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

TÓM TẮT LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN

Hà Nội - 2016

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 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.

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.

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.

2/. Các tính chất [1]

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.

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 $a^p \equiv a \pmod{p}$.

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

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*.

1.2. Một số khái niệm trong đại số

1.2.1. Cấu trúc nhóm [1]

1/. Khái niệm:

2/. Nhóm con của nhóm (G,*)

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.

2/. Cấp của nhóm Cyclic

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]

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

a). Định nghĩa: Cho $a \in Z_n$, nếu tồn tại $b \in Z_n$ sao cho $a.b \equiv 1 (\text{mod } n)$, 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.

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 $x = log_g{}^{\beta} (\text{mod } p)$ với ẩn *x*. Hay phải tìm số *x* duy nhất sao cho: $g^x \equiv \beta (\text{mod } p)$.

4/. Thặng dư bậc hai [5]

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

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

3/. Độ phức tạp về thời gian

$T_A(n) = \max\{t_A(e)$, 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.*

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.

2/. Khái niệm "khó tương đương"

Bài toán A gọi là "khó tương đương" bài toán B, ký hiệu $A \sim B$, 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.

4/. Lớp bài toán NP- Hard

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

5/. Lớp bài toán NP - Complete

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

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*".

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*".

1.4. Các bài toán quan trọng trong mật mã

1.4.1. Bài toán kiểm tra số nguyên tố lớn

1/. Một số ký hiệu toán học [3]

a). Ký hiệu Lagrăng

b). Ký hiệu Jacobi

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 = $a^{(p-1)/2}$ mod 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%.

b). Thuật toán Miller-Rabin [6]

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,…,n-1}
3. Đặt b = a^m mod n
4. Nếu b $\equiv$ 1 mod n thì đây là số nguyên tố. Kết thúc.
5. Cho k chạy từ 0 đến s-1:
   - 5.1 Nếu b $\equiv$ -1 mod n thì đây là số nguyên tố. Kết thúc.
   - 5.2 Thay b:= b^2 mod n
6. Kết luận đây là hợp số. Kết thúc.

c). Thuật toán Lehmann [3]

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

1.4.2. Bài toán phân tích thành thừa số nguyên tố

1/. Thuật toán sàng Eratosthenes [2]

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)

3/. Thuật toán Pollard [2]

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)

4/. Thuật toán p-1 [2]

* *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.

5/. Thuật toán p ± 1 [2]

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

1.4.3. Bài toán tính logarit rời rạc theo modulo

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 = $\left\lfloor \sqrt{p-1} \right\rfloor$

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

2/. Thuật toán Pohlig - Hellman [14]

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

(3.5) j = j +1

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]

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.

2.1.2. Phân loại "chữ ký số"

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

a). Chữ ký có thể khôi phục thông điệp gốc

b). Chữ ký không thể khôi phục thông điệp gốc

2/. Phân loại chữ ký theo mức an toàn

a). Chữ ký "không thể phủ nhận"

b). Chữ ký "một lần"

3/. Phân loại chữ ký theo ứng dụng đặc trưng

2.2. Chữ ký RSA

2.2.1. Sơ đồ chữ ký [1]

1/. Sơ đồ

2/. Ví dụ

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

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

Nếu trong quá trình lập khóa mà người sử dụng vô tình để lộ nhân tử p, q hoặc $\phi(n)$ 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, $\phi(n)$.

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]

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

b). Mã hóa trước, ký sau

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'.

⟶ *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

2/. Ví dụ

3/. Độ an toàn

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:

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

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

⟶ *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

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

2.4. Chữ ký DSS [1]

2.4.1. Sơ đồ chữ ký

1/. Giới thiệu

+ 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:

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

2/. Sơ đồ

3/. Ví dụ

2.4.2. Chú ý

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

2/. Liên quan chung tới DSS (1991)

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

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]

3.2.1. So sánh hai số lớn

3.2.2. Cộng hai số dương lớn

3.2.3. Trừ hai số dương lớn

3.2.4. Nhân hai số lớn

1/. Nhân số lớn với một số nguyên

2/. Nhân hai số lớn với nhau

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

2/. Chia hai số lớn

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

3.2.9. Tìm phần tử nghịch đảo theo modulo p

3.2.10. Phép cộng có dấu

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.

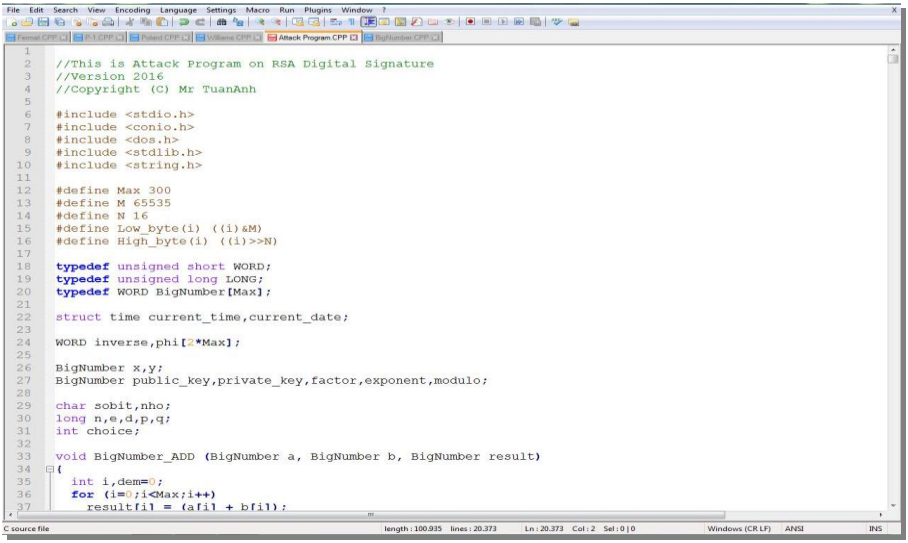
Chương 4. THỬ NGHIỆM CHƯƠNG TRÌNH TẤN CÔNG

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

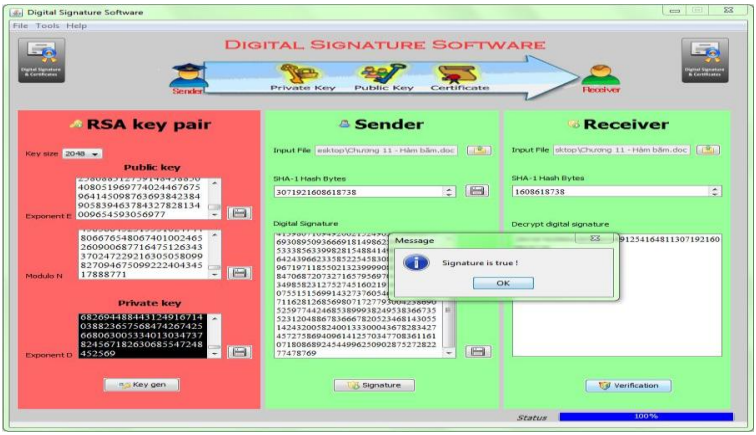


Hình 4.1 Chương trình thực nghiệm

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.



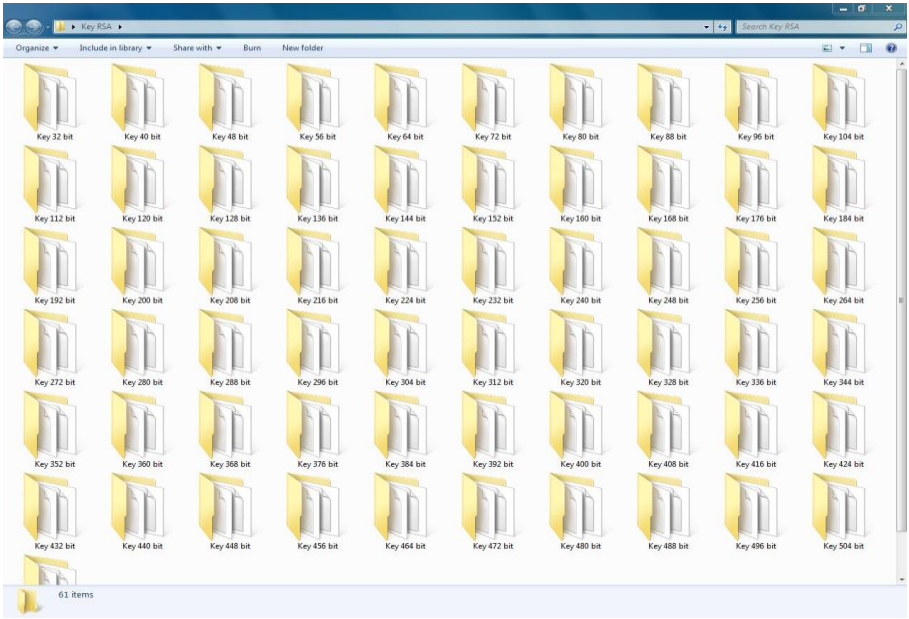
Hình 4.2 Phần mềm tạo chữ ký số RSA

Hình 4.3 Phần mềm mã hóa dữ liệu

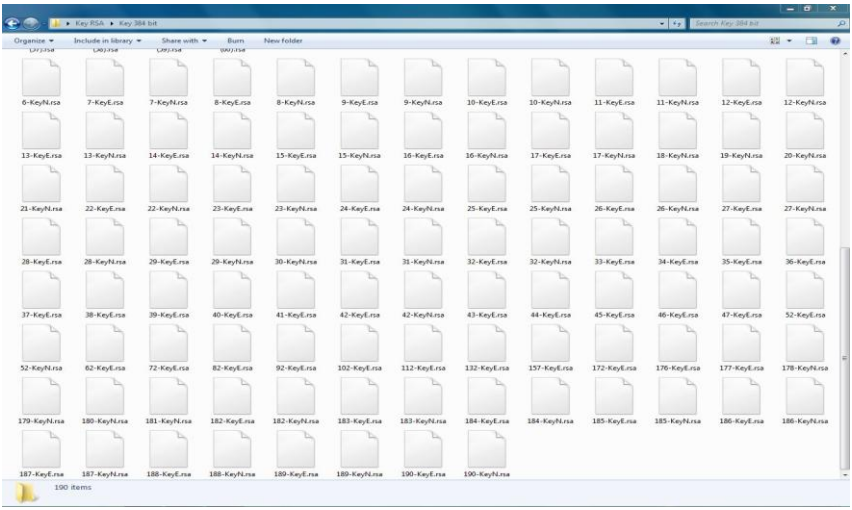
Bộ dữ liệu thực nghiệm được mô tả trong bảng sau:

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

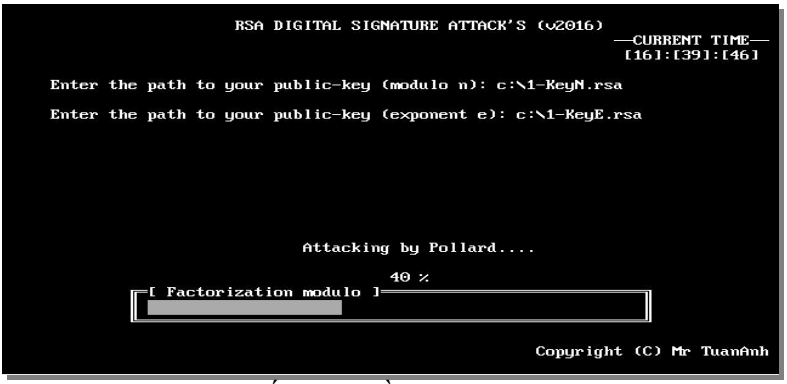
4.3 Tấn công thử nghiệm

```
                    RSA DIGITAL SIGNATURE ATTACK'S (v2016)

    This is a program RSA digital signature attacks. The system used four of
algorithms: Pollard, P-1, Williams and Fermat to factorization value modulo n
become two prime numbers p and q. Thereby, private-key (d) is calculated via
public-key (n, exponent).



    * Choose methods:

            1.  Attack by Pollard
            2.  Attack by P-1
            3.  Attack by Williams
            4.  Attack by Fermat
            5.  Exit

    - Enter choice: _

                                                    Copyright (C) Mr TuanAnh
```

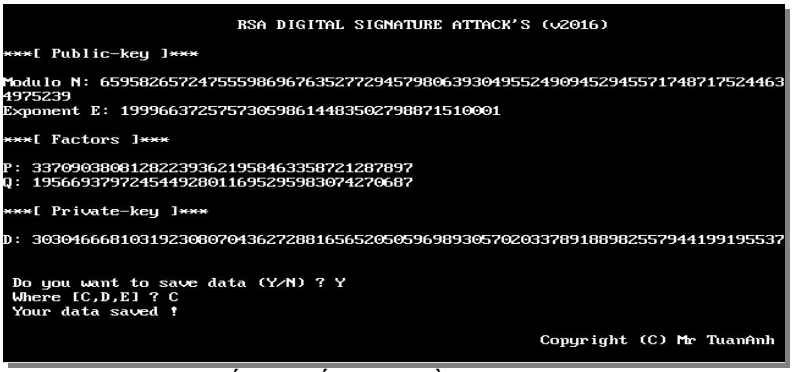
Hình 4.6 Giao diện của chương trình tấn công

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*

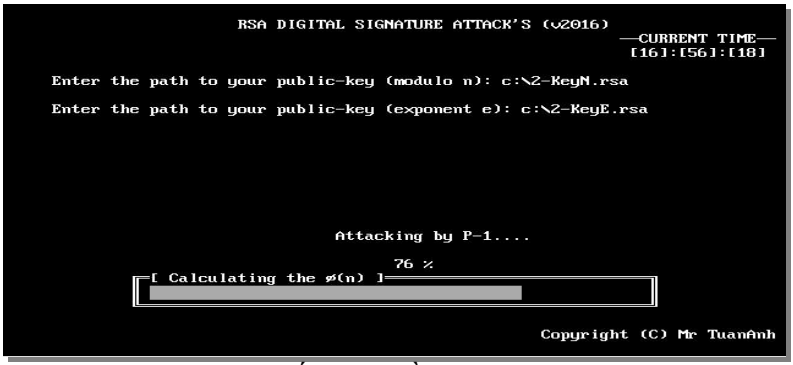


Hình 4.7 Tấn công bằng thuật toán Pollard



Hình 4.8 Kết quả tấn công bằng thuật toán Pollard

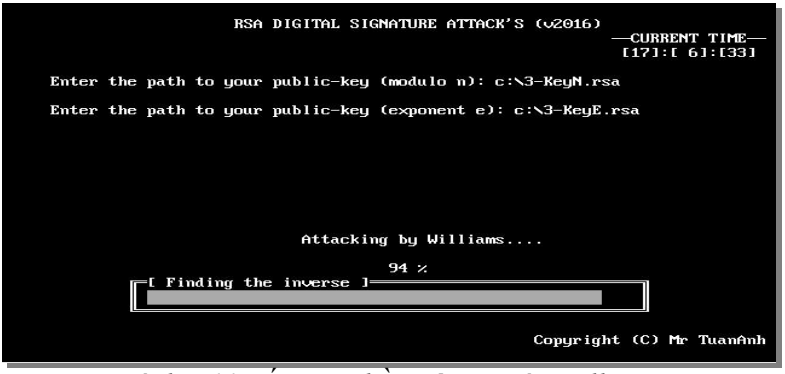
- Kích thước khóa: *320 bit*
- Sử dụng thuật toán: *P-1*



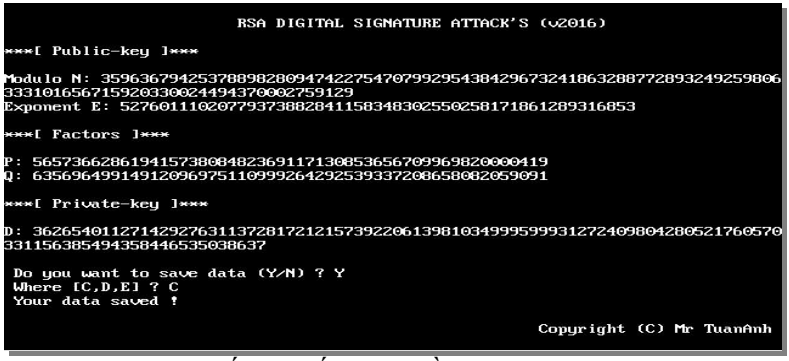
Hình 4.9 Tấn công bằng thuật toán P-1

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

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

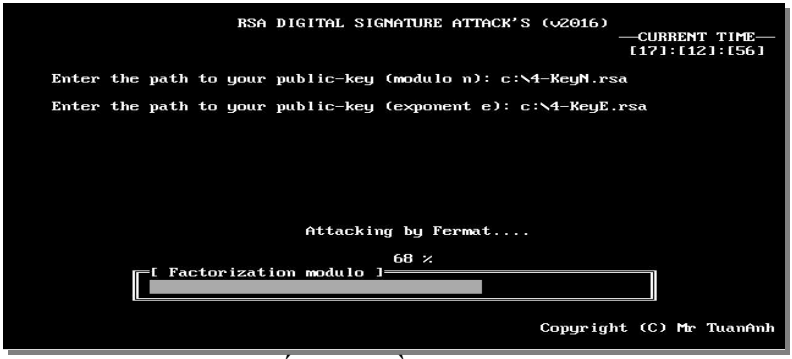


Hình 4.11 Tấn công bằng thuật toán Williams

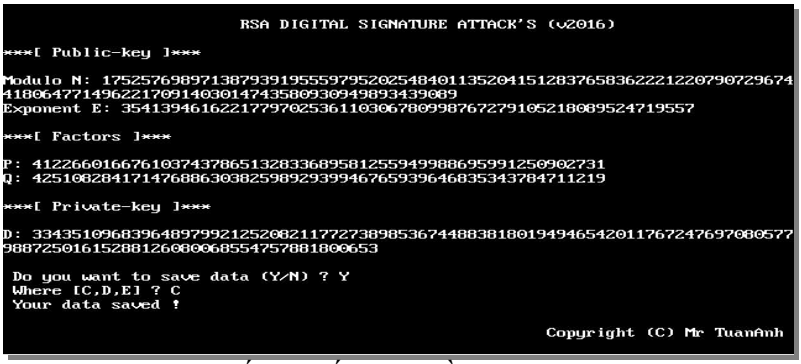


Hình 4.12 Kết quả tấn công bằng thuật toán Williams

- 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

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