 chúng ta 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. [4]

1/. Một số ký hiệu toán học [3]

a). Ký hiệu Lagrăng

Ký hiệu L(a,p) được định nghĩa với a là một số nguyên và p là một số nguyên tố lớn hơn 2. Nó nhận ba giá trị 0, 1, -1:

L(a,p) = 0 nếu a chia hết cho p

L(a,p) = 1 nếu $a \in Q_N$ (a là thặng dư bậc 2 modulo p)

L(a,p) = -1 nếu $a \in \overline{Q}_N$ (a không là thặng dư bậc 2 modulo p)

Một phương pháp dễ dàng để tính toán ra L(a,p) là: $L(a,p) = a^{(p-1)/2} \bmod p$

b). Ký hiệu Jacobi

Ký hiệu Jacobi được viết là J(a,n), nó là sự khái quát hóa của ký hiệu Lagrăng, nó định nghĩa cho bất kỳ cặp số nguyên a và n nào. Ký hiệu Jacobi là một chức năng trên tập hợp số thặng dư thấp của ước số n và có thể tính toán theo công thức sau:

- Nếu n là số nguyên tố, thì J(a,n) = 1 nếu a là thặng dư bậc hai modulo n.
- Nếu n là số nguyên tố, thì J(a,n) = -1 nếu a không là thặng dư bậc hai modulo n.

- Nếu n không phải là số nguyên tố thì Jacobi (a,n) sẽ được tính theo công thức sau: $J(a,n) = J(h,p_1) \times J(h,p_2) \times \ldots \times J(h,p_m)$ với $p_1, p_2, \ldots, p_m$ là các thừa số lớn nhất của n.

Thuật toán này tính ra số Jacobi tuần hoàn theo công thức sau:

(1) $J(1,k) = 1$

(2) $J(a \times b,k) = J(a,k) \times J(b,k)$

(3) $J(2,k) = 1$ nếu $(k^2 - 1)/8$ là chia hết và $J(2,k) = -1$ trong các trường hợp khác.

(4) $J(b,a) = J((b \bmod a), a)$

(5) Nếu gcd(a,b)=1:

$J(a,b) \times J(b,a) = 1$ nếu $(a-1).(b-1)/4$ là chia hết.

$J(a,b) \times J(b,a) = -1$ nếu $(a-1).(b-1)/4$ là còn dư.

Trên thực tế có thể tính được ký hiệu Jacobi một cách thuận lợi hơn nếu dựa vào một trong các tính chất sau, giả sử m và n là các số nguyên lẻ, $a,b \in Z$:

- ✓ $J(a*b, n) = J(a,n) * J(b,n)$ do đó $J(a^2, n) = 1$
- ✓ $J(a, m*n) = J(a,m) * J(a,n)$
- ✓ Nếu $a \equiv b \pmod n$ thì $J(a,n) = J(b,n)$
- ✓ $J(1,n) = 1$
- ✓ $J(-1,n) = (-1)^{(n-1)/2}$
- ✓ $J(m,n) = J(n,m) * (-1)^{(m-1)*(n-1)/4}$

2/. Một số thuật toán kiểm tra tính nguyên tố

a). Thuật toán Soloway-Strassen [3]

Thuật toán này sử dụng hàm Jacobi.

Thuật toán kiểm tra số p là số nguyên tố:

1. Chọn ngẫu nhiên một số a nhỏ hơn p.
2. Nếu ước số chung lớn nhất $\gcd(a, p) \neq 1$ thì p là hợp số.
3. Tính $j = a^{(p-1)/2} \bmod p$.
4. Tính số Jacobi J(a,p).
5. Nếu $j \neq J(a,p)$, thì p không phải là số nguyên tố.
6. Nếu $j = J(a,p)$ thì ta nói p có thể là số nguyên tố với chắc chắn 50%.

Lặp lại các bước này n lần, mỗi lần với một giá trị ngẫu nhiên khác nhau của a. Phần dư của hợp số với n phép thử là không quá 2^n.

Độ phức tạp của thuật toán là $O(\log_2^p)$.

b). Thuật toán Miller-Rabin [6]

Thuật toán này được phát triển bởi Rabin, dựa trên một phần ý tưởng của Miller. Thực tế những phiên bản của thuật toán đã được giới thiệu tại NIST.

Input: Số tự nhiên lẻ n

Output: Nguyên tố hoặc hợp số

1. Phân tích $n - 1 = 2^s.m$, trong đó $s \geq 1$ và m là số tự nhiên lẻ
2. Chọn ngẫu nhiên số tự nhiên $a \in \{2,\ldots,n-1\}$
3. Đặt $b = a^m \bmod n$
4. Nếu $b \equiv 1 \bmod n$ thì đây là số nguyên tố. Kết thúc.
5. Cho k chạy từ 0 đến s-1:

    5.1 Nếu $b \equiv -1 \bmod n$ thì đây là số nguyên tố. Kết thúc.

    5.2 Thay $b := b^2 \bmod n$
6. Kết luận đây là hợp số. Kết thúc.

Xác suất sai lầm của thuật toán này là $\leq \frac{1}{4}$. Độ phức tạp của thuật toán là $O(log_2^n)$.

c). Thuật toán Lehmann [3]

Một phương pháp đơn giản hơn kiểm tra số nguyên tố được phát triển độc lập bởi Lehmann.

Sau đây là thuật toán với số bước lặp là 100:

1. Chọn ngẫu nhiên một số n để kiểm tra.
2. Chắc chắn rằng n không chia hết cho các số nguyên tố nhỏ như 2,3,5,7, và 11
3. Chọn ngẫu nhiên 100 số a_1, a_2, a_3,... a_{100} giữa 1 và n-1
4. Tính $a_i^{(n-1)/2}$ (mod n) cho tất cả $a_i = a_1 \ldots a_{100}$. Dừng lại nếu bạn tìm thấy a_i sao cho phép kiểm tra là sai.
5. Nếu $a_i^{(n-1)/2} = 1$ (mod n) với mọi i thì n có thể là hợp số.

    Nếu $a_i^{(n-1)/2} \neq 1$ hoặc -1 (mod n) với i bất kỳ, thì n là hợp số.

    Nếu $a_i^{(n-1)/2} = 1$ hoặc -1 (mod n) với mọi $i \neq 1$, thì n là số nguyên tố.

d). Thuật toán AKS (Agrawal-Kayal-Saxene) [6]

Tháng 8 năm 2002, ba tác giả Manindra Agrawal, Neeraj Kayal và Nitin Saxena (Viện công nghệ Kanpur Ấn Độ) đã công bố thuật toán kiểm tra tính nguyên tố với độ phức tạp thời gian đa thức.

Thuật toán xuất phát từ ý tưởng sau: một số nguyên n ($n>2$) là số nguyên tố khi và chỉ

khi: $(x-a)^n \equiv (x^n-a)(mod\ n)$ (*) đúng với mọi số nguyên a là số nguyên tố cùng nhau với n (hoặc chỉ cần đúng với số giá trị của a, đặc biệt khi a = 1).

Với định lý trên, thời gian tính của thuật toán sẽ là một hàm mũ. Tiến hành rút gọn hai vế của đẳng thức trên theo modulo $x^r - 1$. Sau đó lại rút gọn các hệ số của kết quả thu được theo modulo n. Ta được biểu thức sau:

$$(x-a)^n \equiv (x^n-a)(mod\ x^r-1, n) \quad (**)$$

Hay ta có:

$$(x-a)^n - (x^n-a) = nf + (x^r-1)g \quad (***)$$

Biểu thức (**) có thể xảy ra trong một số trường hợp p là hợp số, nhưng người ta đã chứng minh được rằng không hợp số p nào thoả mãn (**) với mọi a nằm trong vùng $1 < a < \sqrt{r}\ log\ p$. Như vậy việc kiểm tra (**) cho các số a nằm trong vùng này sẽ tương đương với việc kiểm tra tính nguyên tố của p và thuật toán có độ phức tạp là đa thức.

Ví dụ:

- Với $n = 3$, $a = 1$ ta có:

$$(x-a)^n \bmod n = (x-1)^3 \bmod 3 = (x^3 - 3x^2 + 3x - 1) \bmod 3$$
$$= (x^3 - 1) \bmod 3 = (x^n - a) \bmod n$$

- Với $n = 3$, $a = 2$:

$$(x-a)^n \bmod n = (x-2)^3 \bmod 3 = (x^3 - 6x^2 + 12x - 8) \bmod 3$$
$$= (x^3 - 2) \bmod 3 = (x^n - a) \bmod n$$

Thuật toán:

Input: Số tự nhiên lẻ n > 1

Output: Nguyên tố hoặc hợp số

if (*n* có dạng a^b; $a,b \in N$, $b > 1$) output *n* là hợp số. Kết thúc.

r = 2;

while (r < n) {

if (UCLN(n,r) >1) output *n* là hợp số. Kết thúc.

if (*r* là số nguyên tố) {

Tìm số *q* - là ước nguyên tố lớn nhất của *r-1*;

if ($(q \geq 4\sqrt{r}\ \log_2 n)$ và $(n^{\frac{r-1}{q}} \neq 1 \pmod r)$) break;

}

r:= r + 1; if (r ≥ n) output *n* là số nguyên tố. Kết thúc.

}

if ((n-1) $\leq 2\sqrt{r}\log_2{}^n$) {

for a = (r+1) to (n-1)

if (UCLN(a,n) >1) output *n* là hợp số. Kết thúc.

}

for a = 1 to $2\sqrt{r}\log_2{}^n$

if $((x-a)^n \not\equiv x^n - a \pmod{x^r - 1})$ output *n* là hợp số. Kết thúc.

output *n* là số nguyên tố.

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, nhu cầu tìm kiếm 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 nào đó, thì dùng tiếp một thuật toán tất định (chẳng hạn thuật toán Agrawal-Kayal-Saxena) để đảm bảo chắc chắn 100% rằng số n là nguyên tố. Thuật toán Agrawal-Kayal-Saxena được chứng minh là có độ phức tạp thời gian đa thức cỡ $O((\log n)^{12})$ khi thử trên số n; và nếu số nguyên tố được thử có dạng Sophie Germain, tức dạng $2p+1$, thì độ phức tạp thời gian chỉ còn $O((\log n)^6)$. [4]

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. Đồng thời, đây cũng là một bài toán quan trọng trong việc tấn công hệ mật RSA. Như chúng ta đã biết thì độ an toàn của hệ mật RSA chủ yếu phụ thuộc vào độ "khó" của bài toán phân tích số nguyên lớn modulo n thành tích của hai số nguyên tố p và q. Nếu chúng ta thực hiện được công việc này trong thời gian "*đủ nhanh*" thì việc phá được hệ mật này là hoàn toàn có thể. Có nhiều thuật toán để làm việc này, tuy nhiên trong luận văn này tôi chỉ trình bày một số thuật toán thường được sử dụng nhất.

1/. Thuật toán sàng Eratosthenes [2]

Thuật toán phân tích số nguyên N được mô tả như sau:

(1) p = 1

(2) p = p+1

(3) Tính r = N mod p

Nếu r > 0 thì quay về bước (2).

Ngược lại, p là ước của N. Dừng chương trình.

Đây là thuật toán có tính phổ thông và mặc dù như chúng ta đã biết là thuật toán rất "*tồi*" vì thời gian tính của nó là O($\sqrt{n}$) nhưng nếu N có ước nhỏ thì việc áp dụng thuật toán này lại rất hiệu quả.

2/. Thuật toán sàng đồng dư [2]

Thuật toán được mô tả như sau:

(1) Lấy ngẫu nhiên hai số a và b, với a,b $\in Z_n^*$

(2) Kiểm tra gcd((a-b) mod n,n) > 1 hoặc gcd((a+b) mod n,n) > 1

- Nếu đúng thì gcd((a-b) mod n,n) > 1 hoặc gcd((a+b) mod n,n) > 1 là ước của n. Dừng chương trình.
- Ngược lại thì quay về (1)

Bây giờ, chúng ta hãy tạm dừng để phân tích thuật toán dưới góc độ xác suất như sau:

Cho p là ước nguyên tố nhỏ nhất của n, thế thì "*cần có tối thiểu bao nhiêu cặp a,b được xét đến để xác suất có ít nhất một cặp trong số đó thỏa mãn ((a ± b) mod p)≡ 0 ≥ 0.5 ?* ". Bài toán trên còn được gọi là bài toán "*trùng ngày sinh*" và số m tối thiểu cần tìm trong bài toán sẽ là m ≈ c.p, với c là một hằng số tính được nào đó. Thuật toán có thể thành công với xác suất > 0.5, sau không quá m bước.

Bằng cách duyệt dần thì thời gian của thuật toán không khác gì thời gian của phép sàng. Tác giả J.M.Pollard đã sử dụng một phương pháp còn gọi là "*phương pháp δ*". Chỉ cần thông qua $\sqrt{m}$ bước có thể duyệt được m cặp khác nhau như đã nêu trên trong thuật toán.

3/. Thuật toán Pollard [2]

Thuật toán hiệu quả trong việc tìm các ước nhỏ là thuật toán dựa vào phương pháp δ và được gọi là thuật toán Pollard. Thời gian tính của thuật toán này chỉ còn là O($\sqrt{p}$). Với p là ước nguyên tố nhỏ nhất của n. Trong trường hợp tồi nhất (p≈$\sqrt{n}$) thì thời gian tính của thuật toán cũng chỉ là $\sqrt[4]{n}$.

Phương pháp δ của Pollard:

Tìm hai phần tử đồng dư modulo p (a ≡ ± b mod p) nhưng không đồng dư modulo n.

Lúc này p sẽ là ước của gcd(n, (a ± b) mod n).

Có thể mô tả thuật toán như sau:

Chọn dãy giả ngẫu nhiên { x_i mod n, i= 1,2,…} được xác định như sau: $x_{i+1} \equiv (x_i^2 + a)$ mod n với $a \neq 0$ và $a \neq -2$ còn giá trị đầu x_0 tùy ý.

Thuật toán:

(1) i = 0

(2) i:= i+1

(3) Xét gcd(($x_{2i} - x_i$) mod n, n) > 1

- Nếu đúng, ta có p = gcd(($x_{2i} - x_i$) mod n, n). Dừng chương trình.
- Ngược lại, quay về bước (2)

Chúng ta đi phân tích thời gian của thuật toán:

$x_{2i} - x_i \equiv (x_{2i-1}^2 + a) - (x_{i-1}^2 + a) \equiv (x_{2i-1}^2 - x_{i-1}^2)$

$\equiv (x_{2i-1} - x_{i-1})(x_{2i-1} + x_{i-1})$

$\equiv (x_{2i-1} + x_{i-1})(x_{2i-2} + x_{i-2})...(x_i + x_0)(x_i - x_0)$

Tại bước thứ i, chúng ta xét đến i+1 cặp khác nhau và cũng dễ dàng nhận ra rằng các cặp được xét trong mọi bước là không giống nhau, do đó, hiển nhiên với $\sqrt{p}$ bước chúng ta đã có p cặp khác nhau được xét đến và như đã phân tích ở trên. Thuật toán thành công với xác suất > 0.5 hay thuật toán của Pollard được thực hiện trong O($\sqrt{p}$) bước.

4/. Thuật toán p-1 [2]

Thuật toán p-1 của Pollard là thuật toán phân tích số nguyên n dựa vào phân tích của p-1 với p là một ước nguyên tố của n. Đây là thuật toán có tác dụng nếu ta biết được các ước nguyên tố của một thừa số p của n nói chung và đặc biệt nếu n có một thừa số nguyên tố p mà p-1 chỉ gồm những ước nguyên tố nhỏ nhất thì thuật toán có hiệu quả. Thuật toán này chỉ có hai đầu vào là số nguyên lẻ n cần được phân tích và một số b.

* *Thuật toán*

Input: n, cận b

Output: trả lời:

- Thành công và đưa ra thừa số của n
- Không tìm được thừa số của n

Method:

Bước 1: a:=2

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

Bước 3: d:= UCLN(a-1,n)

Bước 4: If (1 < d < n) then

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

else Write ('Không tìm được thừa số của n');

End.

* *Ví dụ*: Giả sử n = 15770708441 và b = 180.

Áp dụng thuật toán trên, ta thấy rằng: a = 11620221425 ở bước 3, còn d = 135979 và n/d = 115979.

Thực tế, phân tích n thành các thừa số nguyên tố là: 15770708441 = 135979 × 115979

Phép phân tích thành công do (p-1) = (135979 - 1) = 135978 chỉ gồm các thừa số nguyên tố nhỏ: 135978 = 2 × 3 × 131 × 173

Trong thuật toán có (b-1) lũy thừa theo modulo, mỗi lũy thừa cần nhiều nhất là $2\log_2{}^b$ phép nhân modulo dùng thuật toán bình phương và nhân. Việc tìm ước chung lớn nhất có thể được thực hiện trong khoảng thời gian $O((\log_2{}^n)^3)$ bằng thuật toán Euclid. 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, với phép chọn b như vậy thì 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 nhanh hơn phép chia thử.

Điểm bất lợi của thuật toán này là nó yêu cầu n phải có ước nguyên tố p sao cho p-1 chỉ có các thừa số nguyên tố bé. Ta có thể xây dựng được hệ mật RSA với modulo n = p.q hạn chế được việc phân tích theo phương pháp này. Trước tiên, tìm một số nguyên tố lớn p_1 sao cho $p = 2p_1 + 1$ cũng là một số nguyên tố và một số nguyên tố lớn q_1 sao cho $q = 2q_1 + 1$ cũng là một số nguyên tố. Khi đó, modulo của RSA là n = p.q sẽ chống được cách phân tích theo phương pháp p-1.

5/. Thuật toán p ± 1 [2]

Thuật toán này của Williams cũng dựa vào kết quả phân tích của p±1 với p là một ước nguyên tố của n. Để tiện nghiên cứu phương pháp p ± 1, trước hết ta điểm lại một số kết quả liên quan đến dãy Lucas.

* *Định nghĩa dãy Lucas*:

Cho a, b là hai nghiệm của phương trình: $x^2 - px + q = 0$ (1)

Ký hiệu: $u_m = \dfrac{a^m - b^m}{a - b}$ và $v_m = a^m + b^m$ (2)

Các dãy $\{u_m\}$, $\{v_m\}$ với m = 0,1,2,… được gọi là dãy Lucas của phương trình (1). Ngược lại, phương trình (1) được gọi là phương trình đặc trưng của dãy (2).

** Các tính chất*:

- Nếu i là ước của j thì u_i sẽ là ước của u_j.
- Ta có: $u_0 = 0$, $u_1 = 1$, $v_0 = 2$, $v_1 = p$ và $\forall m > 1$ thì các phần tử u_m, v_m của dãy Lucas được tính theo công thức sau:

$$\begin{bmatrix} u_{m+1} & v_{m+1} \\ u_m & v_m \end{bmatrix} = \begin{bmatrix} p & -q \\ 1 & 0 \end{bmatrix}^m \begin{bmatrix} u_1 & v_1 \\ u_0 & v_0 \end{bmatrix}$$

** Định lý*: $\{u_m\}$ là dãy Lucas của phương trình (1) với $p^2 - 4q = d^2\Delta$ có Δ không có ước chính phương (hay còn gọi là bình phương tự do).

Nếu p không là ước của Δq thì $p - \left[\frac{\Delta}{p}\right] \equiv 0 \bmod p$ (với $\left[\frac{\Delta}{p}\right]$ là ký hiệu Lagrăng).

Nếu như cơ sở của phương pháp p-1 là dựa vào kết quả của định lý Fermat thì với kết quả của Lucas chúng ta cũng phát triển thành một phương pháp phân tích số nguyên một cách tương tự nhưng dựa vào kết quả phân tích của $p \pm 1$ với p là ước nguyên tố của N.

** Thuật toán*:

(1) $Q = 2_1^{\log_2^N} \ldots q_k^{\log_{qk}^N}$, i = 1, j = 0

(2) Lấy Δ không có ước chính phương ngẫu nhiên trong Z^*_n. Tìm R, S nguyên sao cho: $R^2 - 4S = \Delta d^2$ với $d \neq 0$ nào đó.

Xét gcd(ΔQ,n) >1

- Nếu đúng, ta có ước của n là gcd(ΔQ, n). Dừng chương trình.
- Ngược lại, tính $b \equiv u_Q \bmod n$ (phần tử thứ Q trong dãy Lucas của phương trình $x^2 - Rx + S = 0$)

(3) Xét đẳng thức b = 0

- Nếu đúng, chuyển sang bước (4)
- Ngược lại, chuyển sang bước (6)

(4) Xét $j < \log_q^n$

- Nếu đúng thì: j = j + 1, Q = Q/q. Quay về bước (3)
- Ngược lại, chuyển sang bước (5)

(5) Xét i < k

- Nếu đúng thì: i = i+1, j = 0

- Nếu b ≠ 1thì: Q = Q*q_i. Quay về bước (4)

- Ngược lại, quay về bước (1)

(6) Xét gcd(b,n) > 1

- Nếu đúng thì ước của n là: gcd(b,n). Dừng chương trình.

- Ngược lại, quay về bước (4).

Trước hết, ta thấy rằng, các bước và việc làm trong mỗi bước của thuật toán gần như giống hệt với thuật toán của Pollard nhằm để vét hết các khả năng p+1 (trong trường hợp $\left[\frac{\Delta}{p}\right]$ = -1) và p-1 (trong trường hợp $\left[\frac{\Delta}{p}\right]$ = 1) là ước của Q. Việc xét đẳng thức b=0 trong mỗi bước, nếu sai nhằm đảm bảo cho ta b không là bội của n và nếu p+1 hoặc p-1 là ước của Q thì theo các kết quả ở tính chất và định lý trên cho ta b là bội của p và như vậy gcd(b,n) là ước thực sự của n.

Tóm lại, thuật toán trên rõ ràng hiệu quả trong cả hai trường hợp p+1 hoặc p-1 chỉ gồm các ước nguyên tố nhỏ. Tuy nhiên, căn cứ vào công thức tính các giá trị của dãy Lucas ta thấy rằng hệ số nhân của thuật toán này là lớn hơn nhiều so với thuật toán của Pollard trong trường hợp cùng phân tích được n với ước p của nó có p-1 chỉ gồm các ước nhỏ bởi vì thay cho việc tính một lũy thừa thông thường thì thuật toán của Lucas phải tính một lũy thừa của một ma trận.

Từ thuật toán trên, ta có thể kết luận:

- p phải là một số lớn.

- Các ước phải có kích thước xấp xỉ nhau.

- Các ước không được xấp xỉ nhau về giá trị.

- Ước nguyên tố p của modulo n không được có p+1 hoặc p-1 phân tích hoàn toàn ra các thừa số nguyên tố nhỏ.

- Không có số Lucas u_i = 0 mod p với i bé đối với các phương trình đặc trưng có biểu thức Δ nhỏ.

- p phải có khoảng cách lũy thừa 2 đủ lớn.

6/. Tìm nhân tử lớn nhất thứ nhất $\leq \sqrt{N}$ [7]

Định lý Fermat: Giả sử n là một số nguyên dương lẻ có dạng n = p.q trong đó p≤q và p, q là các số nguyên tố. Khi đó, biểu thức n có thể được viết dưới dạng: n = $t^2 - s^2$ (trong đó t, s là các số nguyên dương). Các số nguyên t, s, p và q có mối quan hệ:

$$t = \frac{p+q}{2} \text{ và } s = \frac{q-p}{2}$$

Phương pháp này được xây dựng dựa trên định lý Fermat, cụ thể:

(1) Đặt: x = 2.$\lfloor\sqrt{N}\rfloor$ + 1; y = 1; r = $\lfloor\sqrt{N}\rfloor^2$ - n

(2) If r ≤ 0 go to (4)

(3) r = r – y; y = y+2

goto (2)

(4) If (r =0) then Kết thúc thuật toán.

Khi đó chúng ta sẽ có:

n = $\left[\frac{x-y}{2}\right]\left[\frac{x+y-2}{2}\right]$ {Đây chính là hai nhân tử p và q của n}

và $\frac{x-y}{2}$ là phân số có giá trị lớn nhất $\leq \sqrt{N}$

(5) r = r + x

x = x+2

goto (3)

Trong đó: $\lfloor x \rfloor$ *là số nguyên bé nhất* $\geq x$.

1.4.3. Bài toán tính logarit rời rạc theo modulo

Đây cũng là một bài toán khá quan trọng trong hệ mật khóa công khai. Ta biết rằng, độ an toàn của hệ mật mã Elgamal phần lớn phụ thuộc vào bài toán này. Nếu có một thuật toán để giải được bài toán này trong thời gian "*đủ nhanh*" thì khả năng phá được hệ mật này là hoàn toàn có thể.

Bài toán này được phát biểu như sau:

Cho: I = (p, α, β) trong đó p là số nguyên tố, α là phần tử nguyên thủy $\in Z_p^*$ và $\beta \in Z_p^*$

Mục tiêu: Hãy tìm một số nguyên duy nhất a ($1 \leq a \leq p-1$) sao cho: $\alpha^a \equiv \beta \pmod p$.

Ta sẽ xác định số nguyên $a = log_\alpha^\beta \pmod p$.

Thuật toán tầm thường để giải bài toán này đó là duyệt toàn bộ các số a từ 1 đến p-1, cho đến khi tìm được giá trị a thỏa mãn biểu thức: $\alpha^a \equiv \beta \pmod p$. Tuy nhiên, nếu p là một số rất lớn thì thuật toán này sẽ kém hiệu quả. Một biến dạng của thuật toán này với ít nhiều hiệu quả hơn đó là thuật toán Shanks. [4]

1/. Thuật toán Shanks [14]

Đây là thuật toán tính logarit trên trường hữu hạn do Danied Shanks đề xuất.

a). Ý tưởng như sau:

(1) Đặt m = $\lfloor \sqrt{p-1} \rfloor$

(2) Tính $\alpha^{mj} \bmod p$ với $0 \le j \le m-1$

(3) Sắp xếp m cặp với thứ tự (j, $\alpha^{mj} \bmod p$), có lưu ý tới các tọa độ thứ hai của các cặp này, ta sẽ thu được danh sách L_1

(4) Tính $\beta\alpha^{-i} \bmod p$ với $0 \le i \le m-1$

(5) Sắp xếp m cặp với thứ tự (i, $\beta\alpha^{-i} \bmod p$), có lưu ý tới các tọa độ thứ hai của các cặp này, ta sẽ thu được danh sách L_2

(6) Tìm một cặp (j, y) $\in L_1$ và một cặp (i, y) $\in L_2$ (tức là một cặp có tọa độ thứ hai bằng nhau).

(7) Xác định $\log_\alpha\beta = mj + i \bmod (p-1)$

b). Nhận xét:

- Nếu cần, các bước (1) và (2) có thể tính toán trước, điều này không làm ảnh hưởng tới thời gian chạy tiệm cận.

- Tiếp theo, rõ ràng nếu (j, y) $\in L_1$ và (i, y) $\in L_2$ thì: $\alpha^{mj} = y = \beta\alpha^{-i}$, bởi vậy: $\alpha^{mj+i} = \beta$ như mong muốn. Ngược lại, đối với β bất kì ta có thể viết: $\log_\alpha\beta = mj+i$ trong đó $0 \le j,i \le m-1$. Vì thế phép tìm kiếm ở bước (5) chắc chắn thành công.

Như vậy, thuật toán này có thể tìm được logarit rời rạc với thời gian tính cỡ O(m) và không gian nhớ cỡ O(m). Chú ý là bước (5) có thể thực hiện một cách đồng thời qua từng danh sách L_1 và L_2.

c). Ví dụ

Cho p = 809 và ta phải tìm $\log_3 525$.

Ta có: $\alpha = 3$, $\beta = 525$ và m = $\lfloor \sqrt{808} \rfloor = 29$.

Khi đó: $\alpha^{29} \bmod 809 = 99$

Trước tiên, ta tính các cặp (j, $99^j \bmod 809$) với $0 \le j \le 28$.

Ta nhận được danh sách L_1: (0,1); (1,99); (2,93); (3,308); (4,559); (5,329); (6,211); (7,664); (8,207); (9,268); (10,644); (11,654); (12,26); (13,147); (14,800); (15,727); (16,781); (17,464); (18,314); (19,275); (20,582); (21,496); (22,564); (23,15); (24,676); (25,586); (26,575); (27,295); (28,81).

Danh sách L_2 chứa các cặp (i, $525\times(3^i)^{-1} \bmod 809$) với $0 \le i \le 28$.

Danh sách này gồm:

(0,525); (1,175); (2,328); (3,379); (4,396); (5,132); (6,44); (7,554); (8,724); (9,511); (10,440); (11,686); (12,768); (13,256); (14,355); (15,388); (16,399); (17,133); (18,314); (19,644); (20,754); (21,496); (22,564); (23,15); (24,676); (25,356); (26,658); (27,489); (28,163).

Bây giờ, nếu xử lý hết cả hai danh sách thì ta sẽ tìm được cặp (10,644) trong L_1 và (19,644) trong L_2. Bây giờ, ta có thể tính: $\log_3^{525} = 29\times10+19 = 309$. Kiểm tra lại ta thấy rằng quả thực: $3^{309} \equiv 525 \pmod{809}$.

2/. Thuật toán Pohlig - Hellman [14]

Thuật toán thứ hai tôi muốn đề cập đến là thuật toán Pohlig - Hellman. Cơ sở toán học của thuật toán này là định lý phần dư Trung Hoa sau đây. [1]

- *Định lý phần dư Trung Hoa:* Cho tập số nguyên tố cùng nhau từng đôi một: $m_1, m_2 \ldots m_r$. Với mỗi bộ số nguyên bất kỳ: $a_1, a_2, \ldots, a_r$ thì hệ phương trình đồng dư:

$$x \equiv a_i \pmod{m_i} \text{ với } (i = 1,2,\ldots,r)$$

luôn có *nghiệm duy nhất* theo modulo m với ($m = m_1.m_2.\ldots m_r$).

Nghiệm này có thể tính được theo công thức sau:

$x = a_1m_2m_3\ldots m_rb_1 + m_1a_2m_3\ldots m_rb_2 + m_1m_2a_3m_3\ldots m_rb_3 + \ldots + m_1m_2\ldots m_{r-1}a_r\, b_r \pmod{m_1.m_2\ldots m_r}$, trong đó $b_i = (m_1.m_2\ldots m_{i-1}m_{i+1}\ldots m_r)^{-1} \pmod{m_i}$, với ($i = 1,2,\ldots,r$).

- *Nhận xét*: Định lý trên cho phép chúng ta tính được đồng dư theo modulo của một số lớn (tích của nhiều số nguyên tố cùng nhau), thông qua tính toán đồng dư theo modulo các số nhỏ (từng thừa số).

- *Ví dụ*: Tìm nghiệm của hệ phương trình:

$$\begin{cases} x \equiv 3118 \pmod{5353} \\ x \equiv 139 \pmod{391} \\ x \equiv 239 \pmod{247} \end{cases}$$

Vì các số 5353, 391, 247 nguyên tố cùng nhau, nên theo định lý trên hệ phương trình sẽ có nghiệm duy nhất theo modulo m = 5353*391*247 = 516976681.

Để tìm x mod m ta tính:

$m_1 = m/5353 = 96577 \rightarrow y_1 = 96577^{-1} \bmod 5353 = 5329$

$m_2 = m/391 = 1322191 \rightarrow y_2 = 1322191^{-1} \bmod 391 = 16$

$m_3 = m/247 = 2093023 \rightarrow y_3 = 2093023^{-1} \bmod 247 = 238$

$x = 31188.96577.5329 + 139.1322191.16 + 239.2093023.238 \pmod{m}$

x = 13824 (mod m)

Từ định lý trên, nếu p - 1 có dạng phân tích chính tắc là: $p-1=\prod_{i=1}^{k} p_i^{c_i}$ (p_i là số nguyên tố đặc biệt) thì để tính được giá trị a = $\log_\alpha^\beta$ (mod p-1) ta tìm các số a_i sao cho $a_i \equiv$ a mod $p_i^{c_i}$ với $1 \le i \le k$. Sau khi tìm được các số a_i thì hệ phương trình: x $\equiv a_i$ mod $p_i^{c_i}$ ($1 \le i \le k$) được giải theo định lý phần dư Trung Hoa sẽ cho lời giải x = a (mod p-1) cần tìm. Vấn đề là xác định các số a_i mod $p_i^{c_i}$ ($1 \le i \le k$). Để thực hiện điều này, ta giả sử rằng q là số nguyên tố.

$$p - 1 \equiv 0 \pmod{q^c}$$

$$p - 1 \not\equiv 0 \pmod{q^{c+1}}$$

Ta sẽ chỉ ra cách tính giá trị: x = a mod q^c với ($0 \le x \le q^c$-1). Ta có thể biểu diễn x theo cơ số q như sau:

$$x = \sum_{i=0}^{c-1} a_{i.} q^i$$

trong đó $0 \le a_i \le$ q-1 với $0 \le i \le$ c-1. Cũng có thể biểu diễn như sau: a = x + s.q^c với s là một số nguyên nào đó.

Bước đầu tiên của thuật toán tính a_0. Kết quả chính ở đây là:

$$\beta^{(p-1)/q} \equiv \alpha^{(p-1)a_0/q} \pmod{p}$$

Để thấy rõ điều đó cần chú ý rằng:

$$\beta^{(p-1)/q} \equiv \alpha^{(p-1)(x+q^c s)/q} \pmod{p}$$

Điều này đủ để cho thấy:

$$\alpha^{(p-1)(x+q^c s)/q} \equiv \alpha^{(p-1)a_0/q} \pmod{p}$$

Kết quả này đúng khi và chỉ khi:

$$\frac{(p-1)(x+q^c s)}{q} \equiv \frac{(p-1)a_0}{q} \pmod{p-1}$$

Tuy nhiên: $\frac{(p-1)(x+q^c s)}{q} - \frac{(p-1)a_0}{q} = \frac{(p-1)}{q}(x+q^c s-a_0) = \frac{(p-1)}{q}\left(\sum_{i=0}^{c-1} a_i q^i + q^c s - a_0\right)$

$= \frac{(p-1)}{q}\left(\sum_{i=1}^{c-1} a_i q^i + q^c s\right) = (p-1)\left(\sum_{i=1}^{c-1} a_i q^{i-1} + q^{c-1} s\right) \equiv 0 \pmod{p-1}$

Đó là điều cần chứng minh. Do đó, ta sẽ bắt đầu bằng việc tính: $\beta^{(p-1)/q}$ mod p. Nếu $\beta^{(p-1)/q} \equiv 1$ (mod p) thì $a_0 = 0$. Ngược lại, chúng ta sẽ tính liên tiếp các giá trị: $\gamma = \alpha^{(p-1)/q}$

mod p, γ^2 mod p,... cho tới $\gamma^i \equiv \beta^{(p-1)/q} \pmod p$ với một giá trị *i* nào đó. Khi điều này xảy ra ta có a_0 =i.

Bây giờ, nếu c = 1 thì ta đã thực hiện xong. Ngược lại, nếu c > 1 thì phải tiếp tục xác định a_1. Để làm điều đó ta phải xác định: $\beta_1 = \beta\alpha^{-a_o}$ và kí hiệu: $x_1 = \log_\alpha \beta_1 \bmod q^c$.

Dễ thấy rằng:

$$x_1 = \sum_{i=1}^{c-1} a_{i.} q^i$$

Vì thế dẫn đến:

$$\beta_1^{(p-1)/q^2} \equiv \alpha^{(p-1)a_1/q} (\text{mod } p)$$

Như vậy, ta sẽ tính $\beta_1^{(p-1)/q2} \pmod p$ rồi tìm i sao cho:

$$\gamma^i \equiv \beta_1^{(p-1)/q^2} (\text{mod } p)$$

Khi đó $a_1 = i$.

Nếu c = 2 thì công việc kết thúc; nếu không, phải lặp lại công việc này c - 2 lần nữa để tìm $a_2,...,a_{c-1}$.

- *Thuật toán Pohlig - Hellman để tính $\log_\alpha \beta \pmod{q^c}$*

(1) Tính $\gamma = \alpha^{(p-1)/q} \pmod p$ với ($0 \le i \le q-1$)

(2) Đặt j = 0 và $\beta_j = \beta$

(3) While ($j \le c-1$) do

(3.1) Tính $\delta = \beta_j^{(p-1)/q\, j+1} \pmod p$

(3.2) Tìm i sao cho $\delta = \gamma_i$

(3.3) $a_j = i$

(3.4) $\beta_{j+1} = \beta_j \alpha^{-a_j\, qj} \pmod p$

(3.5) j = j +1

Trong thuật toán này, α là phần tử nguyên thuỷ theo modulo p còn q là số nguyên tố.

$$p - 1 \equiv 0 \pmod{q^c}$$

và

$$p - 1 \not\equiv 0 \pmod{q^{c+1}}$$

Thuật toán tính các giá trị $a_0,...,a_{c-1}$ trong đó: $\log_\alpha^\beta \pmod{q^c}$.

$$\log_\alpha \beta \bmod q^c = \sum_{i=0}^{c-1} a_i q^i$$

- *Ví dụ*: Giả sử p = 29; khi đó n = p-1 = 28 = $2^2.7^1$

Giả sử $\alpha = 2$ và $\beta = 18$. Ta phải xác định $a = \log_2{}^{18}$.

Trước tiên, tính a mod 4 rồi tính a mod 7.

Ta sẽ bắt đầu bằng việc đặt q = 2, c = 2.

Trước hết $\gamma_0 = 1$ và $\gamma_1 = \alpha^{28/2} \bmod 29 = 2^{14} \bmod 29 = 28$

Tiếp theo: $\delta = \beta^{28/2} \bmod 29 = 18^{14} \bmod 29 = 28$

Vì $a_0 = 1$. Tiếp theo ta tính: $\beta_1 = \beta_0\alpha^{-1} \bmod 29 = 9$ và ${\beta_1}^{28/4} \bmod 29 = 9^7 \bmod 29 = 28$

Vì: $\gamma_1 \equiv 28 \bmod 29$

Ta có $a_1 = 1$. Bởi vậy $a \equiv 3 \pmod 4$.

Tiếp theo đặt q = 7 và c = 1, ta có $\beta^{28/7} \bmod 29 = 18^4 \bmod 29 = 25$

và $\gamma_1 = \alpha^{28/7} \bmod 29 = 2^4 \bmod 29 = 16$.

Sau đó tính: $\gamma_2 = 24$; $\gamma_3 = 7$; $\gamma_4 = 25$

Bởi vậy $a_0 = 4$ và $a \equiv 4 \pmod 7$

Cuối cùng giải hệ phương trình:

$$\begin{cases} a \equiv 3 \pmod 4 \\ a \equiv 4 \pmod 7 \end{cases}$$

bằng định lý phần dư Trung Hoa, ta nhận được $a \equiv 11 \pmod{28}$. Điều này có nghĩa là ta đã tính được $\log_2{}^{18}$ trong Z_{29} là 11.

Thuật toán Pohlig - 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 mà có ít nhất một thừa số nguyên tố lớn thì thuật toán này cũng kém hiệu quả, trong trường hợp đó thì bài toán tính logarit rời rạc theo *modulo 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 = 2.q+1, trong đó q là số nguyên tố. Đó được 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 nhau, 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.

Kết luận chương 1

Trong chương này, luận văn đã trình bày một số vấn đề về số nguyên tố, độ phức tạp của thuật toán, khái niệm hàm một phía và hàm cửa sập một phía, các bài toán quan trọng trong mật mã.

Chương 2. CÁC PHƯƠNG PHÁP TẤN CÔNG CHỮ KÝ SỐ

2.1. Tổng quan về chữ ký số

2.1.1. Khái niệm chữ ký số

1/. Giới thiệu [1]

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 thường sử dụng 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 chuỗi các bit (*0 hoặc 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 bit nhỏ đặt phía dưới xâu bit 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 bit tài liệu. Kẻ gian khó có thể giả mạo "chữ ký số" nếu như không biết "khóa lập mã".

Để kiểm tra một "*chữ ký số*" thuộc về một "*tài liệu số*", người ta phải 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,"*chữ ký số*" còn dùng để kiểm tra tính toàn vẹn của tài liệu gốc.

Mặt mạnh của "*chữ ký số*" hơn "*chữ ký tay*" còn 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 (điện thoại di động) tại khắp mọi nơi, miễn là kết nối được vào mạng. Đỡ tốn bao thời gian, sức lực, chi phí.

"Ký số" thực hiện trên từng bit tài liệu, nên độ dài của "*chữ ký số*" ít nhất cũng bằng độ dài của 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.

2/. Sơ đồ chữ ký số [1]

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 khoá có thể.

S là tập các thuật toán ký.

V là tập các thuật toán kiểm thử.

Với mỗi khóa $k \in K$, có thuật toán ký $Sig_k \in S$, $Sig_k: P \rightarrow A$, có thuật toán kiểm tra chữ ký $Ver_k \in V, Ver_k: P \times A \rightarrow \{đúng, sai\}$, thoả mãn điều kiện sau với mọi $x \in P$, $y \in A$:

$$Ver_k(x, y) = \begin{cases} \text{Đúng, nếu } y = Sig_k(x) \\ \text{Sai, nếu } y \neq Sig_k(x) \end{cases}$$

** 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*" thì cần giữ bí mật cho 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.

2.1.2. Phân loại "chữ ký số"

Có nhiều loại chữ ký tùy theo cách phân loại, sau đây xin giới thiệu một số cách. [1]

1/. Phân loại chữ ký theo khả năng khôi phục thông điệp gốc

a). Chữ ký có thể khôi phục thông điệp gốc: Là loại chữ ký, trong đó người nhận *có thể* khôi phục lại được thông điệp gốc, đã được "*ký*" bởi "*chữ ký*" này.

Ví dụ: Chữ ký RSA.

b). Chữ ký không thể khôi phục thông điệp gốc: Là loại chữ ký, trong đó người nhận *không thể* khôi phục lại được thông điệp gốc, đã được "*ký*" bởi "*chữ ký*" này.

Ví dụ: Chữ ký Elgamal.

2/. Phân loại chữ ký theo mức an toàn

a). Chữ ký "không thể phủ nhận": Để tránh việc chối bỏ chữ ký hay nhân bản chữ ký

để sử dụng nhiều lần, người gửi chữ ký cũng 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.

b). Chữ ký "một lần": Để bảo đảm an toàn, "khóa ký" chỉ dùng một lần trên một tài liệu.

Ví dụ: Chữ ký một lần Lamport.

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).

2.2. Chữ ký RSA

2.2.1. Sơ đồ chữ ký [1]

1/. Sơ đồ

a). 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 $P = A = Z_n$

Tính bí mật $\phi(n) = (p-1).(q-1)$

Chọn khóa công khai $b < \phi(n)$, nguyên tố cùng nhau 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 cặp khóa (bí mật, công khai) $K = \{(a, b)/\ a,b \in Z_n, a*b \equiv 1(\text{mod } \phi(n))\}$

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

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

* *Chú ý:* So sánh giữa sơ đồ chữ ký và sơ đồ mã hóa RSA ta thấy có sự tương ứng:

- Việc ký chẳng qua là *mã hoá*, việc kiểm thử lại chính là việc *giải mã*.
- Việc "*ký số*" vào x tương ứng với việc "*mã hoá*" 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.

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=A=Z_n = Z_n . Tính bí mật ϕ(n) = (p-1).(q-1) = 2 * 4 = 8.

Chọn khóa công khai b = 3 < ϕ(n), nguyên tố cùng nhau với ϕ(n) = 8.

Khóa bí mật a = 3, là phần tử nghịch đảo của b theo mod ϕ(n): a*b ≡ 1 (mod ϕ(n)).

* *Ký số*:

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

* *Kiểm tra chữ ký*:

Ver_k (x, y) = đúng ⇔ $x \equiv y^b \pmod n$

⇔ $2 \equiv 8^3 \pmod{15}$

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

1/. Bị lộ một trong các giá trị: p, q, ϕ(n)

Nếu trong quá trình lập khóa mà người sử dụng vô tình để lộ nhân tử p, q hoặc ϕ(n) ra ngoài thì kẻ tấn công sẽ dễ dàng tính được khóa bí mật a theo công thức:

$$a*b \equiv 1 \pmod{\phi(n)}$$

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

⟶ *Giải pháp phòng tránh:* Quá trình tạo lập khóa phải được tiến hành ở một nơi kín đáo, bí mật. Sau khi thực hiện xong thì phải giữ cẩn thận khóa bí mật a, đồng thời hủy hết các giá trị trung gian: p, q, ϕ(n).

2/. Tấn công dựa theo khóa công khai n và b của người ký [8]

Lúc này, kẻ tấn công sẽ tìm cách phân tích giá trị *n* ra hai thừa số nguyên tố p và q. Từ đó, sẽ tính được ϕ(n)=(p-1).(q-1); cuối cùng tính được khóa bí mật *a*.

⟶ *Giải pháp phòng tránh:* Nên chọn số nguyên tố *p* và *q* đủ lớn để việc phân tích *n* thành tích của hai thừa số nguyên tố là khó có thể thực hiện được trong thời gian thực. Trong thực tế, người ta thường sinh ra các số lớn (ít nhất 100 chữ số), sau đó kiểm tra tính nguyên tố của nó.

3/. Khi nhiều người cùng sử dụng chung "modulo n" [8]

Khi có k người cùng đăng ký sử dụng chữ ký RSA, trung tâm phân phối khóa CA sẽ sinh ra 2 số nguyên tố p và q, rồi tính số *modulo n* = p*q. Sau đó, sinh ra các cặp khóa mã hóa/giải mã $\{e_i,d_i\}$. Trung tâm sẽ cấp cho người đăng ký thứ i khóa bí mật d_i tương ứng, cùng với các thông tin như: n, danh sách khóa công khai $\{e_i\}$ ($i=1...k$).

Lúc này, bất kỳ ai có thông tin công khai như 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:*

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

- *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.*

**** Việc sử dụng chung "modulo n" dẫn đến:***

a). Trường hợp 1: Một thành viên có thể sử dụng khóa công khai và khóa bí mật của mình để tính ra khóa bí mật của người 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 phải biết $\phi(n)$.

Để tìm được giá trị 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 vì $(e, \phi(n)) = 1$.

Khi đó, do t và e_1 nguyên tố cùng nhau nên tồn tại r và s sao cho $r.t + s.e_1 = 1$.

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 chung lớn nhất f của e_1 và t. Đồng thời cũng phải tìm được hai số r và s thỏa mãn $r.t + s.e_1 = f$

(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 đến 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.

b). Trường hợp 2: Việc sử dụng số *modulo n* 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 khác mà không hề bị phát hiện. Đó là sử dụng phương pháp xác suất để phân tích thừa số *modulo n* thành nhân tử hoặc sử dụng thuật toán tất định để tính toán ra số mũ giải mã mà không cần số *modulo n*. Ý tưởng cơ bản của kiểu tấn công này là phân tích số *modulo n* bằng cách tìm *căn bậc hai không tầm thường của 1 mod n*. Nghĩa là, tìm một số b thỏa mãn:

$$b^2 = 1 \bmod n$$

$$b \neq \pm \bmod n$$

$$1 < b < n-1$$

Nếu tìm được số b như thế, thì số *modulo n* có thể được phân tích theo cách sau:

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

$(b+1).(b-1)=0 \pmod n$. Hay $(b+1).(b-1)=s.n = s.p.q$ với 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 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 này tập trung vào việc 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 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: $a^{2^j \varphi} \equiv 1 \bmod n$

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

(3) Đặt: $b = a^{2^{j-1} \varphi} \bmod n$

(4) Nếu $b \neq -1 \bmod n$ thì nó là căn bậc hai không tầm thường của 1

(5) Nếu $b = -1 \bmod n$ thì quay lại bước (1)

De Laurentis đã chứng minh rằng:

Với a ngẫu nhiên (1<a<n-1)

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

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

Do đó, nếu ta xây dựng thuật toán xác suất, thử lần lượt với m giá trị ngẫu nhiên a theo tính chất: $(\exists j (1 \le j \le k)(a^{2^{j-1}k} \ne \pm 1 \bmod n \wedge a^{2^j \varphi} = 1 \bmod n))$

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 $\ge 1 - \frac{1}{2^m}$. Và khi đó, ta tìm được phân tích p và q của *modulo 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 những thông tin mà mình có. Cách tấn công vừa đề cập tới rất quan trọng vì nó chỉ ra rằng những thông tin về cặp khóa *mã hóa/giải mã* có thể cho phép tìm ra thừa số của modulo n.

⟶ *Giải pháp phòng tránh:* sử dụng giá trị *modulo n* khác nhau cho mỗi người tham gia.

4/. Sử dụng giá trị "modulo n" nhỏ

Như ta đã biết, trong sơ đồ chữ ký RSA thì công thức để tính giá trị chữ ký *y* trên bản rõ *x* như sau: $y = x^a \pmod n$ với ($y \in A$, $x \in P$, $P=A=Z_n$)

Lúc này, kẻ tấn công có thể tính được khóa bí mật a theo công thức sau:

$$a = \log_x^y \pmod n$$

do các giá trị: *x*, y, n là công khai. Đây chính là việc giải bài toán logarit rời rạc trên vành Z_n. Bởi vậy, nếu như giá trị *modulo n* mà nhỏ thì bằng cách áp dụng các thuật toán đã trình bày ở trên kẻ tấn công có thể tìm ra được khóa bí mật a.

⟶ *Giải pháp phòng tránh:* Nên chọn các số nguyên tố *p* và *q* đủ lớn để việc giải bài toán logarit rời rạc trên vành Z_n là khó có thể thực hiện được trong thời gian thực.

5/. Sử dụng các tham số (p-1) hoặc (q-1) có các ước nguyên tố nhỏ [8]

Nếu ta bất cẩn trong việc chọn các tham số p và q để cho (p-1) hoặc (q-1) có các ước nguyên tố nhỏ thì sơ đồ chữ ký sẽ trở nên mất an toàn. Bởi vì, khi (p-1) hoặc (q-1) có các ước nguyên tố nhỏ thì ta có thể dùng *thuật toán* (*p-1*) của Pollar để phân tích giá trị *modulo n* thành thừa số một cách dễ dàng.

⟶ *Giải pháp phòng tránh:* Chọn các tham số p và q sao cho (p-1) và (q-1) phải có các ước nguyên tố lớn.

2.2.3. Tấn công dạng 2: Giả mạo chữ ký (không tính trực tiếp khóa bí mật) [1]

1/. Người gửi G gửi tài liệu *x* cùng chữ ký *y* đến người nhận N, sẽ có 2 cách xử lý:

a). Ký trước, mã hóa sau

Người gửi G ký trước vào x bằng chữ ký $y = Sig_G(x)$, sau đó mã hoá x và y nhận được $z = e_G(x, y)$ rồi gửi z cho N.

Nhận được z, N giải mã z để được x, y. Tiếp theo, kiểm tra chữ ký $Ver_N(x, y) = true$?

b). Mã hóa trước, ký sau

Người gửi G mã hoá x trước bằng $u = e_G(x)$, sau đó ký vào u bằng chữ ký $v = Sig_G(u)$ rồi gửi (u, v) cho N.

Nhận được (u, v), N giải mã u nhận được x. Tiếp theo, kiểm tra chữ ký $Ver_N(u, v) = true$?

2/. Giả sử, H lấy trộm được thông tin trên đường truyền từ G đến N

+ Trong trường hợp a, H lấy được z. Trong trường hợp b, H lấy được (u,v).
+ Để tấn công vào x, trong cả hai trường hợp, H đều phải giải mã thông tin lấy được.
+ Để tấn công vào chữ ký, thay bằng chữ ký giả mạo, thì xảy ra hai trường hợp:
  - Trường hợp a, để tấn công chữ ký y, H phải giải mã z, mới nhận được y.
  - Trường hợp b, để tấn công chữ ký v, H đã có sẵn v', lúc này H chỉ việc thay v bằng v'.

H thay chữ ký v trên u, bằng chữ ký của H là $v' = Sig_H(u)$ rồi gửi (u, v') đến N.

Khi nhận được v', N kiểm thử thấy sai, gửi phản hồi lại cho G.

G có thể chứng minh đó là chữ ký giả mạo.

G gửi chữ ký đúng v cho N, nhưng quá trình truyền tin sẽ bị chậm lại.

Như vậy, trong trường hợp b, H có thể giả mạo chữ ký mà không cần phải giải mã.

⟶ *Giải pháp phòng tránh:* Hãy ký trước, sau đó mã hóa cả chữ ký.

2.3. Chữ ký Elgamal [1]

2.3.1. Sơ đồ chữ ký

1/. Sơ đồ

a). 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 phần tử nguyên thuỷ $g \in Z^*_p$. Đặt $P = Z^*_p$, $A = Z_p \times Z_{p-1}$.

Chọn khóa bí mật là $a \in Z^*_p$. Tính khóa công khai $h \equiv g^a \bmod p$.

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

Các giá trị (p, g, h) được công khai, còn giá trị a được giữ bí mật.

b). 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 nhau với p-1, do đó tồn tại $r^{-1} \bmod (p-1)$)

Chữ ký trên $x \in P$ là $y = Sig_a(x, r) = (\gamma, \delta)$, $y \in A$ (E1)

Trong đó $\gamma \in Z_p^*$, $\delta \in Z_{p-1}$

$$\gamma = g^r \bmod p \text{ và } \delta = (x - a*\gamma) * r^{-1} \bmod (p-1)$$

c). Kiểm tra chữ ký

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

** 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 $\delta = (x - a*\gamma) * r^{-1} \bmod (p-1)$ nên $(a*\gamma + r*\delta) \equiv x \bmod (p-1)$

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 $P = Z_p^*$, $A = Z_p^* \times Z_{p-1}$

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

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

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

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

Các giá trị (p, g, h) được công khai, còn giá trị a được giữ bí mật.

- *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 nhau với p-1, do đó tồn tại $r^{-1} \bmod (p-1)$.

Cụ thể: UCLN(r, p-1) = UCLN(235,462) = 1, nên $r^{-1} \bmod (p-1) = 235^{-1} \bmod 462 = 289$.

Chữ ký trên dữ liệu x = 112 là $(\gamma,\delta) = (16,108)$, trong đó:

$$\gamma = g^r \bmod p = 2^{235} \bmod 463 = 16$$

$$\delta = (x - a*\gamma) * r^{-1} \bmod (p-1) = (112 - 211*16) * 289 \bmod 462 = 108$$

- *Kiểm tra chữ ký*

Ver $_k$ (x, γ, δ) = 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.

3/. Độ an toàn

Bài toán căn bản để bảo đảm độ an toàn cho sơ đồ chữ ký Elgamal chính là bài toán tính logarit rời rạc. Biết khóa công khai $h \equiv g^a \bmod p$. Nên có thể xác định khóa bí mật a bằng cách tính $\log_g h$.

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

Khoá bí mật a có thể bị phát hiện, nếu khóa ngẫu nhiên r bị lộ, hoặc dùng r cho hai lần ký khác nhau.

1/. Số ngẫu nhiên r bị lộ

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

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

2/. Dùng r cho hai lần ký khác nhau

Giả sử dùng r cho 2 lần ký trên x_1 và x_2. Khi đó, (γ, δ_1) là chữ ký trên x_1 còn (γ, δ_2) là chữ ký trên x_2.

Khi đó, kẻ thám mã có thể tính được giá trị a như sau:

$$\beta^{\gamma} * \gamma^{\delta_1} \equiv \alpha^{x_1} \pmod p$$
$$\beta^{\gamma} * \gamma^{\delta_2} \equiv \alpha^{x_2} \pmod p$$

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

Đặt $\gamma = \alpha^r$, ta có $\alpha^{x_1 - x_2} \equiv \gamma^{k*(\delta_1 - \delta_2)} \pmod p \Leftrightarrow x_1 - x_2 \equiv r\,(\delta_1 - \delta_2) \bmod (p-1)$ (1)

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

$$x' = \frac{x_1 - x_2}{d}$$
$$\delta' = \frac{\delta_1 - \delta_2}{d}$$
$$p' = \frac{p-1}{d}$$

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

Vì $(\delta', p') = 1$ nên tính $\varepsilon = (\delta')^{-1} \bmod p'$ và tính $r = x' * \varepsilon \bmod p'$

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

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 giá trị a như trường hợp 1.

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

3/. Khóa bí mật a quá nhỏ

Nếu khóa bí 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 thước gần bằng số modulo n.

4/. Số ngẫu nhiên r quá nhỏ

Tương tự như đối với khóa bí mật a, số ngẫu nhiên r cũng phải bí mật. Trong trường hợp các tham số này quá nhỏ thì bằng phương pháp dò tìm đơn giản người ta cũng có thể tìm được chúng. Khi đó, sơ đồ chữ ký sẽ bị mất an toàn. Nếu r bị lộ, kẻ thám mã sẽ tính được khóa bí 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 thước gần bằng số modulo n.

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

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

+ Tin tặc H cố gắng giả mạo chữ ký trên x mà không hề biết khóa bí mật a. Như vậy, yêu cầu H phải tính được γ và δ.

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

Hiện nay, chưa có cách hữu hiệu nào để tính cả 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 ? Câu trả lời là: chưa rõ !

* Nếu chọn trước γ,δ sau đó tính x, thì tin tặc H sẽ phải đối đầu 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/. Giả mạo chữ ký cùng với tài liệu được ký

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

a). Cách 1

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

Chọn các số nguyên i, j (*0 ≤ i, j ≤ p-2)* và (j, p-1) = 1 và tính:

$$\gamma = g^i h^j \bmod p$$

$$\delta = -\gamma\, j^{-1} \bmod (p-1)$$

$$x = -\gamma\, i\, j^{-1} \bmod (p-1)$$

Trong đó, j^{-1} được tính theo mod (p-1) (*có 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:

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

Ví dụ:

* Chọn các tham số của sơ đồ chữ ký Elgamal:

Số nguyên tố p = 463, phần tử sinh g = 2, khóa bí mật a = 135.

Khóa công khai $h = g^a \bmod p = 2^{135} \bmod 463 = 272$.

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

Chọn i= 89, j = 125, *0 ≤ i, j ≤ p-2*, (j, p-1) = 1. Tính $j^{-1} \bmod (p-1) = 377$.

$\gamma = g^i * h^j \bmod p = 2^{89} * 272^{125} \bmod 463 = 218$

$\delta = -\gamma * j^{-1} \bmod (p-1) = -218 * 377 \bmod 462 = 50$

$x = -\gamma * i * j^{-1} \bmod (p-1) = -218*89*377 \bmod 462 = 292$

* (γ, δ) = (218, 50) là chữ ký trên x = 292, vì thỏa mãn điều kiện kiểm thử:

$h^\gamma * \gamma^\delta = 272^{218} * 218^{50} \equiv 322 (\bmod 463)$

$g^x = 2^{292} \equiv 322 (\bmod 467)$

b). 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 nguyên *k, i, j* thỏa mãn *0 ≤ k,i,j ≤ p-2*, $(k\ \gamma - j\ \delta, p-1) = 1$ và tính:

$$\lambda = \gamma^k g^i h^j \bmod p$$

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

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

* (λ,μ) là chữ ký trên x', vì 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 ứng, 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.

2.4. Chữ ký DSS [1]

2.4.1. Sơ đồ chữ ký

1/. Giới thiệu

Chuẩn chữ ký số (DSS: Digital Signature Standard) được đề xuất năm 1991, là dạng 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 hoá và giải mã 1 lần. 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.

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 đôi số bit của p, do đó nếu p dài 512 bit thì độ dài chữ ký là 1024 bit.

Trong ứng dụng dùng thẻ thông minh 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^*.

+ Trong sơ đồ ký Elgamal, công thức tính δ được sửa thành: $\delta = (x + a*\gamma)r^{-1} \bmod q$

+ Điều kiện kiểm thử $h^{\gamma}\ \gamma^{\delta} \equiv g^{x} \bmod p$ được sửa thành: $\alpha^{x*\delta^{-1}} * \beta^{\gamma*\delta^{-1}} \equiv \gamma (\bmod p)$

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

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

+ Với mỗi khóa (p, q, α, a, h), k'= a bí mật và 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 $r^{-1} \bmod q$)

* *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 $\Leftrightarrow (\alpha^{e_1} * h^{e_2} \bmod p) \bmod q = \gamma$

3/. Ví dụ

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

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 bí 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

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

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

* *Kiểm tra chữ ký:* $(\gamma, \delta) = (115, 87)$ là chữ ký đúng trên x = 1246.

$$e_1 = x*\delta^{-1} \bmod q = 1246 * 11 \bmod q = 83$$

$$e_2 = \gamma * \delta^{-1} \bmod q = 115*11 \bmod q = 70$$

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

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

2.4.2. 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 $\delta^{-1} \bmod q$ 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 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 là nhỏ, thì việc kiểm thử nhanh hơn việc ký. Đối với chữ ký DSS thì 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 một máy tính mạnh hơn, vì vậy nên xây dựng sơ đồ chữ ký ít 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. Đây là dạng cải tiến của chữ ký Elgamal. Bởi vậy, các dạng tấn công vào DSS tương tự như với chữ ký Elgamal.

2.5. Ứng dụng chữ ký số tại Việt Nam

Hiện tại, công nghệ chữ ký số tại Việt Nam có thể sử dụng trong các giao dịch để mua bán hàng trực tuyến, đầu tư chứng khoán trực tuyến. Ngoài ra, Bộ Tài chính cũng đã áp dụng chữ ký số vào kê khai, nộp thuế trực tuyến qua mạng Internet và các thủ tục hải quan điện tử như khai báo hải quan và thông quan trực tuyến mà không phải in các tờ khai, đóng dấu đỏ của công ty và chạy đến cơ quan thuế xếp hàng và ngồi đợi để nộp tờ khai này. Trong tương lai, chữ ký số có thể sử dụng với các ứng dụng chính phủ điện tử bởi cơ quan nhà nước. Sắp tới, sẽ làm việc với người dân hoàn toàn thông qua các dịch vụ công trực tuyến và một cửa điện tử. Khi cần làm thủ tục hành chính hay một sự xác nhận của cơ quan nhà nước, người dân chỉ cần ngồi ở nhà khai vào mẫu đơn và ký số để gửi là xong.

Kết luận chương 2

Trong chương này, luận văn đã trình bày về chữ ký số RSA,ELGAMAL,DSS. Các vấn đề về tấn công chữ ký số để từ đó đưa ra các giải pháp phòng tránh thích hợp.

Chương 3. XÂY DỰNG THƯ VIỆN TÍNH TOÁN SỐ LỚN

3.1. Biểu diễn số lớn [7]

Có nhiều cách để biểu diễn và lưu trữ số lớn. Cách thường dùng nhất là biểu diễn bằng xâu ký tự. Cho một số lớn có n chữ số thập phân được biểu diễn trong hệ cơ số b, có dạng $a = (a_{n-1}a_{n-2}\ldots a_0)_b$ ta sẽ sử dụng một xâu ký tự s có độ dài là n ký tự để biểu diễn giá trị a theo cách:

Chữ số a_0 được lưu vào phần tử s[0]

Chữ số a_1 được lưu vào phần tử s[1]

…….

Chữ số a_{n-1} được lưu vào phần tử s[n-1]

Dấu của số lớn được đặt trong biến trạng thái "dau":

- Nếu dau = 1 thì a là số dương
- Nếu dau = -1 thì a là số âm

Ta quy ước, khi nói đến số lớn a thì a là xâu ký tự, các phần tử của xâu chính là các phần tử của số lớn được biểu diễn ở hệ cơ số b một cách tương ứng. Giả sử, ta đang biểu diễn số lớn ở hệ cơ số c nào đó và ta muốn chuyển số lớn sang biểu diễn ở hệ cơ số b thì sẽ thông qua thuật toán sau:

Input: số nguyên dương a, số nguyên dương b ($2 \le b \le 256$)

Output: biểu diễn ở hệ cơ số b của $a = (a_{n-1}a_{n-2}\ldots a_0)_b$, $n \ge 0$; $a_n \ne 0$

Thuật toán:

(1) i = 0; x = A; $q = \left\lfloor \frac{x}{b} \right\rfloor$; $a_i = x - q.b$;

(2) while (q > 0)

$i = i + 1$; $x = q$; $q = \left\lfloor \frac{x}{b} \right\rfloor$; $a_i = x - p.q$;

(3) return $(a_i\ldots a_0)$

3.2. Các phép toán trong số lớn [7]

Ta quy ước, số lớn được biểu diễn trong hệ cơ số b = 10.

3.2.1. So sánh hai số lớn

Input: Hai số lớn: $x = (x_{n-1}\ldots x_0)$ và $y = (y_{m-1}\ldots y_0)$ có độ dài lần lượt là n và m

Output: - Nếu x > y thì return 1

- Nếu x < y thì return -1

- Nếu x = y thì return 0

Method:

(1) If (x dương, y âm) return 1;

(2) If (x âm, y dương) return -1;

(3) If (n > m) && (x âm) return -1; // x < y

(4) If (n < m) && (y âm) return 1; // x > y

(5) If (n > m) && (x dương) return 1; // x > y

(6) If (n < *m)* && (y dương) return -1; // x < y

(7) If (n == m)

(7.l) If (x dương)

For (i = n - 1; i ≥ 0; i--)

If (x[i] > y[i]) return 1; // x > y

Else return 0;

(7.2) If (x âm)

For (i = n - 1; i ≥ 0; i--)

If (x[i] > y[i]) return -1; // x < y

Else If (x[i] < y[i]) return 1; // x > y

(8) return 0;

End.

3.2.2. Cộng hai số dương lớn

Cho x và y là hai số lớn, có độ dài lần lượt là n và m. Cộng từng phần tử của hai xâu x và y lại, sau đó lưu vào từng phần tử trong xâu z.

Input: Hai số lớn x = ($x_{n-1}…x_0$), y = ($y_{m-1}…y_0$) có độ dài là n và m

Output: z = x + y

Method:

(1) temp = 0, nho = 0; // nho: biến lưu giá trị nhớ của phép cộng

(2) If (n < m) temp = n; else temp = m;

(3) For (i = 0; i < temp; i++)

z[i] = x[i] + y[i] + nho;

if (z[i] > 9) { z[i] = z[i] - 10; nho = l } else nho = 0;

End.

3.2.3. Trừ hai số dương lớn

Cho x, y là hai số lớn, có độ dài lần lượt là n và m. Trừ từng phần tử của hai xâu x và y rồi lưu vào từng phần tử trong xâu z.

Input: Hai số lớn x = (x_{n-1}...x_0), y = (y_{m-1}...y_0) có độ dài là n và m

Output: z = x – y

Method:

(1) nho = 0; // nho: biến lưu giá trị nhớ của phép trừ

(2) for (i = 0; i < n; i++)

(3) if (n > m) y[i] = 0; // chèn 0 vào phần tử y nếu x có độ dài lớn hơn y

(4) if (x[i] < y[i] + nho)

```
        z[i] = x[i] + 10 - y[i] - nho;
        nho = 1;
else z[i] = x[i] - y[i] - 10;
```

(5) nho = 0;

(6) xét dấu cho z;

End.

3.2.4. Nhân hai số lớn

1/. Nhân số lớn với một số nguyên

Input: Số lớn x = (x_{n-1}…x_0) và số nguyên k

Output: z = x*k, trong đó: z có độ dài tối thiểu bằng n

Method:

(1) nho = 0; temp = 0;

(2) for (i = 0; i < n; i++)

(3) temp = x[i]*k + nho;

(4) z[i] = temp % 10;

(5) nho = (temp - z[i])/10;

(6) độ dài của z 1à n;

(7) if (nho != 0)

(8) while (nho != 0)

(9) n = n+l; // Tăng độ dài của z

(10) z[i] = nho % 10;

(11) nho = (nho - z[i])/10;

(12) i++;

(13) return z;

End.

2/. Nhân hai số lớn với nhau

Cho hai số lớn: x = (x_{n-1}...x_0) và y = (y_{m-1}...y_0). Cần tính z = x.y có độ dài là (n+m).

Để nhân hai số lớn ta tiến hành như sau:

- Lấy phần tử của số hạng thứ hai nhân với tất cả các phần tử của số hạng thứ nhất hay nói cách khác lấy từng phần tử của y nhân với toàn bộ x cộng thêm một phần tử nhớ, được kết quả, đem chia cho hệ cơ số, lấy số dư làm kết quả của hàng tương ứng, thương số là số nhớ của số mới.

- Dịch trái một số bước phù hợp.

- Cộng tất cả các kết quả nhân lại.

Cụ thể thuật toán nhân hai số lớn như sau:

Input: Hai số lớn x = (x_{n-1}...x_0) và y = (y_{m-1}...y_0) có độ dài là n và m.

Output: z = x * y

Method:

(1) i = 0, temp = 0, nho = 0;

(2) BigNumber b;

(3) for (i = 0; i < m; i++)

(4) temp = y[i];

(5) b = x * temp; // Nhân một số lớn với một số nguyên

(6) dichtrai (b,i); // dịch trái giá trị b đi *i* vị trí;

(7) z = z + b;

(8) return z;

End.

3.2.5. Phép chia hai số lớn dương

Cho hai số lớn x = (x_{n-1}...x_0) và y = (y_{m-1}...y_0) có độ dài là n và m, ta xét hai trường hợp sau:

1/. Phép chia hai số lớn có thương ≤ 9

Input: Hai số lớn x = (x_{n-1}...x_0) và y = (y_{m-1}...y_0) có độ dài là n và m

Output: z = $\frac{x}{y}$ trong đó z có độ dài là k = n - m;

Method:

(1) For i = 0 to 10 do

(2) BigNumber c;

(3) c = b * i; // Nhân một số lớn với một số nguyên

(4) If (Sosanh(c,a)==1) return i-1; // c > a

Else If (Sosanh(c,a)==0) return i; // c = a

End.

2/. Chia hai số lớn

Thuật toán chia hai số lớn dựa trên ý tưởng của phép chia thông thường.

Input: Hai số lớn x, y có độ đài n, m

Output: z = $\frac{x}{y}$

Method:

(1) BigNumber b, c, z;

(2) If ((y[0]==0) && (m = 1))

printf " Loi chia cho 0";

return b;

(3) If (x < y)

z = 0;

Độ dài của b là 1;

dau = l; // b dương

return z;

(4) t = 0, i = 0;

(5) for j = (n - 1) downto j ≥ 0 // đảo ngược

b[0] = x[j];

if (j == n - 1) then c = c + b; else Dich(c,1); // Dịch c sang trái 1 vị trí

c = c + b;

Chuan(c); // loại bỏ các số 0 (*dạng 00012 thành 12*)

if (Sosanh(c,y) != -1) {

t = Chia(c,y); // chia hai số lớn có thương 9 ≥ t

b = y * t;

z[i] = t;

```
                    i++;
                    c = c - b;
                    Chuan(c); }
        if (i != 0) then
                    z[i]=0;
                    i++;
```

(6) Độ dài của z là i;

(7) Giá trị dấu của z = (giá trị dấu của x) * (giá trị dấu của y)

(8) z = Dao(z); // đảo ngược lại số z;

(9) return z;

End.

3.2.6. Lũy thừa

Input: a, $k \in Z_n$

Output: a^k (mod n)

Method:

(1) Đưa k về dạng: $k = \sum_{i=0}^{i} k_i 2^i$ đây là dạng biểu diễn trong hệ cơ số 2 của k.

(2) Xét b = 1 và nếu k = 0 thì b là kết quả

(3) Xét A = a

(4) Nếu $k_0 = 1$ thì b = a

(5) For (i = 1; i < k; i++)

(5.1) A = A.A (mod n)

(5.2) Nếu k = 1 thì b = A.b (mod n)

(6) return b;

End.

3.2.7. Ước chung lớn nhất

Sử dụng thuật toán Euclid mở rộng tìm ước chung lớn nhất của 2 số.

Input: Hai số lớn a, b với a > b

Output: d = UCLN(a,b) và hai số nguyên x, y thỏa mãn a.x + b.y = d

Method:

(1) if (b==0) then { d = a; x = 1; y = 0; return (d, x, y); }

(2) $a_1 = 1$; $a_2 = 0$; $a_3 = a$; $b_1 = 0$; $b_2 = 1$; $b_3 = b$;

(3) q = a_3 div b_3;

(4) While (b_3 != 0)

(4.1) $c_1 = a_1$; $c_2 = a_2$; $c_3 = a_3$;

(4.2) $a_1 = b_1$; $a_2 = b_2$; $a_3 = b_3$;

(4.3) $b_1 = c_1 - q.b_1$; $b_2 = c_2 - q.b_2$; $b_3 = c_3 - q.b_3$;

(5) if ($a_2 < 0$) then $a_2 = a_2 + a$;

(6) $d = a_2$; $x = a_1$; $y = a_3$;

(7) return (d, x, y);

End.

3.2.8. Phép nhân theo modulo p

Để tính *a.b (mod p)*, ta biểu diễn b dưới dạng nhị phân $b = \sum_{j=0}^{k} 2^i.b_j$ trong đó $b_j = 0$ hoặc $b_j = 1$.

Áp dụng thuật toán Horner: *a.b (mod p)* = $\sum_{j=0}^{k} (2^i.b_j \bmod p) \bmod p$

Thuật toán tính *a.b (mod p)* như sau:

Input: Ba số lớn: a, b, p

Output: z = a.b (mod p)

Method:

(1) $b = \sum_{j=0}^{k} 2^i.b_j$

(2) z = a;

(3) For (i = k - 1; i ≥ 0; i--)

$z = z^2 \bmod p$;

if ($b_j = 1$) then z = z + a;

(4) return z;

End.

3.2.9. Tìm phần tử nghịch đảo theo modulo p

Để tính giá trị nghịch đảo, chúng ta sẽ sử dụng thuật toán Euclid mở rộng.

Input: $a \in Z_n$ với UCLN(a,n) = 1

Output: a^{-1} (mod n)

Method:

(1) Sử dụng thuật toán Euclid mở rộng tìm x, y thỏa mãn điều kiện: ax + by = d

và d = UCLN(a,n).

(2) Nếu d > 1 thì a^{-1} (mod n) không tồn tại, ngược lại thì x là giá trị cần tìm.

End.

3.2.10. Phép cộng có dấu

Phép cộng hai số có dấu được thực hiện dựa trên phép so sánh, phép cộng và phép trừ hai số không âm đã trình bày. Dấu của số lớn được lưu tại bít cao nhất của số lớn.

Input: Hai số lớn x, y

Output: z = x + y

Method:

```
(1) if (x, y cùng dấu)
        z = x + y;
        Đặt z cùng dấu với x;
        Return z;
(2) if (x dương) // và y âm
    (2.1) if ((s = Sosanh(x,y)) != -1)
        z = x - y;
        Đặt z cùng dấu với x;
        Return z;
    (2.2) if (s == (-1))
        z = y - x;
        Đặt z cùng dấu với y;
        Return z;
(3) if (x âm) // và y dương
    (3.1) if ((s = Sosanh (x,y))== -1)
        z = y - x;
        Đặt z cùng dấu với y;
        Return z;
    (3.2) if (s != (-1))
        z = x - y;
        Đặt z cùng dấu với x;
        Return z;
```

End.

3.2.11. Phép trừ có dấu

Phép trừ hai số có dấu được thực hiện dựa trên phép cộng hai số đó.

Input: Hai số lớn x, y

Output: z = x - y

Method:

(1) Đổi dấu của y;

(2) Cộng có dấu z = x + y;

(3) Return z;

End.

3.2.12. Phép nhân có dấu

Phép nhân hai số có dấu được thực hiện dựa trên phép nhân hai số không âm đã được trình bày ở trên.

Input: Hai số lớn x, y

Output: z = x * y

Method:

(1) if (x, y cùng dấu) Return z = x.y;

(2) if (x, y khác dấu)

z = x.y;

Đổi dấu z;

Return z;

End.

Kết luận chương 3

Trong chương này, luận văn đã trình bày các vấn đề về tính toán với số lớn. Biểu diễn và cài đặt các phép toán quan trọng trên số nguyên lớn phục vụ cho việc xây dựng các giải pháp tấn công chữ ký số.

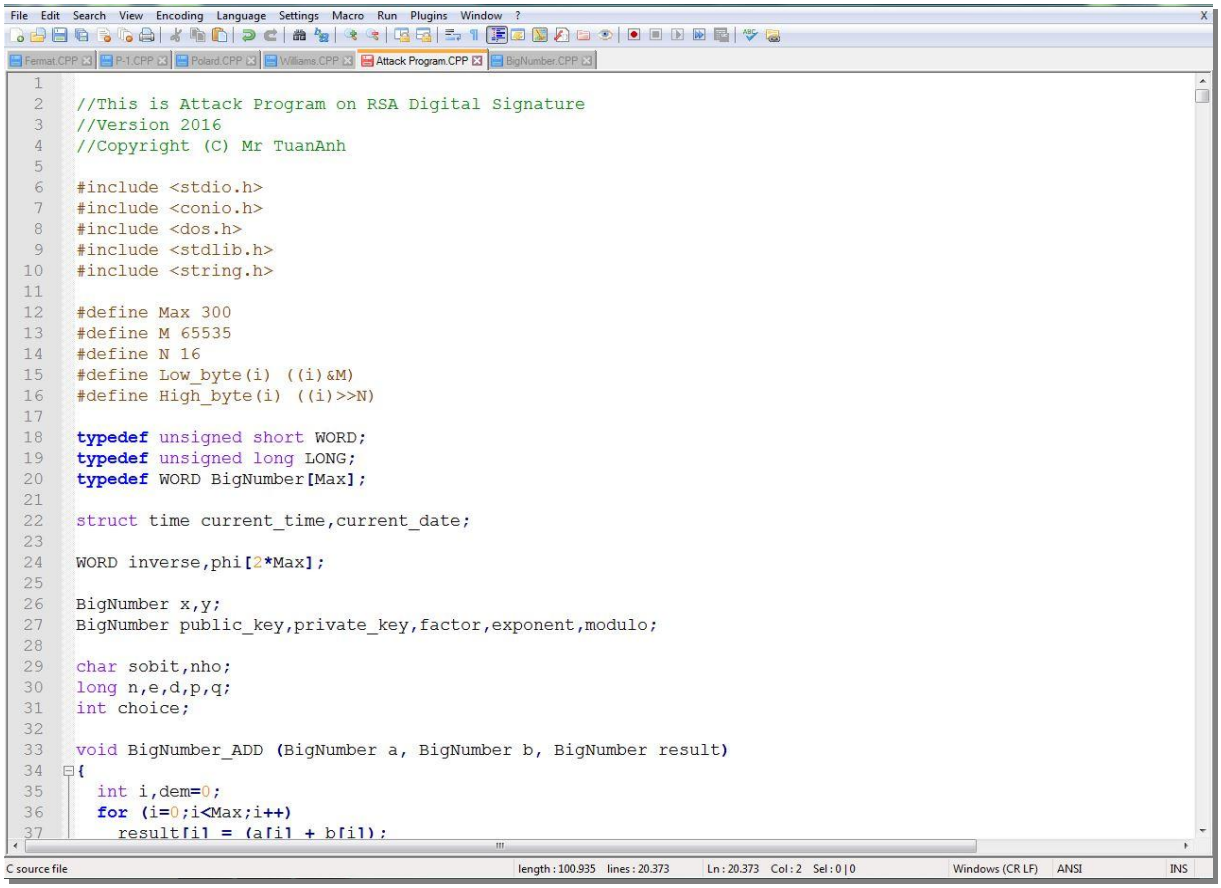
Chương 4. THỬ NGHIỆM CHƯƠNG TRÌNH TẤN CÔNG

Trong chương này, luận văn trình bày về thực nghiệm chương trình tấn công. Giải pháp được lựa chọn ở đây là tấn công chữ ký số RSA ở dạng xác định khóa bí mật dựa vào khóa công khai n và e, sử dụng phương pháp nhân tử hóa giá trị modulo n.

4.1 Chương trình thực nghiệm

Bảng 4.1 Thông tin về chương trình thực nghiệm

Môi trường thực nghiệm	- Processor: Intel® Core™ i7-6950X Extreme Edition (25M Cache, 3.50 GHz) - Memory (Ram): 32GB – DDR4 - SSD: 500 GB – Sata III - Operating System : Windows 10 Pro - System type: 64– bit Operating System, x64 – base processor
Ngôn ngữ sử dụng	Ngôn ngữ lập trình C
Thư viện tính toán	Thư viện *BigNumber* trong chương 3

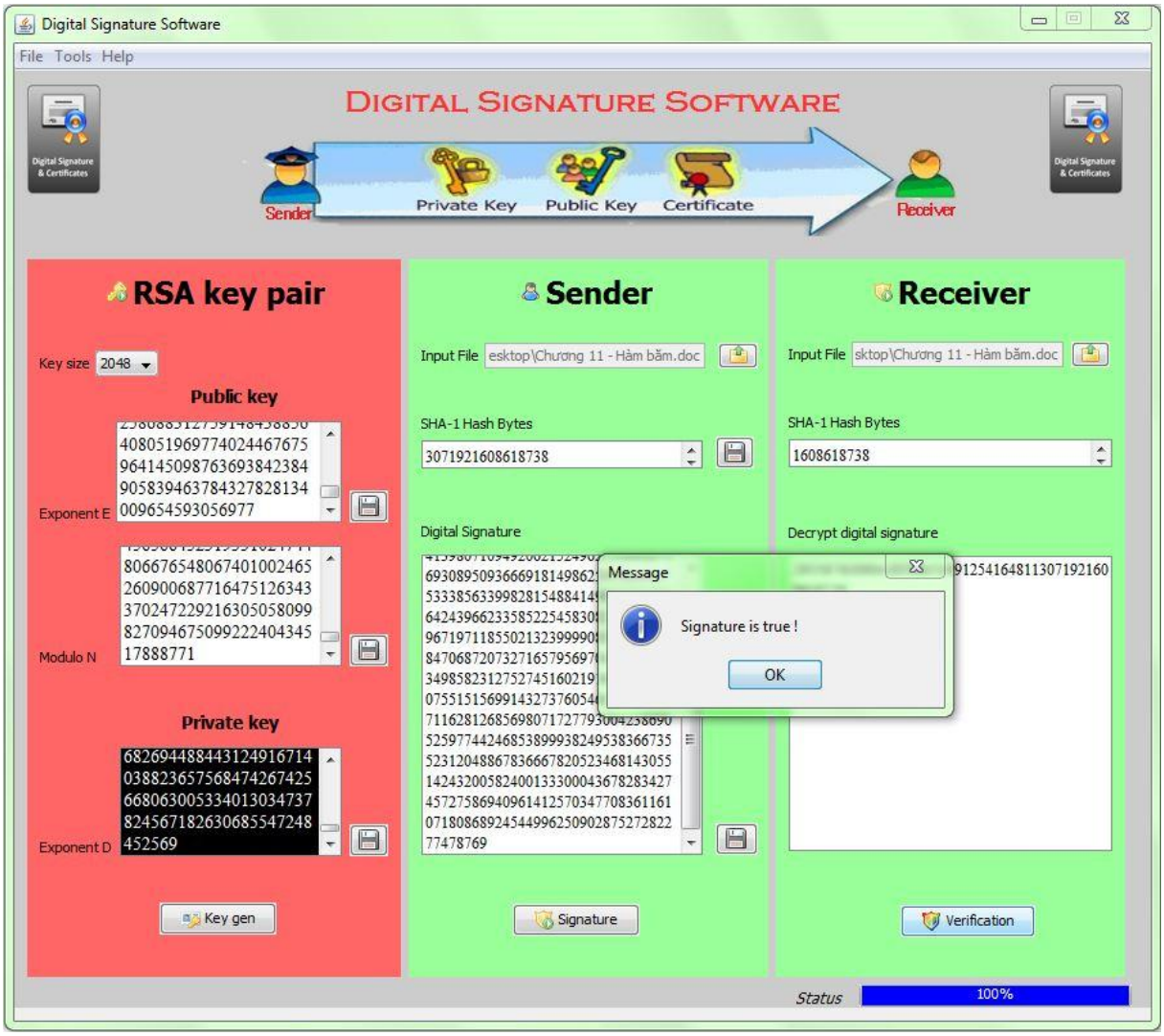


Hình 4.1 Chương trình thực nghiệm

Chương trình thực nghiệm được viết bằng ngôn ngữ lập trình C dưới dạng giao diện dòng lệnh, cài đặt cho 4 thuật toán: Pollard, P-1, Williams và thuật toán tìm nhân tử lớn nhất thứ nhất sử dụng định lý Fermat. Chức năng chính của chương trình là đọc dữ liệu từ hai khóa công khai là: *modulo n* và *exponent e*, sau đó chạy các thuật toán để nhân tử hóa giá trị *modulo n* thành 2 phần tử nguyên tố *p* và *q*. Tiếp theo, tính toán giá trị $\phi(n) = (p-1).(q-1)$. Cuối cùng chạy thuật toán Euclid để xác định phần tử nghịch đảo của *exponent e* trong không gian modulo $\phi(n)$ vừa tìm được. Giá trị tìm được cuối cùng chính là khóa bí mật.

4.2 Dữ liệu thực nghiệm

Tập dữ liệu thực nghiệm là các khóa công khai được cung cấp bởi phần mềm tạo chữ ký số "Digital Signature Software" và phần mềm "Des & RSA Encryption" bao gồm nhiều kích thước khóa lớn, nhỏ khác nhau.



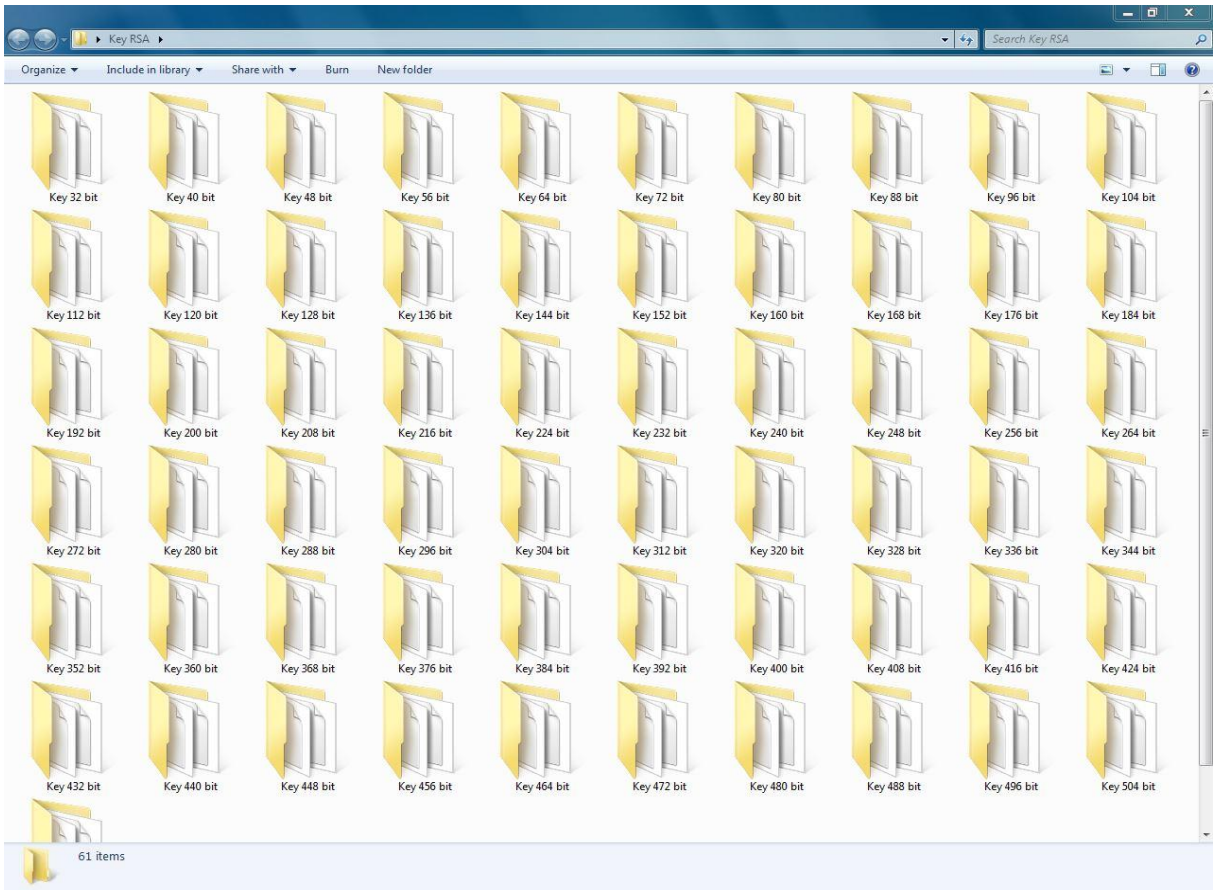
Hình 4.2 Phần mềm tạo chữ ký số RSA

Hình 4.3 Phần mềm mã hóa dữ liệu

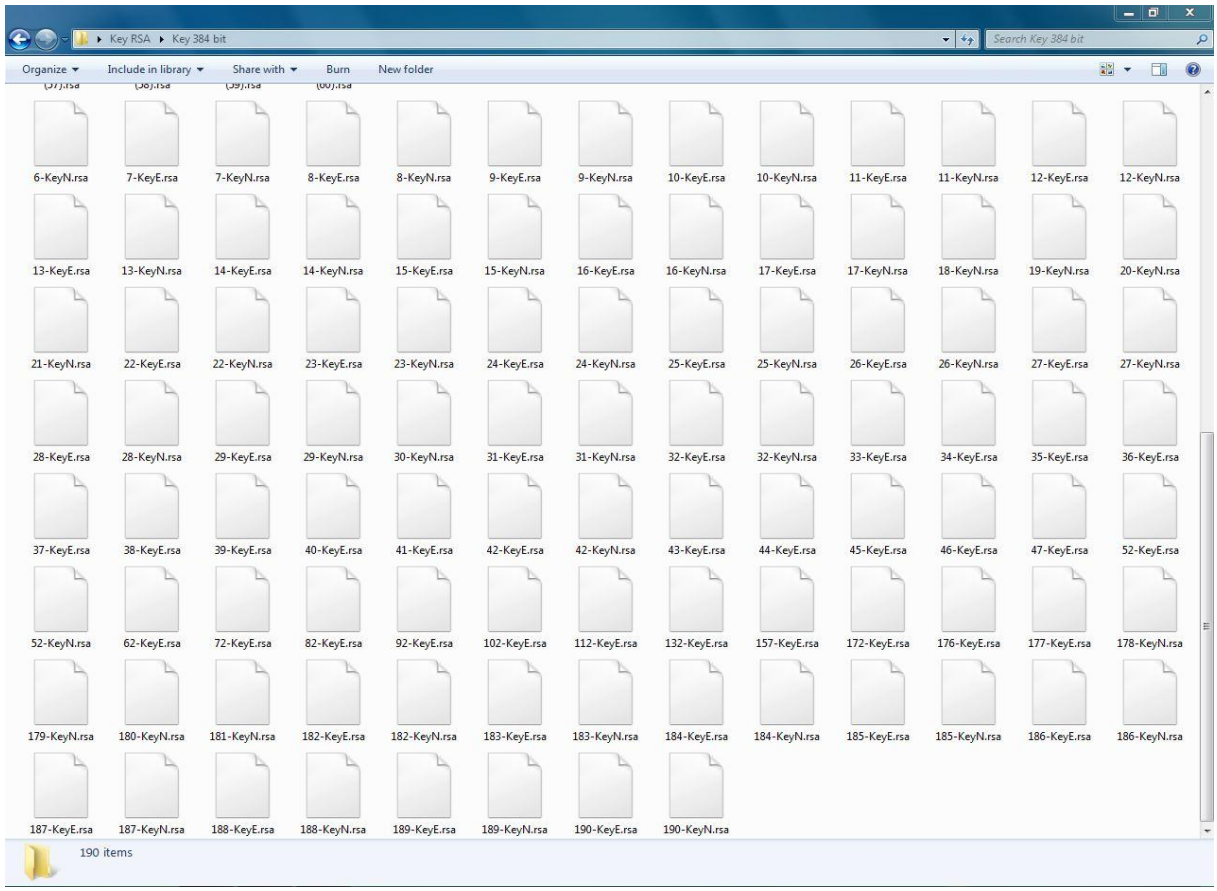
Bộ dữ liệu thực nghiệm được mô tả trong bảng sau:

Bảng 4.2 Bảng mô tả tập dữ liệu thực nghiệm

Key size (*bit*)	32	40	48	56	64	72	80	88	96	104
Số lượng mẫu	*320*	*340*	*336*	*450*	*390*	*429*	*370*	*360*	*450*	*350*
Key size	112	120	128	136	144	152	160	168	176	184
Số lượng mẫu	*210*	*320*	*320*	*150*	*145*	*217*	*180*	*280*	*160*	*254*
Key size	192	200	208	216	224	232	240	248	256	264
Số lượng mẫu	*210*	*200*	*170*	*190*	*190*	*149*	*170*	*160*	*155*	*140*
Key size	272	280	288	296	304	312	320	328	336	344
Số lượng mẫu	*110*	*120*	*130*	*150*	*140*	*137*	*100*	*180*	*160*	*154*
Key size	352	360	368	376	384	392	400	408	416	424
Số lượng mẫu	*110*	*120*	*129*	*150*	*190*	*146*	*170*	*160*	*150*	*150*
Key size	432	440	448	456	464	472	480	488	496	512
Số lượng mẫu	*130*	*120*	*137*	*150*	*140*	*147*	*145*	*124*	*129*	*110*



Hình 4.4 Thư mục chứa khóa công khai



Hình 4.5 Tệp dữ liệu khóa công khai

4.3 Tấn công thử nghiệm

Hình 4.6 Giao diện của chương trình tấn công

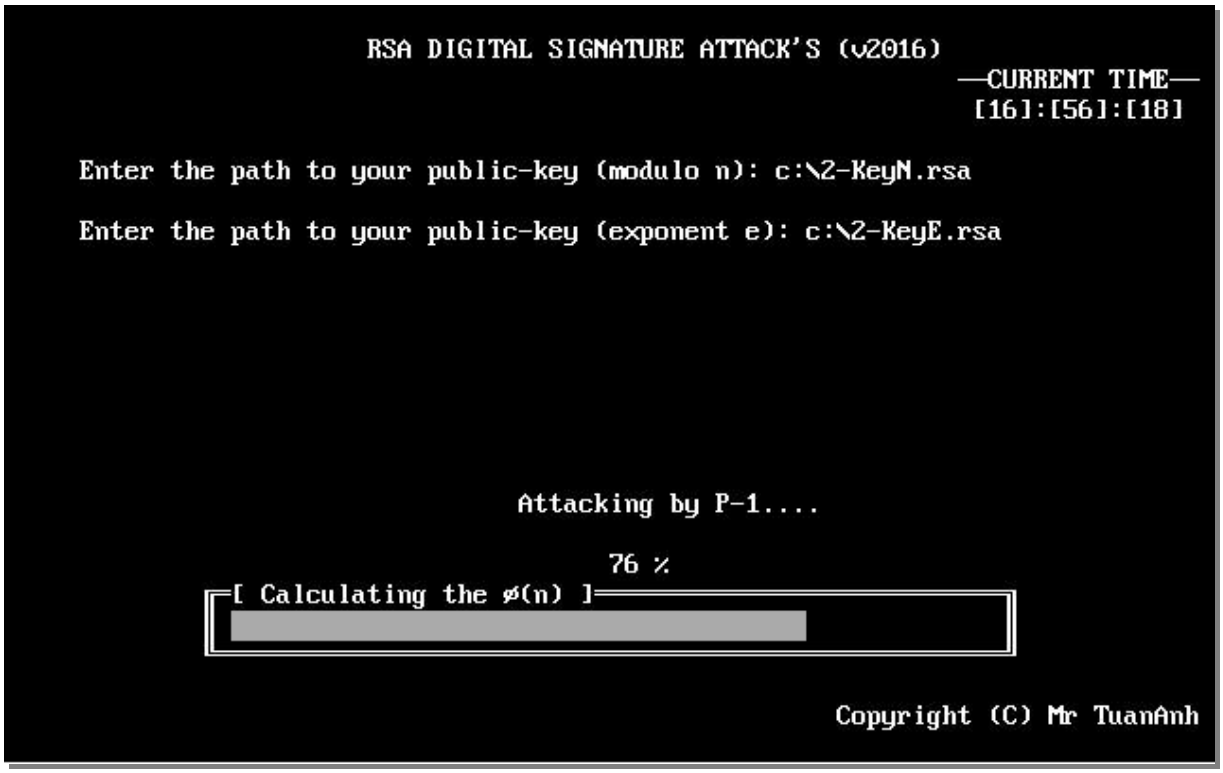
Với tập dữ liệu được mô tả như trong bảng 4.2, tôi đã tiến hành tấn công thực nghiệm. Sau đây là một vài hình ảnh tiêu biểu về cuộc tấn công này:

- Kích thước khóa: *256 bit*
- Sử dụng thuật toán: *Pollard*

Hình 4.7 Tấn công bằng thuật toán Pollard

Hình 4.8 Kết quả tấn công bằng thuật toán Pollard

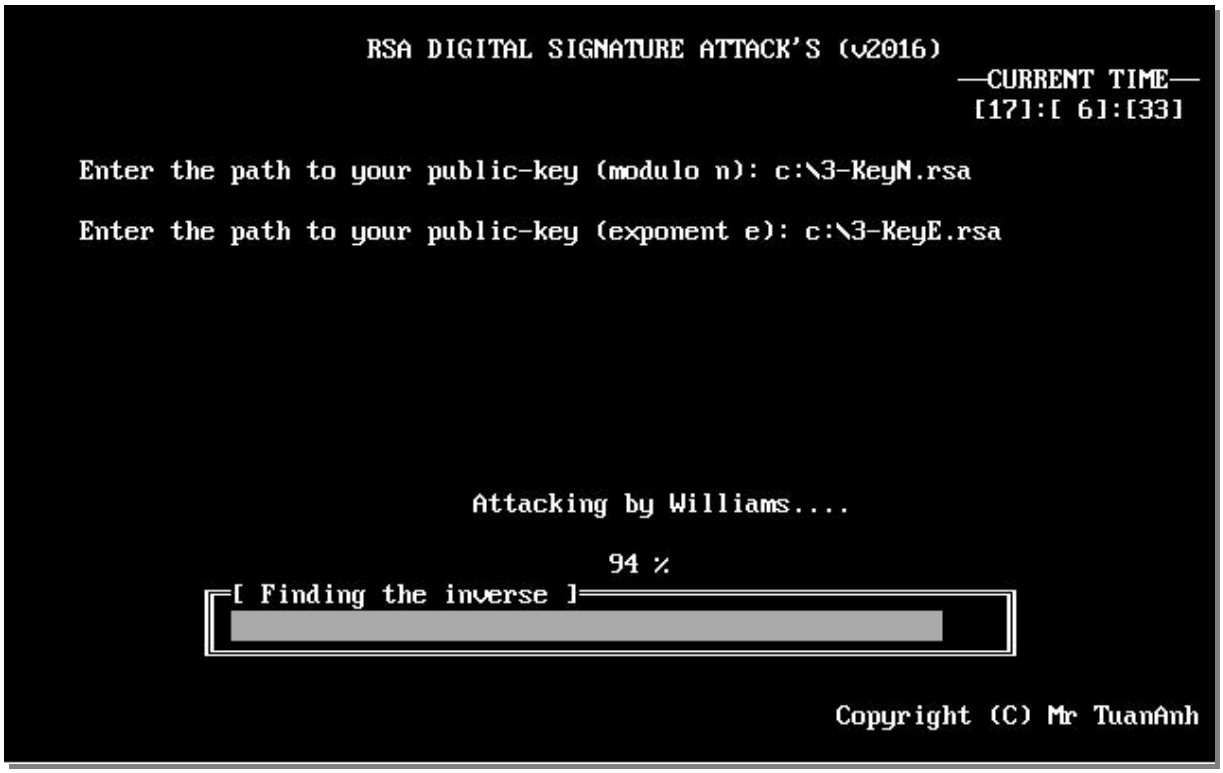
- Kích thước khóa: *320 bit*
- Sử dụng thuật toán: *P-1*



Hình 4.9 Tấn công bằng thuật toán P-1

Hình 4.10 Kết quả tấn công bằng thuật toán P-1

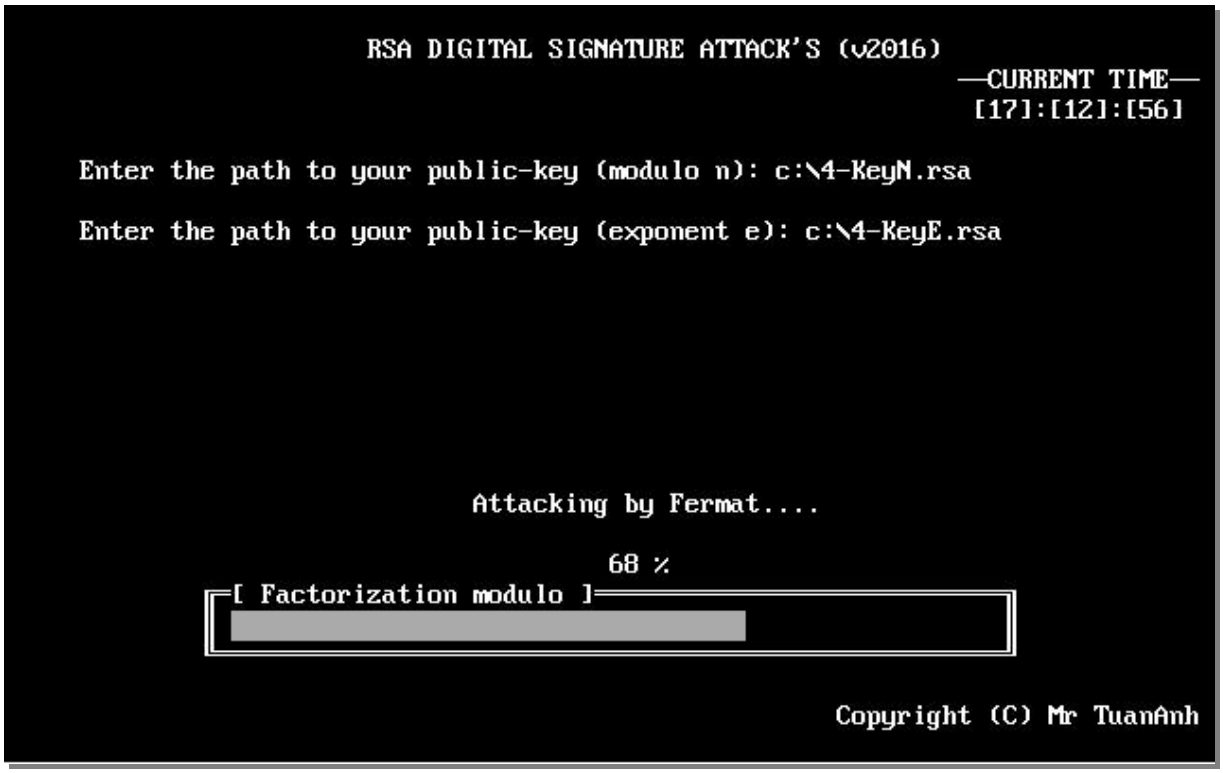
- Kích thước khóa: *352 bit*
- Sử dụng thuật toán: *Williams*



Hình 4.11 Tấn công bằng thuật toán Williams

Hình 4.12 Kết quả tấn công bằng thuật toán Williams

- Kích thước khóa: *384 bit*
- Sử dụng thuật toán: *Fermat*



Hình 4.13 Tấn công bằng thuật toán Fermat

Hình 4.14 Kết quả tấn công bằng thuật toán Fermat

Chú thích:

- *Kích thước khóa*: kích cỡ của giá trị *modulo n*. Đơn vị: *bit*.
- *KeyN.rsa*: File chứa giá trị khóa công khai *modulo n*.
- *KeyE.rsa*: File chứa số mũ công khai *e*.
- *P và Q*: Hai thừa số nguyên tố của *modulo n* đã được nhân tử hóa bởi các thuật toán (Pollard, P-1, Williams, Fermat).
- *D*: Giá trị khóa bí mật tính được.

4.4 Nhận xét và thảo luận

Chương trình đã được thử nghiệm với nhiều bộ dữ liệu có kích thước khác nhau và cho kết quả chính xác. Đây là một minh chứng thực tế cho những gì mà lý thuyết đã trình bày về khả năng tấn công vào các sơ đồ chữ ký số. Càng ngày, tốc độ xử lý của máy tính ngày càng cao, chính điều này đã đe dọa đến sự an toàn của hệ thống chữ ký số, đồng thời, cùng với sự ra đời của máy tính lượng tử trong tương lai thì khả năng phá hủy hệ mật RSA là điều hoàn toàn có thể xảy ra.

Kết luận chương 4

Trong chương này, luận văn đã trình bày về chương trình thực nghiệm, các kết quả đạt được, đưa ra nhận xét và thảo luận về chương trình.

KẾT LUẬN

Ngày nay, ngành công nghệ thông tin 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 thông tin sẽ là một trong những yếu tố quan trọng, giúp đảm bảo an toàn cho việc ứng dụng vào trong thực tiễn, cho các giao dịch điện tử. Một trong những nhiệm vụ của đảm bảo an toàn thông tin là bảo vệ chữ ký, vì vậy, đề tài đã nghiên cứu về chữ ký số. Cụ thể là nghiên cứu các khả năng tấn công chữ ký, từ đó đưa ra các giải pháp phòng tránh thích hợp.

- Các kết quả chính đạt được của luận văn:
  a. Về lý thuyết:
     - Trình bày cơ sở lý thuyết của mật mã học: số nguyên tố, độ phức tạp của thuật toán, các bài toán quan trọng trong mật mã.
     - Trình bày về chữ ký số, các phương pháp tấn công, đưa ra các giải pháp phòng tránh thích hợp.
  b. Về thực nghiệm:
     - Xây dựng thư viện tính toán số nguyên lớn.
     - Cài đặt chương trình tấn công thử nghiệm. Tiến hành thực nghiệm và đánh giá kết quả.
- Hướng nghiên cứu tiếp theo:
  - Tìm hiểu các phương pháp tấn công mới. Cải tiến chương trình thực nghiệm và thuật toán tạo chữ ký số để tăng thêm mức độ bảo mật.

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1] PGS.TS Trịnh Nhật Tiến (2008), "*Giáo trình An toàn dữ liệu"*, Nhà xuất bản Đại học Quốc Gia Hà Nội.

[2] Nguyễn Văn Tảo, Hà Thị Thanh, Nguyễn Lan Oanh (2009), *"Bài giảng An toàn và bảo mật thông tin"*, Trường Đại học Công nghệ thông tin và Truyền thông.

[3] Nguyễn Hữu Tuân (2008), *"Giáo trình An toàn và bảo mật thông tin"*, Trường Đại học Hàng hải.

[4] GS. Phan Đình Diệu (2002), *"Lý thuyết mật mã và an toàn thông tin"*, Nhà xuất bản Đại học Quốc Gia Hà Nội.

[5] Lương Văn Quyên (2013), *"Nghiên cứu khả năng ứng dụng của hệ mật trên bài toán logarit rời rạc trong chữ ký số"*, luận văn thạc sĩ, Học viện Công nghệ bưu chính viễn thông.

[6] Trần Xuân Phương (2015), *"Xác thực điện tử và ứng dụng trong giao dịch hành chính"*, luận văn thạc sĩ, Trường Đại học Công nghệ - ĐHQGHN.

[7] Bùi Tuấn Anh (2009), *"Các phương pháp tấn công RSA"*, khóa luận tốt nghiệp Trường Đại học Công nghệ - ĐHQGHN.

[8] Lê Thị Thu Trang (2009), *"Nghiên cứu một số loại tấn công chữ ký số"*, khóa luận tốt nghiệp Trường Đại học dân lập Hải Phòng.

Tiếng Anh

[9] Douglas R. Stinson (2006), *Cryptography theory and practice 3rd*.

[10] Abderrahmane Nitaj (2008), *A new attack on RSA and CRT-RSA*.

[11] L.Hernández Encinas, J. Munoz Masqué, A. Queiruga Dios (2000), *An algorithm to ontain an RSA modulus with a large private key*.

[12] Seema Verma, Deepak Garg (2014), *An improved RSA Variant*.

Internet

[13] https://primes.utm.edu/largest.html

[14] http://fit.mta.edu.vn/files/FileMonHoc/Chuong%205%20-%20C%C3%A1c%20h%E1%BB%87%20m%E1%BA%ADt%20kh%C3%B3a%20c%C3%B4ng%20khai.doc