BỘ GIÁO DỤC VÀ ĐÀO TẠO

TR- ỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG

-------o0o-------

ĐỒ ÁN TỐT NGHIỆP

NGÀNH CÔNG NGHỆ THÔNG TIN

HẢI PHÒNG 2010

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TR- ỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG
-------o0o-------

ĐỀ TÀI: MỘT SỐ DẠNG TẤN CÔNG HỆ THỐNG THÔNG TIN VÀ PHÒNG CHỐNG BẰNG KĨ THUẬT MẬT MÃ

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Công nghệ Thông tin

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TR- ỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG
-------o0o-------

ĐỀ TÀI: MỘT SỐ DẠNG TẤN CÔNG HỆ THỐNG THÔNG TIN VÀ PHÒNG CHỐNG BẰNG KĨ THUẬT MẬT MÃ

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Công nghệ Thông tin

Sinh viên thực hiện: Ngô Hồng Trang
Giáo viên h- ớng dẫn: PGS.TS. Trịnh Nhật Tiến
Mã số sinh viên: 100256

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TR- ỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆTNAM
Độc lập - Tự do - Hạnh phúc
-------*o0o*-------

NHIỆM VỤ THIẾT KẾ TỐT NGHIỆP

Sinh viên: Ngô Hồng Trang — Mã số: 100256
Lớp: CT1002 — Ngành: Công nghệ Thông tin

Tên đề tài:

Một số dạng tấn công hệ thống thông tin và phòng chống bằng kĩ thuật mật mã

NHIỆM VỤ ĐỀ TÀI

1. Nội dung và các yêu cầu cần giải quyết trong nhiệm vụ đề tài tốt nghiệp

a. Nội dung:

Một số dạng tấn công hệ thống thông tin và phòng chống bằng kỹ thuật mật mã .

b. Các yêu cầu cần giải quyết

* Tìm hiểu và nghiên cứu lý thuyết:

- Một số dạng tấn công hệ thống thông tin (thông qua mạng máy tính, hệ điều hành, cơ sở dữ liệu,….)
- Một số kĩ thuật mật mã.
- Nghiên cứu ph- ơng pháp phòng chống tấn công bằng kĩ thuật mật mã.

* Thử nghiệm Ch- ơng trình DEMO phòng chống tấn công.

2. Các số liệu cần thiết để thiết kế, tính toán

3. Địa điểm thực tập

CÁN BỘ H□ ỚNG DẪN ĐỀ TÀI TỐT NGHIỆP

Ng□ời h□ớng dẫn thứ nhất:

Họ và tên:.....

Học hàm, học vị:.....

Cơ quan công tác:.....

Nội dung h- ớng dẫn: …….....

……

……

……

……

Ng□ời h□ớng dẫn thứ hai:

Họ và tên: …….....

Học hàm, học vị…….....

Cơ quan công tác: ……..

Nội dung h- ớng dẫn: …….....

……

……

……

……

Đề tài tốt nghiệp đ- ợc giao ngày 12 tháng 04 năm 2010

Yêu cầu phải hoàn thành tr- ớc ngày 10 tháng 07 năm 2010

Đã nhận nhiệm vụ: Đ.T.T.N	Đã nhận nhiệm vụ: Đ.T.T.N
Sinh viên	Cán bộ h- ớng dẫn Đ.T.T.N

Hải Phòng, ngày .....tháng.....năm 2010

Hiệu tr- ởng

GS.TS.NGƯT Trần Hữu Nghị

PHẦN NHẬN XÉT TÓM TẮT CỦA CÁN BỘ H□ ỚNG DẪN

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

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

2. Đánh giá chất l- ợng của đề tài tốt nghiệp (so với nội dung yêu cầu đã đề ra trong nhiệm vụ đề tài tốt nghiệp)

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

3. Cho điểm của cán bộ h- ớng dẫn:

   (Điểm ghi bằng số và chữ)

.....
.....

Ngày.....tháng.....năm 2010

Cán bộ h- ớng dẫn chính

(Ký, ghi rõ họ tên)

PHẦN NHẬN XÉT ĐÁNH GIÁ CỦA CÁN BỘ CHẤM PHẢN BIỆN ĐỀ TÀI TỐT NGHIỆP

1. Đánh giá chất lượng đề tài tốt nghiệp (về các mặt như cơ sở lý luận, thuyết minh chương trình, giá trị thực tế, ...)

2. Cho điểm của cán bộ phản biện

(*Điểm ghi bằng số và chữ*)

.....

.....

Ngày.....tháng.....năm 2010

Cán bộ chấm phản biện

(Ký, ghi rõ họ tên)

LỜI CẢM ƠN

Trong suốt quá trình làm khoá luận tốt nghiệp vừa qua, d- ới sự giúp đỡ, chỉ bảo nhiệt tình của thấy giáo h- ớng dẫn PGS.TS. Trịnh Nhật Tiến, khoá luận tốt nghiệp của em đã đ- ợc hoàn thành. Em xin bày tỏ lòng biết ơn sâu sắc tới thầy Trịnh Nhật Tiến.

Và em cũng xin chân thành cảm ơn các thầy cô giáo trong khoa Công Nghệ Thông Tin tr- ờng Đại Học Dân Lập Hải Phòng đã giảng dạy, giúp đỡ và tạo điều kiện tốt nhất để chúng em hoàn thành tốt khoá luận của mình.

Em xin đ- ợc gửi lời cảm ơn của mình tới gia đình và bạn bè, những ng- ời đã động viên giúp đỡ em trong quá trình làm khoá luận tốt nghiệp.

Mặc dù đã cố gắng hết sức cùng với sự tận tâm của thầy giáo h- ớng dẫn song do thời gian ngắn và trình độ còn hạn chế nên em khoá luận của em vẫn còn nhiều thiếu sót. Em rất mong nhận đ- ợc sự chỉ dẫn của các thầy cô và sự góp ý của các bạn để khóa luận của em đ- ợc hoàn thiện hơn.

Sinh viên

Ngô Hồng Trang

MỤC ĐÍCH ĐỀ TÀI

Đề tài: Một số dạng tấn công hệ thống thông tin và phòng chống bằng kĩ thuật mật mã.

Mục đích chính của đề tài được đưa ra trong khóa luận là:

1/. Tìm hiểu một số dạng tấn công hệ thống thông tin (thông qua mạng máy tính, hệ điều hành, cơ sở dữ liệu….)

2/. Tìm hiểu một số kĩ thuật mật mã.

3/. Nghiên cứu phương pháp phòng chống tấn công bằng kĩ thuật mật mã

4/. Viết ít nhất một chương trình mật mã để phòng chống tấn công.

MỤC LỤC

BẢNG CÁC THUẬT NGỮ, CHỮ VIẾT TẮT

Stt	Từ viết tăt	Từ đầy đủ	Nghĩa tiếng Việt
1	DNS	Domain Name System	Hệ thống phân giải tên miền
2	ARP	Address Resolution Protocol	Giao thức phân giải địa chỉ
3	SSL	Secure Socket Layer	Bảo mật lớp
4	ICMP	Internet Control Message Protocol	Giao thức tạo thông điệp điều kiển của Internet
5	P2P	Peer to peer	Mạng ngang hàng
6	UDP	User Datagram Protocol	Giao thức truyền dữ liệu không kết nối
7	DES	Data Encryption Standard	Tiêu chuẩn mã hóa dữ liệu
8	CSDL	Cơ sở dữ liệu	Cơ sở dữ liệu
9	IP	Initial Permutation	Hoán vị ban đầu
10	IP-1	Inverse of the Initial Permotation	Nghịch đảo của hoán vị ban đầu
11	SYN	synchronize	Đồng bộ hóa
12	MD4	Message Digest 4	Tóm tắt thông điệp 4
13	SHA	Secure Hash Algorithm	Thuật toán hàm băm an toàn
14	CA	Certificate Authority	Tổ chức chứng nhận khóa công khai
15	DoS	Denial of Service	Từ chối dịch vụ
16	SQL	Stucted Query Language	Ngôn ngữ truy vấn có cấu trúc
17	IPS	Intrusion Prevention System	Hệ thống ngăn chặn xâm nhập
18	USCLN	Ước số chung lớn nhất	Ước số chung lớn nhất

Chương 1. CƠ SỞ TOÁN HỌC

1.1. CÁC KHÁI NIỆM CƠ BẢN

1.1.1. Khái niệm đồng dư

1.1.1.1. Khái niệm

Cho n là số nguyên dương. Nếu a và b là 2 số nguyên, khi đó a được gọi là đồng dư với b theo modulo n, được viết $a \equiv b \pmod n$ nếu n chia hết cho (a - b) và n được gọi là modulo của đồng dư.

Ví dụ: $24 \equiv 9 \pmod 5$.

1.1.1.2. Tính chất

- $a \equiv b \pmod n$, nếu và chỉ nếu a và b đều trả số dư như nhau khi $n \mid (a-b)$.
- $a \equiv a \pmod n$ (tính phản xạ)
- Nếu $a \equiv b \pmod n$ thì $b \equiv a \pmod n$.
- Nếu $a \equiv b \pmod n$ và $b \equiv c \pmod n$ thì $a \equiv c \pmod n$.
- Nếu $a \equiv a_1 \pmod n$ và $b \equiv b_1 \pmod n$ thì $a+b = (a_1+b_1) \pmod n$

  và $a . b \equiv a_1 . b_1 \pmod n$.

  Có thể cộng, trừ, nhân và nâng lên lũy thừa các đồng dư thức có cùng một modulo.

  Nếu ta có:

  $a_1 \equiv a_2 \pmod n$

  $b_1 \equiv b_2 \pmod n$

Thì ta có:

- $(a_1 + a_2) \equiv (b_1 + b_2) \bmod n$
- $(a_1 - a_2) \equiv (b_1 - b_2) \bmod n$
- $(a_1 * a_2) \equiv (b_1 * b_2) \bmod n$
- $a_1^{\,k} \equiv b_1^{\,k} \bmod n$, với k là số nguyên.

1.1.2. Số nguyên tố.

1.1.2.1 Khái niệm

Số nguyên tố là một số lớn hơn 1, nhưng chỉ chia hết cho 1 và chính nó.

Ví dụ: 2, 3, 5, 7, 11, 13, 17, 19, 23, ……

Số lượng số nguyên tố là vô tận. Các hệ mã thường sử dụng số nguyên tố lớn cỡ 512 bit và lớn hơn.

1.1.2.2.Các tính chất số nguyên tố

1/. Ước số dương bé nhất khác 1 của một hợp số a là một số nguyên tố không vượt quá $\sqrt{a}$

2/. 2 là số nguyên tố nhỏ nhất và cũng là số nguyên tố chẵn duy nhất

3/. Tập hợp các số nguyên tố là vô hạn.

Định lý

Mọi số tự nhiên lớn hơn 1 đều phân tích được thành tích những thừa số nguyên tố, và sự phân tích này là duy nhất nếu không kể đến thứ tự của các thừa số.

Từ đó có dạng phân tích tiêu chuẩn của một số tự nhiên bất kỳ là:

$$n = p_1^{k_1} p_2^{k_2} ... p_m^{k_m}$$

Trong đó p_1, p_2,..., p_m là các số nguyên tố đôi một khác nhau.

Ví dụ: $300 = 2^2 * 5^2 * 3$

1.1.2.3. Thuật toán kiểm tra n có phải số nguyên tố

Chúng ta có thể kiểm tra theo thuật toán sau:

```
int nguyento(int n)
  {
    if (n>1)
       {
          int i, j=1;
          for ( i=2; i<= sqrt ( n ) ; i++ )
                if (n%i == 0) { j= 0; break; }
          return j;
       }
    else return 0;
  }
```

Ngoài ra có nhiều thuật toán kiểm tra số nguyên tố khác như: Soloway-trassen, Rabin-Miller, Lehmann….

1.1.3. Số nguyên tố cùng nhau.

1.1.3.1. Khái niệm

Các số nguyên dương *a* và *b* được gọi là **nguyên tố cùng nhau** nếu chúng có ước chung lớn nhất là 1.

Chẳng hạn: 6 và 35 là nguyên tố cùng nhau

6 và 27 không nguyên tố cùng nhau ví có Ước chung lớn nhất là 3.

Số 1 là nguyên tố cùng nhau với mọi số nguyên.

Một phương pháp xác định tính nguyên tố cùng nhau của hai số nguyên là sử dụng thuật toán Euclid như sau:

Input: 2 số nguyên không âm a và b sao cho $a \geq b$

Output: ước số chung lớn nhất của a và b.

```
Procedure USCLN (a, b: positive integers)
Begin
        x: =a
        y: = b
        while y ≠ 0
        begin
        r: = x mod y
        x: = y
        y: =r
        end  { x la USCLN can tim }
```

1.1.3.2.Hàm phi Euler

Hàm phi Euler của một số nguyên dương *n* là số các số nguyên giữa 1 và *n* , và nguyên tố cùng nhau với *n*. Kí hiệu: ∅ (*n*).

Ví dụ:

∅ (*9*)= 6 vì có sáu số 1, 2, 4, 5, 7 và 8 là nguyên tố cùng nhau với 9.

Nhận xét:

Nếu p là một số nguyên tố thì ∅ (*p*) = p-1.

Ví dụ:

∅ (7) = 6 vì có sáu số 1, 2, 3, 4, 5 và 6 là nguyên tố cùng nhau với 7.

Tính chất hàm phi Euler:

1/. Nếu p là số nguyên tố thì **∅ (p) =p – 1.**

2/. Nếu n là tích của 2 số nguyên tố p và q, n = p * q,

thì: ∅ (n) = ∅ (p) * ∅ (q) = (p – 1) * (q - 1).

3/. Nếu a là nguyên tố cùng nhau với n, nghĩa là ƯCLN (a , n) = 1,

thì: $a^{\emptyset(n)} = 1 \bmod n$.

4/. Nếu khai triển $n = p_1^{e1} * p_2^{e2} * \ldots * p_k^{ek}$,

thì: $\emptyset(n) = n * (1 - \frac{1}{p_1}) * (1 - \frac{1}{p_2}) * \ldots * (1 - \frac{1}{p_k})$.

Ví dụ:

∅ (21) = ∅ (3) * ∅ (7) = 2 * 6= 12

1.2. MỘT SỐ KHÁI NIỆM TRONG ĐẠI SỐ

1.2.1. Nhóm

1.2.1.1. Khái niệm

Nhóm (*G*, *) là một tập hợp *G*, cùng với một phép toán 2 ngôi trên G

Ký hiệu " * ", từ $G \times G$ vào *G* thỏa mãn các tiên đề sau:

G1.Tính kết hợp: phép toán "*" có tính kết hợp, nghĩa là

(*a* * *b*) * *c* = *a* * (*b* * *c*) với mọi *a*, *b* và *c* thuộc *G*.

G2.Phần tử trung hòa:Trong *G* tồn tại một phần tử được gọi là phần tử trung hòa θ sao cho với mọi phần tử *a* thuộc *G* thì *a**θ = θ**a* = *a*.

G3.Phần tử đối lập: với mỗi phần tử *a* thuộc *G* tồn tại một phần tử *x*, gọi là phần tử đối lập của *a*, sao cho: *x* * *a* = *a* * *x* = θ.

Trong định nghĩa của nhóm phép "*" không đòi hỏi có tính chất giao hoán (*a***b*=*b***a*) nếu G thỏa mãn thêm tính chất này thì *G* được gọi là nhóm giao hoán, hay ***nhóm Abel***. Nếu *G* không có tính giao hoán thì *G* được gọi là ***phi giao hoán*** **hay** ***không Abel***

Ví dụ: **Nhóm giao hoán (nhóm Abel)**

1/. Tập các số nguyên, *Z*, với phép toán là phép cộng thông thường, phần tử đơn vị là 0.

2/. Tập các số hữu tỷ dương với phép toán là phép nhân thông thường, phần tử đơn vị là 1.

1.2.1.2 Khái niệm Nhóm con của nhóm (G,)*

Cho một nhóm *G* với phép toán hai ngôi *, và tập con *H* của *G*. *H* được gọi là **nhóm con** của *G* nếu chính *H* là một nhóm với phép toán * của *G*.

Các điều kiện tương đương:

Cho tập con *H* của nhóm *G*. Các điều kiện sau là tương đương:

1. *H* là nhóm con của *G*;
2. Với mọi *a*, *b* $\in H$ ta có $a * b \in H$ và $a^{-1} \in H$;
3. Với mọi *a*, *b* $\in H$ ta có $a * b^{-1} \in H$;

Ví dụ:

1/. Nhóm con sinh bởi tập hợp gồm một số nguyên k là {x . k | x $\in$ Z}

2/. Nhóm con sinh bởi tập m số nguyên $\{k_1, k_2, \ldots, k_m\}$ là tập $\{k_1 * x_1 + k_2 * x_2 + \ldots + k_m * x_m\}$

1.2.1.3. Nhóm Cyclic

1/. Khái niệm:

Một 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

$g^n = g * g * \ldots * g = a$. (là g * g với n lần).

Khi đó g được gọi là ***phần tử sinh*** hay ***phần tử nguyên thủy*** của nhóm G.

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

Ví dụ:

Các căn bậc *n* của đơn vị tạo thành một n nhóm Cyclic cấp *n* với phép nhân, nghĩa là: $0 = z^3 - 1 = (z - s^0)(z - s^1)(z - s^2)$

trong đó $s^i = e^{(2 . \pi . i)/3}$ $\{s^0, s^1, s^2\}$ với phép nhân là Cyclic.

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 $\mathbf{g^n = e}$, thì G sẽ chỉ gồm có n phần tử khác nhau: e, g, g^2, …, g^{n-1}. Khi đó G được gọi là nhóm Cyclic hữu hạn cấp n.

Nếu không tồn tại số tự nhiên **n** để $\mathbf{g^n = e}$, 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 $\boldsymbol{\alpha} \in \mathbf{G}$ được gọi là có cấp d, nếu d là số nguyên dương nhỏ nhất sao cho $\boldsymbol{\alpha}^{\mathbf{d}} = \mathbf{e}$, trong đó **e** là phần tử trung lập của **G.**

Như vậy phần tử $\boldsymbol{\alpha}$ có cấp 1, nếu $\boldsymbol{\alpha} = \mathbf{e}$.

1.2.1.4. Phần tử nghịch đảo.

1/. Khái niệm:

Cho a $\in Z_n$, nếu tồn tại b $\in Z_n$ sao cho a * b = 1 mod n, ta nói b là phần tử nghịch đảo của a trong Z_n và kí hiệu a^{-1}.

Ví dụ: Xét trong tập Z_{14} = {0, 1, 2,…, 13} cho a = 3 thì a^{-1} = 5 vì : 3 * 5= 1 mod 14.

2/. Tính chất khả nghịch

Cho a $\in Z_n$, ta nói a là khả nghịch khi và chỉ khi gcd (a, n)=1

(tức là có tồn tại a^{-1}).

Định lý:

USCL (a, n) = 1 tương đương với phần tử a $\in Z_n$ có phần tử nghịch đảo.

1.2.1.5. Nhóm nhân của tập Z_n

a/. Nhóm nhân của tập Z_n là Z^*_n được định nghĩa như sau:

Z^*_n= { a $\in Z_n$ | gcd (a, n)=1}

Nếu n là số nguyên tố thì : Z^*_n= {a | 1 ≤ a ≤ n - 1}

b/. Cấp của Z^*_n là số phần tử của Z^*_n= ∅ (n).

Ví dụ:

Cho n = 26 thì Z^*_n={1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25}

vậy ta có: ∅ (26)= 12

c/. Cấp của a $\in Z^*_n$

Cấp của a kí hiệu là ord (a) là số nguyên dương t nhỏ nhất sao cho a^t = 1 mod n.

Ví dụ:

Z_{21} = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20}

Z^*_{21}={1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20}

Ord (2) = 6 vì 2^6 = 1 mod 21.

1.1.2.6. Phần tử sinh (phần tử nguyên thủy).

Định nghĩa:

Cho $\alpha \in Z^*_n$, nếu ord $(\alpha) = \varnothing(n)$ thì α được gọi là phần tử sinh hay phần tử nguyên thủy của Z^*_n.

Nếu Z^*_n, có phần tử sinh thì Z^*_n, được gọi là nhóm Cyclic.

Tính chất:

1/. Nếu α là phần tử sinh của Z^*_n thì: $Z^*_n = \{\alpha^i \bmod n \mid 0 \le i \le (\varnothing(n) - 1)\}$

2./ Giả sử α là phần tử sinh của Z^*_n khi đó $b = \alpha^i \pmod n$ cũng là phần tử sinh khi và chỉ khi $\gcd(i, \varnothing(n)) = 1$

Khi Z^*_n là nhóm cyclic thì số phần tử sinh là: $\varnothing(\varnothing(n))$.

3/. Z^*_n có phần tử sinh khi n = 2, 4, ..P^k hay 2 P^k khi P là số nguyên tố lẻ và $k \ge 1$.

Ví dụ: cho $Z_7 = \{0, 1, 2, 3, 4, 5, 6\}$ thì

$Z^*_7 = \{1, 2, 3, 4, 5, 6\}$

$\varnothing(7) = 6$

các phần tử sinh của Z_7 là: { 3, 5}

vì:

$1^1 \bmod 7 = 1 \bmod 7 = 1$ nên $1^1 \equiv 1 \bmod 7$ vậy ord (1) = $1 \ne \varnothing(7)$

$2^3 \bmod 7 = 8 \bmod 7 = 1$ nên $2^3 \equiv 1 \bmod 7$ vậy ord (2) = $3 \ne \varnothing(7)$

$3^6 \bmod 7 = 729 \bmod 7 = 1$ nên $3^6 \equiv 1 \bmod 7$ vậy ord (3) = $6 = \varnothing(7)$

$4^3 \bmod 7 = 64 \bmod 7 = 1$ nên $4^3 \equiv 1 \bmod 7$ vậy ord (4) = $3 \ne \varnothing(7)$

$5^6 \bmod 7 = 15625 \bmod 7 = 1$ nên $5^6 \equiv 1 \bmod 7$ vậy ord (5) = $6 = \varnothing(7)$.

$6^2 \bmod 7 = 36 \bmod 7 = 1$ nên $6^2 \equiv 1 \bmod 7$ vậy ord (6) = $2 \ne \varnothing(7)$.

Vậy Z_7^* có 2 phần tử sinh là 3 và 5.

1.2.2. Độ phức tạp của thuật toán

1.2.2.1.Khái niệm Thuật toán

"Thuật toán **"** được hiểu đơn giản là cách thức để giải một bài toán. Cũng có thể được hiểu bằng 2 khái niệm: Trực giác hay Hình thức như sau:

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

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

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

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

Thuật toán được chia làm 2 loại: Đơn định và không đơn định.

Thuật toán đơn định

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

Thuật toán không đơn định

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

1.2.2.2. Khái niệm Độ phức tạp của thuật toán

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

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

Chi phí thời gian của một quá trình tính toán là thời gian cần thiết để thực hiện một quá trình tính toán.

Chi phí bộ nhớ của một quá trình tính toán là số ô nhớ cần thiết để thực hiện một quá trình tính toán.

Gọi A là một thuật toán, e là dữ liệu vào của bài toán đã được mã hóa bằng cách nào đó. Thuật toán A tính trên dữ liệu vào e phải trả một giá trị nhất định. Ta kí hiệu $t_A(e)$ là giá trị thời gian và $l_A(e)$ là giá bộ nhớ.

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

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

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

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

4/. Độ phức tạp tiệm cận: độ phức tạp PT (n) được gọi là tiệm cận với hàm f (n), kí hiệu O (f (n)) nếu tồn tại các số n_0, c mà PT (n) $\leq$ c . f (n) , $\forall$ n $\geq n_0$.

5/. Độ phức tạp đa thức:

Độ phức tạp PT (n) được gọi là đa thức, nếu có 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.

1.2.2.3. Phân lớp bài toán theo độ phức tạp

1/.Khái niệm "dẫn về được".

Bài toán được gọi là "dẫn về được" bài toán A một cách đa thức, kí hiệu B ∞ A, nếu có thuật toán đơn định đa thức để giải bài toán A, thì cũng có thuật toán đơn định để giải bài toán B.

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

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

Quan hệ ∞ có tính chất bắc cầu: Nếu C ∞ B và B ∞ A thì C ∞ A.

2/. Khái niệm "khó tương đương"

Bài toán A gọi là "khó tương đương" bài toán B, kí hiệu A ~ B, nếu A ∞ B và B ∞ A.

1.2.2.4. Các lớp bài toán

1/. Khái niệm 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 là tập con của NP.

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

2/. Khái niệm lớp bài toán NP- Hard

Bài toán A được gọi là NP-Hard (NP-khó) nếu mọi L$\in$ NP đều là L ∞ A.

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

3/.Khái niệm 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 bao gồm tất cả những bài toán NP- Complete.

1.2.2.5. Khái niệm hàm một phía, hàm cửa sập một phía.

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

Ví dụ:

Hàm $f(x) = g^{x} \pmod p$, với p là số nguyên tố lớn là hàm một phía.

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

Ví dụ: $f(x) = x^{\alpha} \pmod n$ (n là tích của 2 số nguyên tố lớn, n = p * q) là hàm một phía. Nếu chỉ biết a và n thì tính $x = f^{-1}(y)$ rất "khó", nhưng nếu biết cửa sập p hoặc q thì tính được $f^{-1}(y)$ là khá "dễ".

Chương 2. MỘT SỐ KĨ THUẬT MẬT MÃ

2.1. VẤN ĐỀ MÃ HÓA

2.1.1. Khái niệm mật mã

Mật mã học là một lĩnh vực liên quan với các kỹ thuật ngôn ngữ và toán học để đảm bảo an toàn thông tin, cụ thể là trong thông tin liên lạc. Về phương diện lịch sử, mật mã học gắn liền với quá trình mã hóa.

Trước đây mật mã chỉ được dùng trong ngành an ninh quốc phòng, ngày nay việc bảo đảm an toàn thông tin là nhu cầu của mọi ngành, mọi người(do thông tin chủ yếu truyền trên mạng công khai), vì vậy kĩ thuật mật mã là công khai cho mọi người dùng. Điều bí mật nằm ở khóa mật mã.

Hiện nay có nhiều kĩ thuật mật mã khác nhau, mỗi kĩ thuật có ưu và nhược điểm riêng. Tùy theo yêu cầu của môi trường ứng dụng ta dùng kĩ thuật này hay kĩ thuật khác. Có môi trường cần phải an toàn tuyệt đối, bất kể thời gian và chi phí. Có môi trường lại cần phải dung hòa giữa bảo mật và chi phí thực hiện.

Mật mã cổ điển chủ yếu dùng để "che dấu" dữ liệu. Với mật mã hiện đại ngoài khả năng "che dấu" dữ liệu, còn dùng để thực hiện: Ký số (ký điện tử), tạo giao diện thông điệp, giao thức bảo toàn dữ liệu, xác thực thực thể, giao thức thỏa thuận, giao thức phân phối khóa….

Theo nghĩa hẹp, mật mã dùng để bảo mật dữ liệu, người ta quan niệm: Mật mã học là môn khoa học nghiên cứu mật mã: tạo mã và phân tích mã (thám mã).

Vai trò của mật mã:

- Bảo đảm bí mật (Bảo mật): Thông tin không bị lộ đối với người không được phép.
- Bảo đảm toàn vẹn (Bảo toàn):

Ngăn chặn hay hạn chế việc bổ sung, loại bỏ và sửa chữa dữ liệu không được phép.

- Bảo đảm xác thực (Chứng thực):

Xác thực đúng thực thể cần kết nối, giao dịch.

Xác thực đúng thực thể có trách nhiệm về nội dung thông tin (xác thực nguồn gốc thông tin).

- Bảo đảm sẵn sàng: Thông tin sẵn sàng cho người dùng hợp pháp.

2.1.2. Khái niệm mã hóa

- **Mã hóa** là phương pháp để biến thông tin (như phim ảnh, văn bản, hình ảnh…) dạng hiểu được sang dạng thông tin không thể hiểu được nếu không có phương tiện giải mã.
- **Giải mã** là phương pháp để đưa từ dạng thông tin đã được mã hóa về dạng thông tin ban đầu, quá trình ngược của mã hóa.
- **Hệ mã hóa:**

Việc mã hóa theo quy tắc nhất định, quy tắc đó gọi là **hệ mã hóa**

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

P là tập hữu hạn các bản rõ có thể.

C là tập hữu hạn các bản mã có thể.

K tập hữu hạn các khóa có thể.

E là tập các hàm lập mã.

D là tập các hàm giải mã.

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

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

sao cho $\mathbf{d}_{\mathbf{kd}}(\mathbf{e}_{\mathbf{ke}}(\mathbf{x})) = \mathbf{x}$, $\forall\ x \in \mathbf{P}$.

Ở đây **x** được gọi là bản rõ, $\mathbf{e}_{\mathbf{ke}}(\mathbf{x})$ được gọi là bản mã.

- **Thuật toán mã hóa** hay **giải mã** là thủ tục để thực hiện mã hóa hay giải mã.

- **Khóa mã hóa** là một giá trị làm cho thuật toán mã hóa thực hiện theo cách riêng biệt và sinh ra bản rõ riêng. Thông thường khóa càng lớn thì bản mã càng an toàn.

Phạm vi các giá trị có thể có của khóa được gọi là ***không gian khóa***.

2.1.3. Phân loại mã hóa.

Có 2 loại mã hóa: *mã hóa khóa đối xứng* và *mã hóa khóa bất đối xứng*.

- **Hệ mã hóa khóa đối xứng** có khóa lập mã và khóa giải mã "giống nhau", theo nghĩa biết được khóa này thì "dễ" tính được khóa kia. Vì vậy phải dữ bí mật cả hai khóa.
- **Hệ mã hóa khóa bất đối xứng** có khóa lập mã khác khóa giải mã, biết được khóa này cũng khó tìm được khóa kia. Vì vậy, chỉ cần bí mật khóa giải mã, còn công khai khóa lập mã.

2.1.4. Hệ mã hóa khóa đối xứng

2.1.4.1.Khái niệm mã hóa khóa đối xứng

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

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

Ví dụ:

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

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

2.1.4.2. Một số đặc điểm của hệ mã hóa khóa đối xứng

1/. Ưu điểm

Các hệ mã hóa khóa đối xứng mã hóa và giải mã ***nhanh hơn*** hệ mã hóa khóa công khai.

2/. **Hạn chế**

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

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

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

Mặt khác khi hai người (lập mã và giải mã) cùng biết chung một bí mật, thì càng khó giữ được bí mật.

3/. Tấn công đối với các mật mã đối xứng

Các mã đối xứng thường rất dễ bị ảnh hưởng bởi các loại tấn công gọi là tấn công với văn bản thuần túy biết trước (known-plaintext attacks), tấn công với văn bản thuần túy chọn trước (chosen plaintext attacks), thám mã vi phân (differential cryptanalysis) và thám mã tuyến tính (linear cryptanalysis). Nếu mỗi hàm số sử dụng trong các vòng tính toán được thiết kế một cách cẩn thận, thì nó sẽ giảm khả năng chìa khóa của mã bị tấn công rất nhiều.

2.1.4.3. Một số hệ mã khóa cổ điển

1/. Mã dịch chuyển

a/. Sơ đồ:

Đặt P = C = K = Z_{26}. Bản mã y và bản rõ x $\in$ Z_{26}.

Với khóa k $\in$ K, ta định nghĩa:

Hàm mã hóa $y = e_k (x) = (x + k) \bmod 26$.

Hàm giải mã $x = d_k (y) = (y – k) \bmod 26$.

b/. Ví dụ:

- Bản rõ chữ x= ”I L O V E Y O U” và khóa k = 3
- Bản rõ số : 8 11 14 21 4 24 14 20

Với phép mã hóa $y = e_k (x) = (x + k) \bmod 26 = (x + 3) \bmod 26$, ta nhận được:

- Bản mã số: 11 14 17 24 7 1 17 23
- Bản mã chữ: L O R Y H B R X

Giả mã: $x = d_y (y) = (y – k) \bmod 26 = (y – 3) \bmod 26$, ta nhận được bản rõ.

c/. Các dạng tấn công vào mã dịch chuyển: Tìm cách xác định khóa k.

Do chỉ có 26 khóa nên việc thám mã có thể thực hiện theo kiểu “biết bản mã ” bằng duyệt tuần tự các khóa cho tới khi nhận được bản rõ có ý nghĩa.

Ví dụ: khi thám mã có trong tay một bản mã là: “LORYHBRX”. Thám mã sẽ thực hiện duyệt lần lượt khóa k = 1 -> k = 26 để tìm ra bản rõ.

Với k = 3 được bản rõ:”ILOVEYOU”

→Giải pháp phòng tránh:

Mở rộng vùng không gian khóa lớn. Ví dụ như bảng chữ cái Tiếng Việt có thanh (gồm 94 kí tự) thì việc thử tất cả các khóa cũng lâu hơn bảng Tiếng Anh.

2/. Mã Affine

a/. Sơ đồ:

Đặt P = C = Z_{26}. Bản mã y và bản rõ x $\in Z_{26}$.

Tập khóa: K = {(a, b) với a, b $\in Z_{26}$, gcd (a, 26) = 1}

Với khóa k = (a, b) $\in$K, ta định nghĩa:

Phép mã hóa: $y = e_k (x) = (a * x + b) \bmod 26$.

Phép giải mã: $x = d_k (y) = a^{-1} (y – b) \bmod 26$.

b/. Ví dụ:

Bản rõ chữ: "C N T T"

Chọn khóa: k = (5, 8)

Bản rõ số: 2 13 19 19

Mã hóa theo công thức $y = e_k (x) = (a * x + b) \bmod 26 = (5 * 2 + 8) \bmod 26$

Bản mã số: 18 21 25 25

Bản mã chữ: S V Z Z

Giải mã theo công thức:

$x = d_k(y) = a^{-1}(y–b) \bmod 26 = 5^{-1} * (y–8) \bmod 26 = 21*(y–8) \bmod 26$.

c. Các dạng tấn công vào Affine: Tìm cách xác định khóa k

Khóa mã Affine có dạng k = (a, b) với a, b $\in Z_{26}$ và gcd (a, 26)= 1.

Kí tự mã y và kí tự bản rõ x tương ứng có quan hệ: y = (a . x + b) mod 26

Thám mã sử dụng phương pháp **sắc suất thống kê**, dựa vào tần suất xuất hiện của các kí tự trong Tiếng Anh. Từ đó biết được 2 cặp (x,y) khác nhau và có được hệ phương trình tuyến tính 2 ẩn, giải hệ đó tìm ra giá trị a, b tức là tìm ra khóa k. Kết hợp với 12 số thuộc Z_{26} nguyên tố với 26, nên số khóa là: 12 * 26 = 312.

→Giải pháp phòng tránh tấn công:

Mở rộng vùng không gian khóa lớn. Ví dụ như bảng chữ cái tiếng Việt có thanh (gồm 94 ký tự). Số kí tự nhiều thì tần suất xuất hiện của các chữ cái cũng không khác biệt nhau nhiều lắm, do đó để phát hiện được kí tự "nổi bật" cũng khó khăn hơn. Và khi đó số khóa có thể có lớn hơn 312, việc thử tất cả các trường hợp sẽ lâu hơn.

2.1.4.4.Hệ mã hóa DES.

DES (Data Encryption Standard, hay Tiêu chuẩn Mã hóa Dữ liệu) là một phương pháp mã hóa được FIPS (Tiêu chuẩn Xử lý Thông tin Liên bang Hoa Kỳ) chọn làm chuẩn chính thức vào năm 1976. Sau đó chuẩn này được sử dụng rộng rãi trên phạm vi thế giới. Ngay từ đầu, thuật toán của nó đã gây ra nhiều tranh cãi, do nó bao gồm các thành phần thiết kế mật, độ dài khóa tương đối ngắn và nghi ngờ về cửa sau để Cơ quan An ninh quốc gia Hoa Kỳ (NSA) có thể bẻ khóa. Do đó, DES đã được giới nghiên cứu xem xét rất kỹ lưỡng, việc này đã thúc đẩy hiểu biết hiện đại về mật mã khối (block cipher) và các phương pháp thám mã tương ứng.

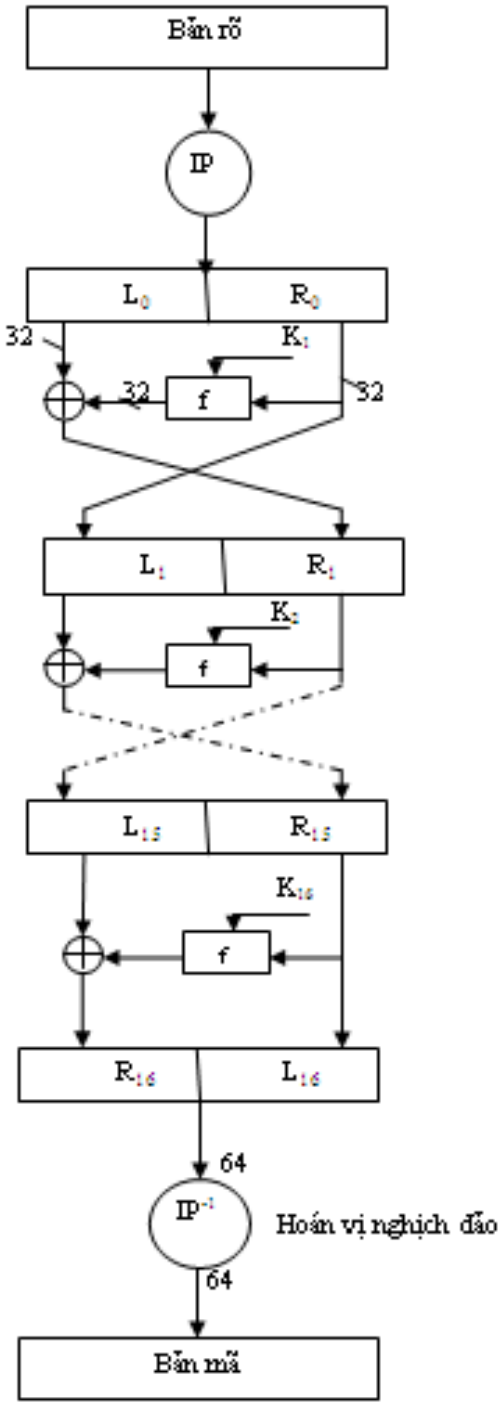
Hiện nay DES được xem là không đủ an toàn cho nhiều ứng dụng. Nguyên nhân chủ yếu là độ dài 56 bit của khóa là quá nhỏ. Khóa DES đã từng bị phá trong vòng chưa đầy 24 giờ. Đã có rất nhiều kết quả phân tích cho thấy những điểm yếu về mặt lý thuyết của mã hóa có thể dẫn đến phá khóa, tuy chúng không khả thi trong thực tiễn. Thuật toán được tin tưởng là an toàn trong thực tiễn có dạng Triple DES (thực hiện DES ba lần), mặc dù trên lý thuyết phương pháp này vẫn có thể bị phá. Gần đây DES đã được thay thế bằng AES (Advanced Encryption Standard, hay Tiêu chuẩn Mã hóa Tiên tiến).

1/. Mô tả thuật toán

DES là thuật toán mã hóa khối, nó xử lý từng khối thông tin của bản rõ có độ dài xác định và biến đổi theo những quá trình phức tạp để trở thành khối thông tin của bản mã có độ dài không thay đổi, độ dài mỗi khối là 64 bit. DES cũng sử dụng khóa để cá biệt hóa quá trình chuyển đổi. Nhờ vậy, chỉ khi biết khóa mới có thể giải mã được bản mã. Khóa dùng trong DES có độ dài toàn bộ là 64 bit. Tuy nhiên chỉ có 56 bit thực sự được sử dụng; 8 bit còn lại chỉ dùng cho việc kiểm tra. Vì thế, độ dài thực tế của khóa chỉ là 56 bit.

Một khối dữ liệu 64 bit được mã hóa bằng việc cho qua một bảng hoán vị ban đầu IP(Initial Permutation), sau đó qua một quá trình tính toán phức tạp có sự tham gia của khóa K, được thực hiện và cuối cùng bản mã nhận được sau khi qua một bản hoán vị nghịch đảo (IP^{-1} Inverse of the Initial Permutation).

Sơ đồ tổng quát như sau:



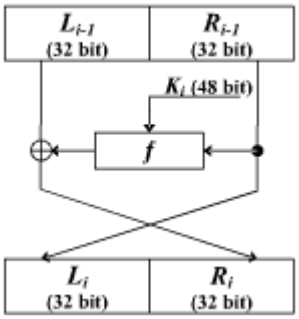
Ký hiệu : ⊕ thể hiện phép toán XOR

2/. Quá trình mã hóa:

64 bit của khối dữ liệu đầu vào được hoán vị bằng IP. Trong đó, bit thứ 58 của khối dữ liệu là bít đầu tiên, bít thứ 50 của khối dữ liệu là bit thứ 2,… bít cuối cùng của khối dữ liệu sau khi hoán vị là bít thứ 7 của khối dữ liệu vào. Kết quả của phép hoán vị ban đầu sẽ được chia làm 2 phần: L0R0 (L0 với 32bit là nửa trái, R0 với 32bit là nửa phải), sau đó thực hiện 16 lần lặp liên tiếp theo công thức:

$$L_i = R_{i-1} \text{ và } R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

và nhận được $L_{16} R_{16}$. Trong đó f được gọi là hàm mật mã. Hàm f có 2 đối là 2 khối bit, một là R_{i-1} với 32 bit và hai là K_i với 48 bit:



Kết quả của 16 lần lặp liên tiếp là một khối 64 bit ($R_{16} L_{16}$) được gọi là dữ liệu tiền kết quả, khối 64 bit này được cho qua một hoán vị nghịch đảo IP^{-1}.

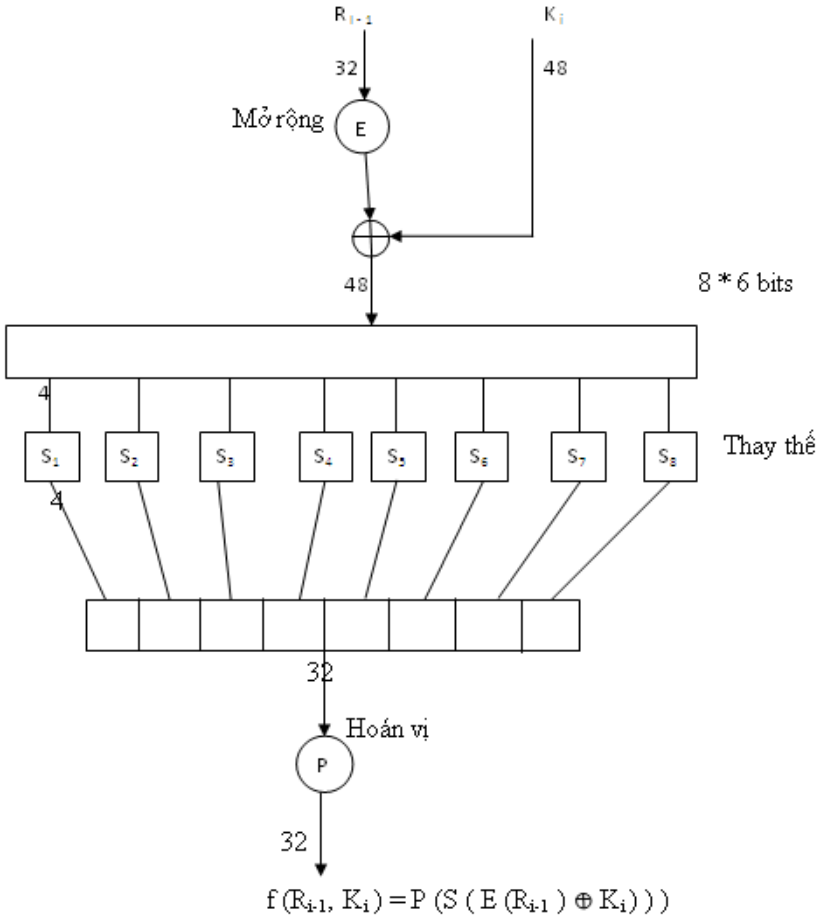
IP							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

IP^{-1}							
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Cuối cùng sau phép hoán vị nghịch đảo ta nhận được một bản mã.

- **Hàm mật mã f:**

Hàm f được tính như sau: $f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$



Hàm E là hoán vị mở rộng. Hàm E thực hiện chức năng mở rộng khối 32bit thành khối 48 bit. E(R) có 3 bit đầu lần lượt là bit thứ 32, 1, 2 của R…2 bit cuối là bit 32 và bit 1 của R.

E					
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

P			
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Sau đó tính E(R) $\oplus$ K với K là khóa 48 bit. Kết quả của phép cộng modulo 2 này sẽ được viết thành 8 nhóm, mỗi nhóm 6 bit dạng $B = B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8$.
Mỗi nhóm B_r 6 bit ($1 \leq r \leq 8$) được đưa qua một hộp đen S_r (S_1, S_2…S_8) để nhận được $S_r(B_r)$ 4 bit đầu ra. Mỗi hộp S_r là một ma trận 4 * 16 trong đó mỗi phần tử của S_r là một số nguyên nằm trong khoảng 0 ->15(có thể biểu diễn tối đa đến 4 bit nhị phân).

Với mỗi $B_r = b_1 b_2 b_3 b_4 b_5 b_6$, ta tính $S_r(B_r)$ như sau: $b_1 b_6$ là biểu diễn nhị phân của số hiệu hàng **i** trong S_r , $b_2 b_3 b_4 b_5$ là biến nhị phân của số hiệu **j** trong S_r.
$C_r = S_r(B_r)$ là phần tử tại hàng i cột j của S_r.

Ví dụ:

Ta tính $S_r(B)$ với B=011011 thì hàng i=01=1, cột j= 1101=13.
Xác định tại cột S_1 giá trị tại hàng 1, cột 13 có giá trị là 5 do đó C= 5= 0101.
$C = C_1 C_2 C_3 C_5 C_6 C_7 C_8$ (32 bit) được cho qua một hoán vị P và nhận được kết quả của hàm f: f = P(C)

Các bảng S_1, S_2, ….S_8

row	column number															
	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]
S_1																
[0]	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
[1]	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
[2]	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
[3]	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2																
[0]	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
[1]	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
[2]	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
[3]	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3																
[0]	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
[1]	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
[2]	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
[3]	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4																
[0]	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
[1]	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
[2]	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
[3]	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5																
[0]	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
[1]	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
[2]	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
[3]	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6																
[0]	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
[1]	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
[2]	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
[3]	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7																
[0]	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
[1]	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
[2]	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
[3]	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8																
[0]	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
[1]	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
[2]	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
[3]	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

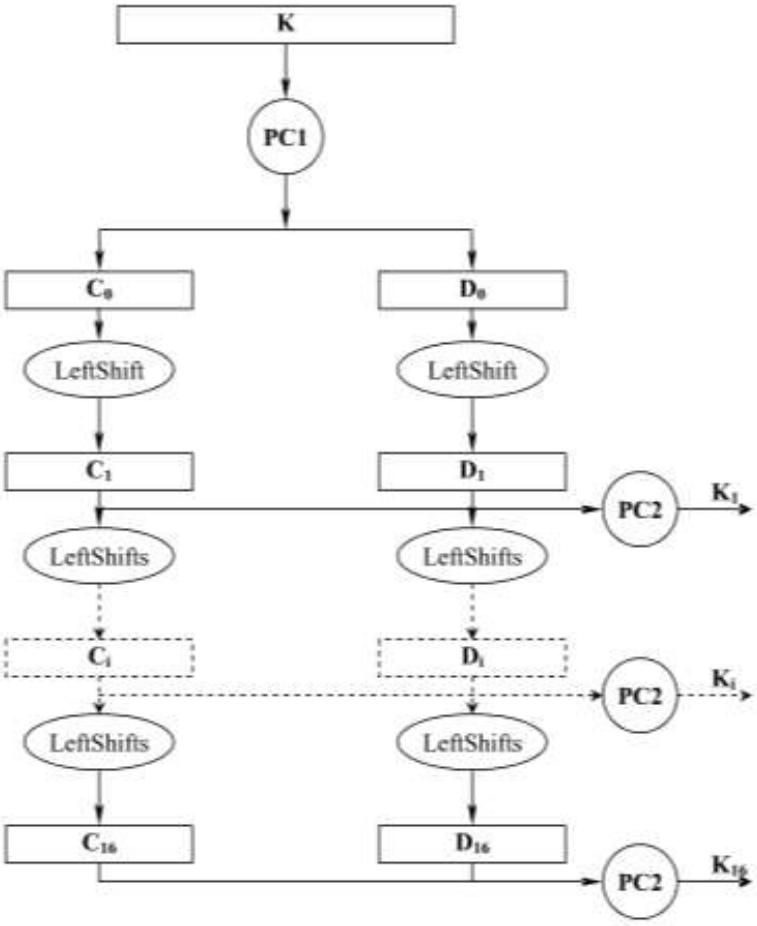
PC 1:

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

PC 2

14	17	11	24	1	5
3	28	50	6	21	10
23	19	59	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

- **Sơ đồ sinh khóa K.**



3/. Quá trình giải mã.

Quy trình giải mã của DES tương tự như quy trình lập mã, nhưng theo trình tự khóa ngược lại: k_{16}, k_{15}, …, k_1.

Đầu vào từ bản mã y, đầu ra là bản rõ x.

4/. Vấn đề thám mã hệ mã hóa DES

Mặc dù đã có nhiều nghiên cứu về phá mã DES hơn bất kỳ phương pháp mã hóa khối nào khác nhưng phương pháp phá mã thực tế nhất hiện nay vẫn là tấn công kiểu duyệt toàn bộ. Nhiều đặc tính mã hóa của DES đã được xác định và từ đó ba phương pháp phá mã khác được xác định với mức độ phức tạp nhỏ hơn tấn công duyệt toàn bộ. Tuy nhiên các phương pháp này đòi hỏi một số lượng bản rõ quá lớn (để tấn công lựa chọn bản rõ) nên hầu như không thể thực hiện được trong thực tế.

a/. Tấn công kiểu duyệt toàn bộ

Đối với bất cứ phương pháp mã hóa nào, kiểu tấn công cơ bản và đơn giản nhất là tấn công kiểu duyệt toàn bộ: thử lần lượt tất cả các khóa có thể cho đến khi tìm ra khóa đúng. Độ dài của khóa sẽ xác định số lượng phép thử tối đa cần thực hiện và do đó thể hiện tính khả thi của phương pháp.

Trong trường hợp của DES, nghi ngờ về độ an toàn của nó đã được đặt ra ngay từ khi nó chưa trở thành tiêu chuẩn. Người ta cho rằng chính NSA đã ủng hộ (nếu không muốn nói là thuyết phục) IBM giảm độ dài khóa từ 128 bit xuống 64 bit và tiếp tục xuống 56 bit. Điều này dẫn đến suy đoán rằng NSA đã có hệ thống tính toán đủ mạnh để phá vỡ khóa 56 bit ngay từ những năm 1970.

b/. Các kiểu tấn công hiệu quả hơn duyệt toàn bộ:

Hiện nay có 3 kiểu tấn công có khả năng phá vỡ DES (với đủ 16 chu trình) với độ phức tạp thấp hơn duyệt toàn bộ: **phá mã vi sai** (differential cryptanalysis - DC), **phá mã tuyến tính** (linear cryptanalysis - LC) và **phá mã Davies** (Davies' attack). Tuy nhiên các dạng tấn công này chưa thực hiện được trong thực tế.

Phá mã vi sai

Eli Biham và Adi Shamir tìm ra vào cuối những năm 1980 mặc dù nó đã được IBM và NSA biết đến trước đó. Để phá mã DES với đủ 16 chu trình, phá mã vi sai cần đến 247 văn bản rõ. DES đã được thiết kế để chống lại tấn công dạng này.

Phá mã tuyến tính

Mitsuru Matsui tìm ra năm 1993, cần 243 bản rõ, phương pháp này đã được Matsui thực hiện và là thực nghiệm phá mã đầu tiên được công bố. Không có bằng chứng chứng tỏ DES có khả năng chống lại tấn công dạng này.

Một phương pháp tổng quát hơn, phá mã tuyến tính đa chiều (multiple linear cryptanalysis), được Kaliski và Robshaw nêu ra vào năm 1994, Biryukov và cộng sự tiếp tục cải tiến vào năm 2004. Nghiên cứu của họ cho thấy mô phỏng tuyến tính đa chiều có thể sử dụng để giảm độ phức tạp của quá trình phá mã tới 4 lần (chỉ còn 241 bản rõ).

Kết quả tương tự cũng có thể đạt được với kiểu tấn công tuyến tính kết hợp với lựa chọn bản rõ (Knudsen and Mathiassen, 2000). Junod (2001) đã thực hiện một số thực nghiệm để tìm ra độ phức tạp thực tế của phá mã tuyến tính và thấy rằng quá trình thực tế nhanh hơn dự đoán: 239×241.

Phá mã Davies:

Trong khi phá mã vi sai và phá mã tuyến tính là các kỹ thuật phá mã tổng quát, có thể áp dụng cho các thuật toán khác nhau, phá mã Davies là một kỹ thuật dành riêng cho DES.

Dạng tấn công này được đề xuất lần đầu bởi Davies vào cuối những năm 1980 và cải tiến bởi Biham và Biryukov (1997). Dạng tấn công mạnh nhất đòi hỏi 250 văn bản rõ, độ phức tạp là 250 và có tỷ lệ thành công là 51%.

Ngoài ra còn có những kiểu tấn công dựa trên bản thu gọn của DES - DES với ít hơn 16 chu trình. Những nghiên cứu này cho chúng ta biết số lượng chu trình cần có và ranh giới an toàn của hệ thống.

Năm 1994, Langford và Hellman đề xuất phá mã vi sai - tuyến tính (differential-linear cryptanalysis) kết hợp giữa phá mã vi sai và tuyến tính.

c/. Một vài đặc điểm về cách giải mã

DES có tính chất bù: trong đó là phần bù của x theo từng bít (1 thay bằng 0 và ngược lại). EK là bản mã hóa của E với khóa K. P và C là văn bản rõ (trước khi mã hóa) và văn bản mã (sau khi mã hóa). Do tính bù, ta có thể giảm độ phức tạp của tấn công duyệt toàn bộ xuống 2 lần (tương ứng với 1 bít) với điều kiện là ta có thể lựa chọn bản rõ.

Ngoài ra DES còn có 4 khóa yếu (weak keys). Khi sử dụng khóa yếu thì mã hóa (E) và giải mã (D) sẽ cho ra cùng kết quả:

$$EK(EK(P)) = P, EK = DK$$

Tuy nhiên có thể dễ dàng tránh được những khóa này khi thực hiện thuật toán, có thể bằng cách thử hoặc chọn khóa một cách ngẫu nhiên. Khi đó khả năng chọn phải khóa yếu là rất nhỏ.

DES đã được chứng minh là không tạo thành nhóm. Nói một cách khác, tập hợp {EK} (cho tất cả các khóa có thể) theo phép hợp thành không tạo thành một nhóm hay gần với một nhóm (Campbell and Wiener, 1992). Vấn đề này vẫn còn để mở và nếu như không có tính chất này thì DES có thể bị phá vỡ dễ dàng hơn và việc áp dụng DES nhiều lần (ví dụ như trong Triple DES (3 DES)) sẽ không làm tăng thêm độ an toàn của DES.

2.1.5. Mã hóa khóa bất đối xứng

2.1.5.1. Tổng quan về mã hóa khóa bất đối xứng

Mã hóa khóa bất đối xứng là một dạng mã hóa cho phép người sử dụng trao đổi thông tin mật mà không cần phải trao đổi khóa chung bí mật trước đó. Điều này được thực hiện bằng cách sử dụng một cặp khóa (có quan hệ toán học với nhau) là khóa công khai và khóa bí mật.

Thuật ngữ **mã hóa khóa công khai** thường được dùng đồng nghĩa với mã hóa khóa bất đối xứng mặc dù hai khái niệm không hoàn toàn tương đương. Có những thuật toán mã hóa khóa bất đối xứng không có tính chất khóa công khai và bí mật như đề cập ở trên mà cả hai khóa (cho mã hóa và giải mã) đều cần phải giữ bí mật.

Trong mã hóa khóa công khai, khóa bí mật phải được giữ bí mật trong khi khóa công khai được phổ biến công khai. Trong 2 khóa, một dùng để mã hóa và khóa còn lại dùng để giải mã. Điều quan trọng đối với hệ thống là "khó" thể tìm ra khóa bí mật nếu chỉ biết khóa công khai.

Hệ thống mã hóa khóa công khai có thể sử dụng với các mục đích:

- Mã hóa: giữ bí mật thông tin và chỉ có người có khóa bí mật mới giải mã được.
- Tạo chữ kí số: cho phép kiểm tra một văn bản có phải đã được tạo với một khóa bí mật nào đó hay không.
- Thỏa thuận khóa: cho phép thiết lập khóa dùng để trao đổi thông tin mật giữa 2 bên.

Thông thường, kỹ thuật mã hóa khóa công khai đòi hỏi khối lượng tính toán nhiều hơn kỹ thuật mã hóa khóa đối xứng nhưng những lợi điểm mà chúng mang lại khiến cho chúng được áp dụng trong nhiều ứng dụng.

1/. Mức an toàn

Về khía cạnh an toàn, mã hóa khóa bất đối xứng cũng không khác nhiều với mã hóa khóa đối xứng. Có những thuật toán được dùng rộng rãi, có thuật toán chủ yếu trên lý thuyết; có thuật toán vẫn được xem là an toàn, có thuật toán đã bị phá vỡ... Cũng cần lưu ý là những thuật toán được dùng rộng rãi không phải lúc nào cũng đảm bảo an toàn.

Một số thuật toán có những chứng minh về độ an toàn với những tiêu chuẩn khác nhau. Nhiều chứng minh gắn việc phá vỡ thuật toán với những bài toán nổi tiếng vẫn được cho là không có lời giải trong thời gian đa thức.

Nhìn chung, chưa có thuật toán nào được chứng minh là an toàn tuyệt đối. Vì vậy, cũng giống như tất cả các thuật toán mật mã nói chung, các thuật toán mã hóa khóa công khai cần phải được sử dụng một cách thận trọng.

2/. Các ứng dụng

Ứng dụng rõ ràng nhất của mã hóa khóa công khai là bảo mật: một văn bản được mã hóa bằng khóa công khai của một người dùng thì chỉ có thể giải mã với khóa bí mật của người đó.

Các thuật toán tạo chữ kí số (khóa công khai) có thể dùng để xác thực. Một người dùng có thể mã hóa văn bản với khóa bí mật của mình. Nếu người khác có thể giải mã với khóa công khai của người gửi, thì có thể tin rằng văn bản thực sự xuất phát từ người gắn với khóa công khai đó.

Các đặc điểm trên còn có ích cho nhiều ứng dụng khác như: tiền điện tử, thỏa thuận khóa,....

2.1.5.2. Hệ mã hóa RSA

Trong mật mã học, **RSA** là một thuật toán mã hóa khóa công khai. Đây là thuật toán đầu tiên phù hợp với việc tạo ra chữ kí điện tử đồng thời với việc mã hóa. Nó đánh dấu một sự tiến bộ vượt bậc của lĩnh vực mật mã học trong việc sử dụng khóa công khai. RSA đang được sử dụng phổ biến trong thương mại điện tử và được cho là đảm bảo an toàn với điều kiện độ dài khóa đủ lớn.

Thuật toán RSA có hai khóa: khóa công khai và khóa bí mật. Khóa công khai được công bố rộng rãi cho mọi người và được dùng để mã hóa. Những thông tin được mã hóa bằng khóa công khai chỉ có thể được giải mã bằng khóa bí mật tương ứng. Nói cách khác, mọi người đều có thể mã hóa nhưng chỉ có người biết khóa cá nhân (bí mật) mới có thể giải mã được.

1/.Tạo khóa

- Chọn 2 số nguyên tố lớn p và q với $\mathbf{p \neq q}$.
- Tính: $\mathbf{n = p * q}$;. Tính: giá trị hàm số Ơle: $\mathbf{\varnothing(n) = (p - 1) * (q - 1).}$
- Chọn số ngẫu nhiên b sao cho $\mathbf{1 < b < \varnothing(n)}$ và là số nguyên tố cùng nhau với $\mathbf{\varnothing n,}$ tức $\mathbf{gcd (b, \varnothing n) = 1.}$
- Tìm **a**, sao cho $\mathbf{a * b = 1 \bmod(\varnothing n)}$.
- Hình thành cặp khóa: K(K',K''), trong đó:

  K'= (n, b) là khóa công khai và **K''= (a)** là khóa bí mật.

2/. Mã hóa:

Mã hóa bản rõ *m* theo công thức:

$$\mathbf{c = m^{b} \ (mod\ n)} \quad \mathbf{(m \in Z_n)}$$

3/. Giải mã:

$$\mathbf{m = c^{a} \ (mod\ n).}$$

Chuyển đổi văn bản rõ:

Trước khi thực hiện mã hóa, ta phải thực hiện việc chuyển đổi văn bản rõ (chuyển đổi từ *M* sang *m*) sao cho không có giá trị nào của M tạo ra văn bản mã không an toàn. Nếu không có quá trình này, RSA sẽ gặp phải một số vấn đề sau:

- Nếu $m = 0$ hoặc $m = 1$ sẽ tạo ra các bản mã có giá trị là 0 và 1 tương ứng
- Khi mã hóa với số mũ nhỏ (chẳng hạn $e = 3$) và m cũng có giá trị nhỏ, giá trị m^e cũng nhận giá trị nhỏ (so với n). Như vậy phép modulo không có tác dụng và có thể dễ dàng tìm được m bằng cách khai căn bậc e của c (bỏ qua modulo).
- RSA là phương pháp mã hóa xác định (không có thành phần ngẫu nhiên) nên kẻ tấn công có thể thực hiện tấn công dựa vào bản rõ bằng cách tạo ra một bảng tra giữa bản rõ và bản mã. Khi gặp một bản mã, kẻ tấn công sử dụng bảng tra để tìm ra bản rõ tương ứng.

Một số tiêu chuẩn, chẳng hạn như PKCS, đã được thiết kế để chuyển đổi bản rõ trước khi mã hóa bằng RSA. Các phương pháp chuyển đổi này bổ sung thêm bít vào M. Các phương pháp chuyển đổi cần được thiết kế cẩn thận để tránh những dạng tấn công phức tạp tận dụng khả năng biết trước được cấu trúc của bản rõ. Phiên bản ban đầu của PKCS dùng một phương pháp đặc ứng (ad-hoc) mà về sau được biết là không an toàn trước tấn công lựa chọn bản rõ thích ứng(adaptive chosen ciphertext attack). Các phương pháp chuyển đổi hiện đại sử dụng các kỹ thuật như chuyển đổi mã hóa bất đối xứng tối ưu (Optimal Asymmetric Encryption Padding - OAEP) để chống lại tấn công dạng này. Tiêu chuẩn PKCS còn được bổ sung các tính năng khác để đảm bảo an toàn cho chữ ký RSA.

Chú ý:

RSA có tốc độ thực hiện chậm hơn đáng kể so với DES và các thuật toán mã hóa đối xứng khác. Trên thực tế, khi sử dụng thuật toán mã hóa đối xứng nào đó để mã hóa văn bản chỉ dùng RSA để mã hóa khóa để giải mã (thông thường khóa ngắn hơn nhiều so với văn bản).

4/. Vấn đề tấn công hệ mã hóa RSA

a/. Tìm cách xác định khóa bí mật:

Nếu kẻ tấn công tìm được 2 số nguyên tố p và q sao cho: $n = p * q$ thì có thể dễ dàng tìm được giá trị $(p - 1)*(q - 1)$ và qua đó xác định $\boldsymbol{a}$ theo công thức a * b = 1 mod ($\varnothing$**n**).

Khi tìm ra khóa bí mật thì chúng sẽ giải mã và tìm ra bản rõ.

→Cách phòng tránh: Chọn số nguyên tố p,q lớn để việc phân tích n thành 2 thừa số nguyên tố là khó có thể thực hiện được trong thời gian thực.

b/. Tấn công đứng giữa (Man-in-the-middle attack)

Cũng giống như các thuật toán mã hóa khác, cách thức phân phối khóa công khai là một trong những yếu tố quyết định đối với độ an toàn của RSA. Quá trình phân phối khóa cần chống lại được tấn công đứng giữa (*man-in-the-middle attack*). Giả sử Eve (kẻ tấn công) có thể gửi cho Bob một khóa bất kỳ và khiến Bob tin rằng đó là khóa (công khai) của Alice. Đồng thời Eve có khả năng đọc được thông tin trao đổi giữa Bob và Alice. Khi đó, Eve sẽ gửi cho Bob khóa công khai của chính mình (mà Bob nghĩ rằng đó là khóa của Alice). Sau đó, Eve đọc tất cả văn bản mã hóa do Bob gửi, giải mã với khóa bí mật của mình, giữ 1 bản copy đồng thời mã hóa bằng khóa công khai của Alice và gửi cho Alice. Về nguyên tắc, cả Bob và Alice đều không phát hiện ra sự can thiệp của người thứ ba. Các phương pháp chống lại dạng tấn công này thường dựa trên các ***chứng thực khóa công khai.***

c/. Tấn công dựa trên thời gian:

Vào năm 1995, Paul Kocher mô tả một dạng tấn công mới lên RSA: nếu kẻ tấn công nắm đủ thông tin về phần cứng thực hiện mã hóa và xác định được thời gian giải mã đối với một số bản mã lựa chọn thì có thể nhanh chóng tìm ra khóa $\boldsymbol{a}$. Dạng tấn công này có thể áp dụng đối với hệ thống chữ ký điện tử RSA.

Để chống lại tấn công dựa trên thời gian là đảm bảo quá trình giải mã luôn diễn ra trong thời gian không đổi bất kể bản mã. Tuy nhiên, cách này có thể làm giảm hiệu suất tính toán. Thay vào đó, hầu hết các ứng dụng hệ mật RSA sử dụng kỹ thuật gọi là che mắt.

Kỹ thuật này dựa trên tính nhân của RSA: Thay vì tính $\mathbf{c^a \bmod n}$, Alice đầu tiên chọn một số ngẫu nhiên **r** và tính $\mathbf{((r^b)* c)^d \bmod n.}$

Kết quả của phép tính này là (**r * m) mod n** và tác động của **r** sẽ được loại bỏ bằng cách nhân kết quả với nghịch đảo của **r**. Đối với mỗi văn bản mã, người ta chọn một giá trị của **r**. Vì vậy, thời gian giải mã sẽ không còn phụ thuộc vào giá trị của văn bản mã.

d/. Tấn công dùng modulo n chung.

Giả sử có 2 người tham gia A và B cùng sử dụng một modulo chung n trong khóa công khai của mình, chẳng hạn A chọn khóa công khai **(n, b)** và giữ khóa bí mật **a**. Người B chọn khóa công khai **(n, d)** và giữ khóa bí mật **c**. Một người tham gia thứ 3 gửi một văn bản cần bảo mật **x** đến cả A và B thì dùng khóa công khai nói trên để gửi đến A bản mã: $\mathbf{y = x^b \bmod n}$ và gửi đến B bản mã $\mathbf{z = x^d \bmod n.}$ Người thám mã O có thể dựa vào thông tin của n, b, d, y, z trên đường công khai để xác định ra bản rõ x như sau:

- Tính $c = b^{-1} \bmod d$;

-Tính $h = \frac{c * b - 1}{d}$; được $x = y^c (z^h)^{-1} \bmod n$.

Thực vậy, ta có

$$\mathbf{y^c (z^h)^{-1} \bmod n= (x^{cb})*(x^{d(cb-1)/d})^{-1} \bmod n = (x^{cb}) *(x^{cb-1})^{-1} \bmod n= x}$$

x là bản rõ cần tìm. Vậy trong trường hợp này, việc truyền tập tin bảo mật không còn an toàn nữa.

→Giải pháp phòng tránh:

Dùng *modulo n* khác nhau cho mỗi người tham gia.

2.1.5.3. Hệ mã hóa Elgamal

Được xây dựng trên bài toán khó giải logarit rời rạc.

Đặc trưng bài toán:

Cho một số nguyên tố p và phần tử sinh $\alpha \in Z_p$, $\beta \in Z^*_p$.

Hãy tìm một số nguyên duy nhất a, $0 \le a \le p - 2$, sao cho $\alpha^a \equiv \beta \pmod p$.

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

1/. Tạo cặp khóa (bí mật, công khai) (a, β):

Chọn số nguyên tố lớn **p** sao cho bài toán logarit rời rạc trong $\mathbf{Z_p}$ là "khó" giải.

Chọn phần tử nguyên thủy $\alpha \in Z^*_p$. Đặt $P = Z^*_p$, $C = Z^*_p \times Z^*_p$.

Chọn khóa bí mật $a \in Z^*_p$. Tính khóa công khai $\beta = \alpha^a \pmod p$.

Định nghĩa tập khóa: $K = \{ (\alpha, \beta, a, p\}: \beta = \alpha^a \pmod p) \}$

Các giá trị **α, β, p** được công khai, phải giữ bí mật **a**.

Với bản rõ $x \in P$ và bản mã $y \in C$, với khóa $k \in K$ định nghĩa:

2/. Lập mã:

Chọn số ngẫu nhiên dương bí mật $t (0 \le t \le p - 2)$

Bản mã là $y = e_k (x, t) = (y_1, y_2)$

trong đó $y_1 = \alpha^t \bmod p$ và $y_2 = x * \beta^t \bmod p$

3/. Giải mã:

$$d_{k"}(y_1, y_2) = y_2 * (y_1^a)^{-1} (\bmod p).$$

Mô tả sơ lược cách làm việc của hệ mã Elgamal:

Bản mã x được che dấu bằng cách nhân nó với β^t để tạo y_2, giá trị α^t cũng được gửi đi như một phần của bản mã. Người biết được số mũ bí mật a có thể tính được β^t từ α^t. Sau đó anh ta sẽ tháo mặt nạ bằng cách chia y_2 cho β^t để thu được x.

Chú ý: $y^{-a} = y^{p-1-a}$.

Ví dụ:

Cho p = 13, $\alpha = 2$, a = 3. Khi đó

$\beta = 2^3 \bmod 13 = 8$

Bây giờ ta giả sử B muốn gửi thông báo x = 7 tới A. Giả sử số ngẫu nhiên mà B chọn là k =1. Sau đó B tính

$y_1 = 2^1 \bmod 13 = 2$

$y_2 = 7 \times 8^1 \bmod 13 = 4$

Khi đó A thu được bản mã y = (2, 4), A tính $x = 4 \times 2^{13-1-3} \bmod 13 = 7$

Đó chính là bản rõ mà B đã mã hóa.

4/. Thám mã với hệ mã Elgamal

a/. Thuật toán Shank

Thuật toán này còn có tên gọi khác là thuật toán cân bằng thời gian – bộ nhớ (Time memory Trade Off), có nghĩa là nếu chúng ta có đủ bộ nhớ thì có thể dùng bộ nhớ đó để giảm thời gian thực hiện của bài toán xuống.

Input: số nguyên p, phần tử sinh nguyên thủy $\alpha \in Z^*_p$, số nguyên β.

Output: cần tìm a sao cho $\alpha^a \bmod p = \beta$.

Thuật toán:

Gọi $m=[(p - 1)^{1/2}]$ (lấy phần nguyên).

Bước 1: Tính $\alpha^{mj} \bmod p$ với $0 \le j \le m - 1$.

Bước 2 : Sắp xếp các cặp $(j, \alpha^{mj} \bmod p)$ và lưu vào trong danh sách L_1.

Bước 3: Tính $\beta \alpha^{-i} \bmod p$ với $0 \le i \le m - 1$.

Bước 4: Sắp xếp các cặp $(i , \beta \alpha^{-i} \bmod p)$ và lưu vào danh sách L_2.

Bước 5: Tìm trong 2 danh sách L_1 và L_2 và xem có tồn tại cặp $(j, \alpha^{mj} \bmod p)$ và $(i, \beta\alpha^{-1} \bmod p)$ nào mà $\alpha^{mj} \bmod p = \beta \alpha^{-i} \bmod p$ (tọa độ thứ 2 của 2 cặp bằng nhau).

Bước 6: $a = (m j + i) \bmod (p – 1)$. Kết quả này có thể kiểm chứng từ công thức $\alpha^{mj} \bmod p = \beta \alpha^{-i} \bmod p \Rightarrow \alpha^{mj+i} \bmod p = \beta \bmod p \Rightarrow a = (mj+i) \bmod (p-1)$.

Độ phức tạp của thuật toán phụ thuộc vào m = $[(p-1)^{1/2}]$, với giá trị của m, chúng ta cần tính các phần tử thuộc 2 danh sách L_1 và L_2 đều là các phép toán lũy thừa phụ thuộc vào j và i, i và j lại phụ thuộc vào m nên có thể nhận thấy là thuật toán này có thể áp dụng trong trường hợp mà p nhỏ.

Ví dụ:

Cho p=13, α=2; β=8. Tìm a sao cho $\alpha^a \bmod p=\beta$.

Giải:

Tính: $m=[(p-1)^{1/2}]=3$.

B1: Tính $\alpha^{mj} \bmod p$ với $0 \le j \le m - 1$

Với j= 0 thì $\alpha^{mj} \bmod p=2^{3*0} \bmod 13=1$

Với j= 1 thì $\alpha^{mj} \bmod p=2^{3*1} \bmod 13=8$

Với j= 2 thì $\alpha^{mj} \bmod p=2^{3*2} \bmod 13=12$

B2: Lưu vào danh sách L_1 -> Vậy ta có L_1= {(0,1);(1,8);(2,12)}

B3: Tính $\beta\alpha^{-i} \bmod p$ với $0 \le i \le m - 1$

Với i=0 thì $\beta \alpha^{-i} \bmod p=8 * 2^{-0} \bmod 13=8$;

Với i=1 thì $\beta \alpha^{-i} \bmod p=8 * 2^{-1} \bmod 13=4$;

Với i=2 thì $\beta \alpha^{-i} \bmod p= 8 * 2^{-2} \bmod 13=2$;

B4: lưu vào danh sách L_2-> L_2={(0,8);(1,4);(2,2)}

B5: Tìm trong 2 danh sách L_1 và L_2 cặp (j, $\alpha^{mj} \bmod p$) và (i, $\beta \alpha^{-1} \bmod p$) nào mà có $\alpha^{m*j} (\bmod p) = \beta * \alpha^{-i} (\bmod p)$ (tọa độ thứ 2 của 2 cặp bằng nhau).

Ta thấy cặp (1, 8) của L_1 và (0, 8) của L_2 có tọa độ thứ 2 bằng nhau.

B6: vậy a= (3 * 1 + 0) mod (13-1)=3

b/. Tìm cách xác định khóa bí mật

Sử dụng modulo p nhỏ

Khi sử dụng số nguyên tố p nhỏ thì tập Z^*_p nhỏ, do đó việc tìm được phần tử sinh $\alpha \in Z^*_p$ cũng không khó khăn lắm. Khi biết được α và giá trị β từ khóa công khai thám mã có thể biết được khóa bí mật a.

→Giải pháp phòng tránh: Chọn modulo p là số nguyên tố sao cho p -1 có ít nhất một số nguyên tố lớn. Điều đó là thực hiện được nếu số nguyên tố p được chọn là số nguyên tố Sophie Germain (tức có dạng 2 q + 1, với q cũng là số nguyên tố lớn).

c/. Tìm cách xác định bản rõ

Dùng cùng một số k trong nhiều lần lập mã: giả sử dùng cùng một số ngẫu nhiên k cho 2 lần lập mã, một lần cho x_1, một lần cho x_2 và được các bản mã tương ứng (y_1,y_2) và (z_1, z_2).

Vì dùng cùng một k nên $y_1 = z_1$ và do đó theo công thức lập mã ta có:

$z_2 / y_2 = x_2 / x_1$, tức là $x_2 = x_1 * z_2 / y_2$. Như vậy, một người thám mã, một lần biết cả bản rõ dễ dàng phát hiện bản rõ trong các lần sau.

→Giải pháp phòng tránh: Mỗi lần lập mã thì sử dụng một số k khác nhau.

2.2. CHỮ KÍ SỐ

2.2.1. Sơ đồ chữ kí số

Sơ đồ chữ ký là bộ 5: (P, A, K, S, V).Trong đó:

- P là tập hữu hạn các văn bản có thể.
- A là một tập hữu hạn các chữ ký có thể.
- K là tập hữu hạn các khoá có thể.
- S là tập hữu hạn các thuật toán ký
- V là tập hữu hạn các thuật toán kiểm thử.

Với mỗi khóa k ∈ K, có thuật toán ký **$Sig_k \in S$, $Sig_k : P \rightarrow A$,**

có thuật toán kiểm tra chữ ký **$Ver_k \in V$, $P \times A \rightarrow$** {đúng, sai},

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

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

2.2.2. Chữ kí RSA

1/. Sơ đồ chữ ký RSA

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

Chọn bí mật 2 số nguyên tố lớn p, q, tính n = p * q, công khai n, đặt P = A = Z_n.

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

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

Tập cặp khóa (bí mật, công khai) K = {(a, b) / a,b ∈ $\mathbf{Z_n}$, a * b ≡ 1 mod (∅ (n))}.

- **Ký số**:

  Chữ ký trên x ∈ **P** là y = sig k (x) = x a (mod n), y ∈A.

- **Kiểm tra chữ ký**:

  ver k (x, y) = đúng ⇔ x ≡ y b (mod n). (x∈P, ∀ y∈A)

2/. Ví dụ

a/. Tạo khóa:

Cho p = 3, q = 5, n = p * q =3 * 5 = 15

∅ (n) = (p - 1).(q - 1) = (3 - 1). (5 - 1) = 2 * 4 = 8

Chọn b = 3 là nguyên tố cùng nhau với ∅ (15)

Vậy theo công thức a * b ≡ 1 mod (∅ (n)) ta có a = 3

Khóa K (K' , K") với K' = (a) = (3) và K" = (b, n) = (3, 15)

b/. Ký số

Giả sử ký trên x = 2

Hàm ký: $y = sig_{k'} (x) = x^{a} (mod\ n) = 2^{3} (mod\ 15) = 8$

c/. Kiểm tra chữ ký

$ver_{k''} (x, y) = ver_{k"} (2, 8)$ = Đúng

vì y^{b} (mod n).= 8^{3} (mod 15) = 2 = x

2.2.3. Chữ kí Elgamal

1/. Sơ đồ chữ kí Elgamal

- ***Tạo cặp khóa (bí mật, công khai) (a, β)***

Chọn số nguyên tố lớn **p** sao cho bài toán logarit rời rạc trong $\mathbf{Z_p}$ là "khó" giải.

Chọn phần tử nguyên thủy $\alpha \in Z^*_p$. Đặt $\mathbf{P = Z_p^*}$, $\mathbf{A = Z_p^* \times Z_{p-1}^*}$.

Chọn khóa bí mật $a \in Z_p^*$. Tính khóa công khai $\beta = \alpha^a \pmod p$.

Định nghĩa tập khóa**:** $K = \{ (\alpha, \beta, a, p\}: \beta = \alpha^a \pmod p) \}$

Các giá trị **α, β, p** được công khai, phải giữ bí mật **a**.

- ***Ký số***

Dùng 2 khóa ký: khóa a và 1 khóa ngẫu nhiên bí mật t ($t \in Z_{p-1}^*$)

Chữ ký trên $x \in \mathbf{P}$ là $y = sig_k (x, t) = (\gamma, \delta)$, $y \in A$

$y = \alpha^t \bmod p$ và $\delta = (x - a\,y) * t^{-1} \bmod (p - 1)$ với $x,y \in Z^*p$, $\delta \in Zp\text{-}1$

- **Kiểm tra chữ ký**

Verk (x ,(y, δ)) = đúng khi và chỉ khi: $(\beta)^y * y^\delta \equiv \alpha^x \pmod p$

Nếu chữ kí được thiết lập đúng thì xác minh sẽ thành công vì:

$$\beta^y y^\delta \equiv \alpha^{ay} * \alpha^{t(x-ay)*t^{-1}} \equiv \alpha^x$$

2/. Ví dụ

Giả sử : Cho p =467 , α=2 , a=127 khi đó:

$\beta = \alpha^a \bmod p = 2^{127} \bmod 467 = 132.$

Nếu Bob muốn ký trên bức điện x = 100 và chọn số ngẫu nhiên k = 213 (chú ý là USCLN(213,466) =1 và $213^{-1} \bmod 466 = 431$). Khi đó :

$y = 2^{213} \bmod 467 = 29$ và $\delta = (100 - 127 \times 29)\, 431 \bmod 466 = 51$

Bất kì ai cũng có thể xác minh chữ kí này bằng cách kiểm tra:

$132^{29} * 29^{51} \equiv 189 \pmod{467}$ và $2^{100} \equiv 189 \pmod{467}$

Vì thế chữ kí là hợp lệ.

2.3. HÀM BĂM

2.3.1. Tổng quan hàm băm

2.3.1.1. Khái niệm Hàm băm

Hàm băm là thuật toán không dùng khóa để mã hóa, nó có nhiệm vụ lọc (băm) tài liệu và cho kết quả là một giá trị "băm" có kích thước cố định, gọi là " đại diện tài liệu" hay "đại diện thông điệp".

Hàm băm là hàm một chiều, theo nghĩa giá trị của hàm băm là duy nhất, và từ giá trị băm này, "khó thể" suy ngược lại nội dung hay độ dài ban đầu của tài liệu gốc.

2.3.1.2. Đặc tính của hàm băm

Hàm băm h là hàm một chiều với các đặc tính sau:

1/. Với tài liệu đầu vào (bản tin gốc) x, chỉ thu được giá trị băm duy nhất z = h (x).

2/. Nếu dữ liệu trong bản tin x bị thay đổi hay bị xóa để thành bản tin x', thì giá trị băm h (x') ≠ h (x). Cho dù là một thay đổi nhỏ, ví dụ thay đổi một bit dữ liệu của bản tin gốc x, thì giá trị băm h (x) của nó cũng vẫn thay đổi. Điều này có nghĩa là: hai thông điệp khác nhau thì giá trị băm của chúng cũng khác nhau.

3/. Nội dung của bản tin gốc " khó" thể suy ra từ giá trị băm của nó. Nghĩa là: với thông điệp x thì "dễ" tính được z = h (x), nhưng lại "khó" tính ngược lại được x nếu chỉ biết giá trị băm h (x).

2.3.1.3. Các tính chất của Hàm băm

Tính chất 1: Hàm băm h là không va chạm yếu**.**

Hàm băm h được gọi là không va chạm yếu, nếu cho trước bản tin x, khó có thể tính toán để tìm ra x' ≠ x mà h (x') = h (x).

Tính chất 2: Hàm băm h là không va chạm mạnh.

Hàm băm h được gọi là không va chạm mạnh nếu "khó " thể tính toán để tìm ra hai bức thông điệp khác nhau x' và x (x' ≠ x) mà có h (x') = h(x).

Tính chất 3: Hàm băm h là hàm một chiều.

Hàm băm h được gọi là hàm một chiều nếu khi cho trước một bản tóm lược thông báo z thì "khó" thể tính toán để tìm ra thông điệp ban đầu x sao cho h (x) = z.

2.3.2. Hàm băm MD4

2.3.2.1. Khái niệm "Thông điệp đệm"

"***Thông điệp đệm***" là xâu bit có độ dài chia hết cho 512.

"Thông điệp đệm" được lưu trong mảng M = M [0] M[1] … M[N-1]

Trong đó M[i] là xâu bit có độ dài 32 bit, gọi là word.

$N \equiv 0 \bmod 16$. (32 * 16 = 512)

M được xây dựng từ bản tin gốc a bằng thuật toán:

1/. $d = 447 - (|a| \bmod 512)$

2/. Giả sử **l** là kí hiệu biểu diễn nhị phân của $|a| \bmod 2^{64}$

3/. $M = a \,||\, 1 \,||\, 0^d \,||\, \mathbf{l}$

Độ dài của xâu $M = a \,||\, 1 \,||\, 0^d$ là $|a| + 1 + d = 448 \bmod 512$.

Độ dài của thông điệp đệm M là:

$448 \bmod 512 + |\mathbf{l}| = 448 \bmod 512 + 64 = 512 \bmod 512$

Chú ý: Vì $M = a \,||\, 1 \,||\, 0^d \,||\, \mathbf{l}$ nên

$d = |M| - (|a| + 1 + 1)$

$= |512| - (|a| + 1 + 64) = 512 - (|a| + 65) = 447 - (|a| \bmod 512)$.

2.3.2.2. Thuật toán MD4

Input: Thông điệp là một xâu a có độ dài b bit.

Output: Bản băm đại diện cho thông điệp gốc, có độ dài cố định 128 bit.

1/. Tóm tắt thuật toán

Bước 1: khởi tạo thanh ghi.

Có 4 thanh ghi để tính toán nhằm đưa ra các đoạn mã: A, B, C, D. Mỗi thanh ghi có độ dài 32 bit. Các thanh ghi này được khởi tạo giá trị hexa.

Word A : = 67 45 23 01 word C : = 98 ba dc fe

Word B : = ef cd ab 89 word D : = 10 32 54 76

Bước 2:

Xử lý thông điệp a trong 16 khối word, có nghĩa là cùng một lúc xử lý 16 word là 512 bit.

Chia mảng M thành các khối 512 bit, đưa từng khối 512 bit vào mảng T [j]. Mỗi lần xử lý một khối 512 bit. Lặp lại N/16 lần.

2/. Thuật toán MD4

A : = 67 45 23 01 C : = 98 ba dc fe

B : = ef cd ab 89 D : = 10 32 54 76

/* Xử lý thông điệp theo từng khối 16 word*/

For i : = 0 to N/16 - 1 do

For j := 0 to 15 do /* Copy khối i vào trong T. */

T [j] = M [16 i + j]; /* lặp trên j */

end

/* Lưu A vào AA, B vào BB, C vào CC, D và DD . */

AA = A

BB = B

CC = C

DD = D

/*mỗi lần xử lý 16 tử, mỗi từ 32 bit*/

Vòng 1

Vòng 2

Vòng 3

/* cộng vào mỗi thanh ghi giá trị của nó trước khi vào vòng lặp */

A = A + AA

B = B + BB

C = C + CC

D = D + DD

end /* lặp trên i */

3/. Các phép tính và các hàm dùng trong thuật toán MD4

Các phép toán logic được sử dụng trong 3 vòng

X $\wedge$ Y là phép toán AND theo bit giữa X và Y

X $\vee$ Y là phép toán OR theo bit giữa X và Y.

X $\oplus$ Y là phép toán XOR theo bit giữa X và Y

– X chỉ phép bù của X

X + Y là phép cộng theo modulo 2^{32}

X <<< s là phép dịch vòng trái X đi s vị trí ($0 \leq s \leq 31$)

3 hàm F, G, H dùng tương ứng trong vòng 1, 2, 3.

Mỗi hàm này là một hàm logic tính theo bít.

$F(X, Y, Z) = (X \wedge Y) \vee ((- X) \wedge Z)$

$G(X, Y, Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$

$H(X, Y, Z) = X \oplus Y \oplus Z$

4/. Ba vòng "băm"

Vòng 1

1.	A = (A +F (B, C, D) + T [0]) <<< 3
2.	D = (D +F (A, B, C) + T [1]) <<< 7
3.	C= (C +F (D, A, B) + T [2]) <<< 11
4.	B = (B +F (C, D, A) + T [3]) <<< 19
5.	A = (A +F (B, C, D) + T [4]) <<< 3
6.	D = (D +F (A, B, C) + T [5]) <<< 7
7.	C= (C +F (D, A, B) + T [6]) <<< 11
8.	B = (B +F (C, D, A) + T [7]) <<< 19
9.	A = (A +F (B, C, D) + T [8]) <<< 3
10.	D = (D +F (A, B, C) + T [9]) <<< 7
11.	C= (C +F (D, A, B) + T [10]) <<< 11
12.	B = (B +F (C, D, A) + T [11]) <<< 19
13.	A = (A +F (B, C, D) + T [12]) <<< 3
14.	D = (D +F (A, B, C) + T [13]) <<< 7
15.	C= (C +F (D, A, B) + T [14]) <<< 11
16.	B = (B +F (C, D, A) + T [15]) <<< 19

Vòng 2

1. A = (A + G (B, C, D) + T [0] +5A827999) <<< 3
2. D = (D + G (A, B, C) + T [4] + 5A827999) <<< 5
3. C = (C + G (D, A, B) + T [8] + 5A827999) <<< 9
4. B = (B + G (C, D, A) + T [12] + 5A827999) <<< 13
5. A = (A + G (B, C, D) + T [1] +5A827999) <<< 3
6. D = (D + G (A, B, C) + T [5] + 5A827999) <<< 5
7. C = (C + G (D, A, B) + T [9] + 5A827999) <<< 9
8. B = (B + G (C, D, A) + T [13] + 5A827999) <<< 13
9. A = (A + G (B, C, D) + T [2] +5A827999) <<< 3
10. D = (D + G (A, B, C) + T [6] + 5A827999) <<< 5
11. C = (C + G (D, A, B) + T [10] + 5A827999) <<< 9
12. B = (B + G (C, D, A) + T [14] + 5A827999) <<< 13
13. A = (A + G (B, C, D) + T [3] +5A827999) <<< 3
14. D = (D + G (A, B, C) + T [7] + 5A827999) <<< 5
15. C = (C + G (D, A, B) + T [11] + 5A827999) <<< 9
16. B = (B + G (C, D, A) + T [15] + 5A827999) <<< 13

Giá trị: 5A827999 là một hằng số ở dạng hecxa có độ dài 32 bit.

Vòng 3

A = (A + H (B, C, D) + T[0] + 6ED9EBA1)<<< 3
D = (D + H (A, B, C) + T[8] + 6ED9EBA1)<<< 9
C = (C + H (D, A, B) + T[4] + 6ED9EBA1)<<< 11
B = (B + H (C, D, A) + T[12] + 6ED9EBA1)<<< 15
A = (A + H (B, C, D) + T[2] + 6ED9EBA1)<<< 3
D = (D + H (A, B, C) + T[10] + 6ED9EBA1)<<< 9
C = (C + H (D, A, B) + T[6] + 6ED9EBA1)<<< 11
B = (B + H (C, D, A) + T[14] + 6ED9EBA1)<<< 15
A = (A + H (B, C, D) + T[1] + 6ED9EBA1)<<< 3
D = (D + H (A, B, C) + T[9] + 6ED9EBA1)<<< 9
C = (C + H (D, A, B) + T[5] + 6ED9EBA1)<<< 11
B = (B + H (C, D, A) + T[13] + 6ED9EBA1)<<< 15
A = (A + H (B, C, D) + T[3] + 6ED9EBA1)<<< 3
D = (D + H (A, B, C) + T[11] + 6ED9EBA1)<<< 9
C = (C + H (D, A, B) + T[7] + 6ED9EBA1)<<< 11
B = (B + H (C, D, A) + T[15] + 6ED9EBA1)<<< 15

Giải thích: 6ED9EBA1 là một hằng số ở dạng hexa có độ dài 32 bit

5/. Kết quả "băm"

Kết quả ra là đoạn mã có độ dài 128 bit, được thu gọn từ thông điệp a có độ dài b bit. Đoạn mã này thu được từ 4 thanh ghi A, B, C, D bắt đầu từ byte thấp của thanh ghi A đến byte cao của thanh ghi D.

Ví dụ: a = "ABC "

Kết quả cuối cùng là đại diện văn bản

A = 6A8CA15F

B = 671E4A93

C = 93F85626

D = 3409907C

2.3.3. Hàm băm SHA

2.3.2.1. Giới thiệu

SHA (Secure Hash Algorithm hay thuật giải băm an toàn)

Năm thuật giải SHA là *SHA-1* (trả lại kết quả dài 160 bit), *SHA-224* (trả lại kết quả dài 224 bit), *SHA-256* (trả lại kết quả dài 256 bit), *SHA-384* (trả lại kết quả dài 384 bit), và *SHA-512* (trả lại kết quả dài 512 bit).

Thuật giải SHA là thuật giải băm mật được phát triển bởi cục an ninh quốc gia (National Security Agency hay NSA) và được xuất bản thành chuẩn của chính phủ Mĩ bởi viện công nghệ và chuẩn quốc gia Mĩ (National Institute of Standards and Technology hay NIST). Bốn thuật giải sau thường được gọi chung là *SHA-2*.

SHA-1 được dùng trong nhiều ứng dụng và giao thức an ninh khác nhau, bao gồm TLS và SSL,PGP,SSH và IPSec. SHA-1 được coi là thuật giải thay thế MD5.

SHA-2 bao gồm bốn giải thuật SHA-224, SHA-256, SHA-384 và SHA-512. Ba thuật giải SHA-256, SHA-384 và SHA-512 được xuất bản lần đầu năm 2001.

Về giải thuật, các biến thể của SHA-2 không khác nhau. Mặc dù chúng sử dụng giá trị biến và hằng số cũng như độ dài từ, v.v. khác nhau.

Mặc dù Gilbert và Handschuh (2003) đã nghiên cứu và không tìm ra điểm yếu của những biến thể này, chúng vẫn chưa được kiểm chứng kĩ như SHA-1.

2.3.3.2. Thuật toán

1/. Khởi gán các biến:

h0 := 0x67452301

h1 := 0xEFCDAB89

h2 := 0x98BADCFE

h3 := 0x10325476

h4 := 0xC3D2E1F0

2/. Tiền xử lý

Thêm bit 1 vào cuối thông điệp

Thêm vào k bit 0 sao cho độ dài thông điệp nhận được đồng dư 448 (mod 512)

Thêm 64 bit biểu diễn độ dài dài của thông điệp gốc.

Chia thông điệp thành các khối 512 bit

Với mỗi khối 512-bit:

Chia thành 16 word (32 bit) w[0..15]

Mở rộng 16 word (32 bit) thành 80 word (32 bit)

w [i]=(w[i - 3] ⊕ w [i – 8] ⊕ w[i -14] ⊕ w [i - 16]) <<< 1 với $16 \leq i < 80$

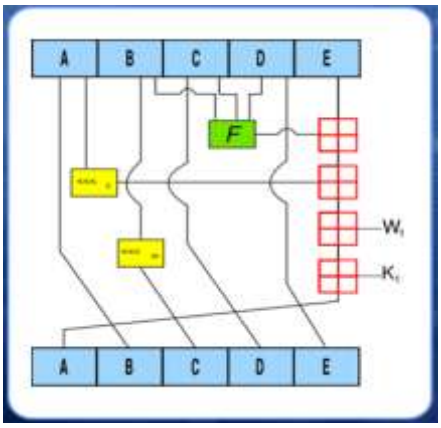
A= h0, B= h1, C= h2, D= h3, E= h4

80 chu kỳ xử lý

h0+= A, h1+= B, h2+= C, h3+= D, h4+= E

Kết quả:= h0 | h1 | h2 | h3 | h4

3/. Chu kỳ xử lý trong SHA1



t là số thứ tự của chu kỳ

A, B, C, D, E là 5 word (32 bit) của trạng thái

F là hàm phi tuyến (thay đổi tùy theo chu kỳ)

<<< *n* là phép quay trái *n* vị trí

⊞ phép cộng modulo 2^{32}, K_t là hằng số

for *i* **from** 0 to 79

$f = F[t]$ (B, C, D)

temp = (A <<< 5) + f + E + K_t + w [i]

E = D

D = C

C = B <<< 30

B = A

A = temp

Giá trị của K:

$$K_t = \begin{cases} 0x5a827999, 0 \le t \le 19 \\ 0x6ed9eba1, 20 \le t \le 39 \\ 0x8f1bbcdc, 40 \le t \le 59 \\ 0xca62c1d6, 60 \le t \le 79 \end{cases}$$

Công thức của hàm F [t] có thể được viết lại như sau:

$$F[t](X,Y,Z) = \begin{cases} Z \oplus (X \wedge (Y \oplus Z)), & 0 \le t \le 19 \\ (X \wedge Y) \vee (Z \wedge (X \vee Y)), & 20 \le t \le 39 \\ (X \wedge Y) \vee (Z \wedge (X \oplus Y)), & 40 \le t \le 59 \\ (X \wedge Y) + (Z \wedge (X \oplus Y)), & 60 \le t \le 79 \end{cases}$$

Chương 3. MỘT SỐ DẠNG "TẤN CÔNG" HỆ THỐNG THÔNG TIN

3.1. TẤN CÔNG MẠNG MÁY TÍNH VÀ CÁCH PHÒNG CHỐNG

3.1.1. Một số dạng tấn công mạng máy tính

3.1.1.1. Kỹ thuật đánh lừa

Đây là thủ thuật được nhiều kẻ tấn công dùng cho các cuộc tấn công và thâm nhập vào hệ thống mạng và máy tính, bởi tính đơn giản mà hiệu quả của nó. Thường được sử dụng để lấy cấp mật khẩu, thông tin, tấn công vào và phá hủy hệ thống.

Ví dụ : Kỹ thuật đánh lừa Fake Email Login.

Về nguyên tắc, mỗi khi đăng nhập vào hộp thư, thì chúng ta phải nhập thông tin tài khoản của mình bao gồm username (tên người dùng) và password (mật khẩu) rồi gửi thông tin đến Mail Server xử lý. Lợi dụng việc này, kẻ tấn công đã thiết kế trang web giống hệt như trang đăng nhập mà chúng ta hay sử dụng. Tuy nhiên, đó là một trang web giả và tất cả thông tin mà chúng ta điền vào đều được gởi đến cho họ. Kết quả, chúng ta bị đánh cắp mật khẩu.

Nếu là người quản trị mạng, chúng ta nên chú ý và dè chừng trước những email, những tin nhắn, các cú điện thoại yêu cầu khai báo thông tin. Những mối quan hệ cá nhân hay những cuộc tiếp xúc đều là một mối nguy hiểm tiềm tàng.

3.1.1.2. Kỹ thuật tấn công vào vùng ẩn

Những phần bị dấu đi trong các website thường chứa những thông tin về phiên làm việc của các client (máy khách). Các phiên làm việc này thường được ghi lại ở máy khách chứ không tổ chức cơ sở dữ liệu trên máy chủ. Vì vậy, kẻ tấn công có thể sử dụng chiêu chức View Source của trình duyệt để đọc phần đầu đi này và từ đó có thể tìm ra các sơ hở của trang Web mà họ muốn tấn công. Từ đó, có thể tấn công vào hệ thống máy chủ.

3.1.1.3. Nghe trộm

Các hệ thống truyền đạt thông tin qua mạng đôi khi không chắc chắn lắm và lợi dụng điều này, kẻ tấn công có thể truy cập vào các đường dẫn dữ liệu để nghe trộm hoặc đọc trộm luồng dữ liệu truyền qua.

Kẻ tấn công nghe trộm sự truyền đạt thông tin, dữ liệu sẽ chuyển đến kẻ trộm. Nó sẽ thu thập những thông tin quý giá về hệ thống như một gói chứa mật khẩu và tên người dùng của một ai đó. Các chương trình nghe trộm còn được gọi là các sniffing. Các sniffing này có nhiệm vụ lắng nghe các cổng của một hệ thống mà kẻ tấn công muốn nghe trộm. Nó sẽ thu thập dữ liệu trên các cổng này và chuyển về cho kẻ tấn công

3.1.1.4. Tấn công Man-in-the-Middle – Giả mạo DNS

Mỗi truy vấn DNS được gửi qua mạng đều có chứa một số nhận dạng duy nhất, mục đích của số nhận dạng này là để phân biệt các truy vấn và đáp trả chúng. Điều này có nghĩa rằng nếu một máy tính đang tấn công của chúng ta có thể chặn một truy vấn DNS nào đó được gửi đi từ một thiết bị cụ thể, thì tất cả những gì chúng ta cần thực hiện là tạo một gói giả mạo có chứa số nhận dạng đó để gói dữ liệu đó được chấp nhận bởi mục tiêu.

Chúng ta sẽ hoàn tất quá trình này bằng cách thực hiện hai bước với một công cụ đơn giản. Đầu tiên, chúng ta cần giả mạo ARP cache thiết bị mục tiêu để định tuyến lại lưu lượng của nó qua các máy chủ đang tấn công của mình, từ đó có thể chặn yêu cầu DNS và gửi đi gói dữ liệu giả mạo. Mục đích của kịch bản này là lừa người dùng trong mạng mục tiêu truy cập vào website độc thay vì website mà họ đang cố gắng truy cập. Để rõ hơn chúng ta có thể tham khảo thêm hình tấn công bên dưới.

Việc tạo ra một kiểu tấn công mới là mục đích của các kẻ tến công. Trên mạng Internet hiện nay.

3.1.2. Phòng chống tấn công qua mạng bằng kĩ thuật mật mã

3.1.2.1. Phương pháp mã hóa

Với *phương pháp mã hóa* thì trước khi được truyền trên mạng, dữ liệu đã được mã hóa để đảm bảo trong quá trình truyền đi dữ liệu đó không bị xem trộm.

Giao thức SSL (Secure Socket Layer) tổ hợp nhiều giải thuật mã hóa nhằm đảm bảo quá trình trao đổi thông tin trên mạng được bảo mật. Việc mã hóa dữ liệu diễn ra một cách trong suốt, hỗ trợ nhiều giao thức khác chạy trên nền giao thức TCP.

Cơ chế hoạt động của giao thức SSL dựa trên nền tảng các ứng dụng mã hóa đã được kiểm chứng như: giải thuật mã hóa đối xứng như DES, 3 DES và mã hóa bất đối xứng như RSA, Elgamal, giải thuật băm (hash) một chiều, v.v...

3.1.2.2. Phương pháp chứng thực khóa công khai

Ví dụ, A muốn gửi thông điệp cho B và mã hóa theo phương pháp khóa công khai. Lúc này A cần phải mã hóa thông điệp bằng khóa công khai của B. Trường hợp khóa công khai bị giả mạo thì sao? Kẻ tấn công có thể tự sinh ra một cặp khóa gồm khóa công khai và khóa bí mật, sau đó đưa cho A khóa công khai này và nói đây là khóa công khai của B. Nếu A dùng khóa công khai giả này mà tưởng là của B thì dẫn đến hệ quả mọi thông tin A truyền đi đều bị kẻ tấn công đọc được.

Vấn đề này được giải quyết nếu có một bên thứ ba được tin cậy, gọi là C, đứng ra chứng nhận khóa công khai. Những khóa công khai đã được C chứng nhận gọi là chứng nhận điện tử (public key certificate hay digital certificate).

Một chứng nhận điện tử có thể được xem như là một "hộ chiếu" hay "chứng minh thư". Nó được một tổ chức tin cậy (như VeriSign, Entrust, CyberTrust, v.v...) tạo ra. Tổ chức này được gọi là tổ chức chứng nhận khóa công khai Certificate Authority (CA). Một khi khóa công khai đã được CA chứng nhận thì có thể dùng khóa đó để trao đổi dữ liệu trên mạng với mức độ bảo mật cao.

Cấu trúc của một chứng nhận điện tử gồm các thành phần chính như sau:

+ Tên của CA tạo ra chứng nhận.

+ Ngày hết hạn của chứng nhận.

+ Những thông tin về thực thể được chứng nhận.

+ Khóa công khai được chứng nhận.

+ Chữ kí: do khóa bí mật của CA tạo ra và đảm bảo giá trị của chứng nhận

3.2. TẤN CÔNG HỆ ĐIỀU HÀNH VÀ CÁCH PHÒNG CHỐNG

3.2.1. Một số dạng tấn công hệ điều hành

3.2.1.1. Tấn công vào hệ thống có cấu hình không an toàn

Cấu hình không an toàn cũng là một lỗ hổng bảo mật của hệ thống. Các lỗ hổng này được tạo ra do các ứng dụng có các thiết lập không an toàn hoặc người quản trị hệ thống định cấu hình không an toàn. Chẳng hạn như cấu hình máy chủ web cho phép ai cũng có quyền duyệt qua hệ thống thư mục. Việc thiết lập như trên có thể làm lộ các thông tin nhậy cảm như mã nguồn, mật khẩu hay các thông tin của khách hàng.

Nếu quản trị hệ thống cấu hình hệ thống không an toàn sẽ rất nguy hiểm vì nếu người tấn công duyệt qua được các tệp tin mật khẩu thì họ có thể tải về và giải mã ra, khi đó họ có thể làm được nhiều thứ trên hệ thống.

3.2.1.2. Tấn công mật khẩu cơ bản (Password-base Attact)

Thông thường, hệ thống khi mới cấu hình có tên người dùng và mật khẩu mặc định. Sau khi cấu hình hệ thống, một số người quản trị vẫn không đổi lại các thiết lập mặc định này. Đây là lỗ hổng giúp kẻ tấn công có thể thâm nhập vào hệ thống bằng con đường hợp pháp. Khi đã đăng nhập vào, kẻ tấn công có thể tạo thêm người dùng, cài cửa sau (backboor) cho lần viếng thăm sau.

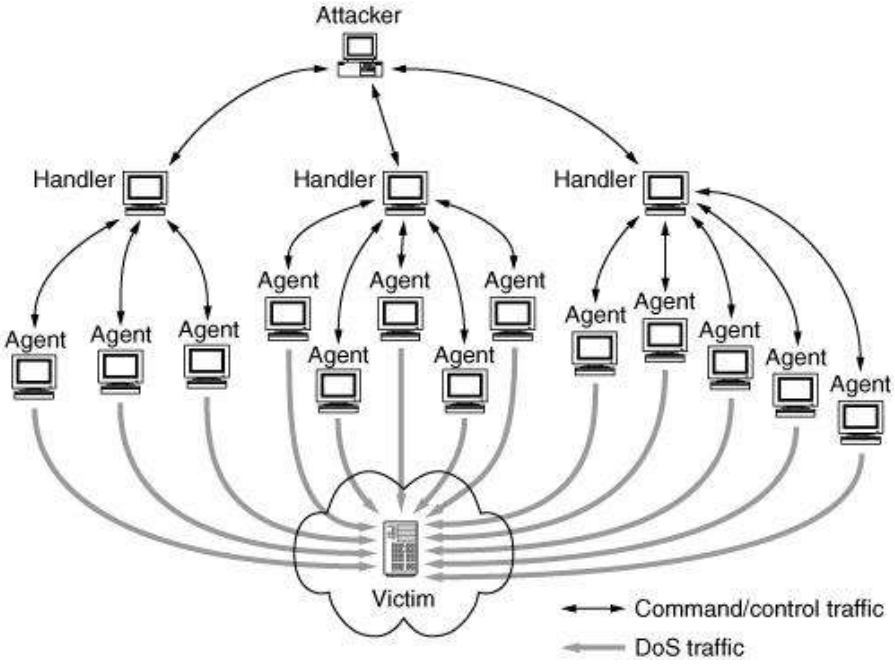
3.2.1.3. Tấn công từ chối dịch vụ (DoS)

Tấn công từ chối dịch vụ (DoS) là cuộc tấn công trên hệ thống mạng nhằm ngăn cản những truy xuất tới một dịch vụ như là WEB, Email,… Tấn công DoS phá huỷ dịch vụ mạng bằng cách làm tràn ngập số lượng kết nối, quá tải máy chủ hoặc chương trình chạy trên máy chủ, tiêu tốn tài nguyên của máy chủ, hoặc ngăn chặn người dùng hợp lệ truy nhập tới các dịch vụ mạng.

Cách phân loại phổ biến thường dựa vào giao thức trong hình thức tấn công DoS, ví dụ như tràn ngập ICMP với Smurf, Ping of Death, khai thác điểm yếu của TCP trong hoạt động của giao thức và phân mảnh gói tin với SYN flood, hay các ứng dụng ở lớp ứng dụng như với Flash Crowds (hay tên gọi khác là X-flash).

Phân loại theo phương thức tấn công, DoS có thể được thực hiện bằng một vài gói tin đơn lẻ gửi thẳng tới server gây rối loạn hoạt động (như slammer worm), hoặc kích hoạt để gửi từ nhiều nguồn (tấn công từ chối dịch vụ phân tán DdoS). Tấn công có thể thực hiện trên mạng Internet (sử dụng ngay các web server), hoặc broadcast trong mạng từ bên trong (insider attacks như với Blaster worm), trên các mạng ngang hàng P2P (P2P index poinsioning) hay Wireless (WLAN authentication rejection attack-spoof sender). Tuy nhiên, có thể thấy các cách phân loại trên dựa chủ yếu vào cách nhìn từ sự phát sinh nguồn tấn công và vì thế, không hệ thống hoá được phương thức phòng tránh.

Zombie (hay còn gọi là daemons, slaves hoặc agent) là đối tượng được lợi dụng trở thành thành phần phát sinh tấn công. Một số trường hợp điển hình như là thông qua rootkit (một dạng phần mềm được kích hoạt mỗi khi hệ thống khởi động, trước cả khi hệ điều hành khởi động xong. Rootkit cho phép cài một file có thuộc tính ẩn, một tiến trình, hoặc một tài khoản người sử dụng lên hệ điều hành. Rootkit có khả năng chặn bắt dữ liệu từ các thiết bị đầu cuối, từ các kết nối mạng và từ bàn phím), hay các thành phần hoạt động đính kém trong email, hoặc trang Web. Để phòng chống, hệ thống mạng cần có những công cụ theo dõi và lọc bó nội dung (content filtering) nhằm ngăn ngừa việc tuyển mộ zombie của các tin tặc.



Có rất nhiều các công cụ tấn công từ chối dịch vụ DoS, chủ yếu là tấn công từ chối dịch vụ phân tán DdoS như là TFN, TFN2000 (Trible Flood Network), tấn công dựa vào nguyên lý hoạt động của các giao thức như là Smurf, UDP, SYN, hay ICMP. Các công cụ này có đặc điểm là cần phải có các kênh phát động để zombie thực hiện tấn công tới một máy đích cụ thể. Hệ thống cần phải có các công cụ để giám sát và ngăn ngừa các kênh phát động đó.

1/. Ngăn chặn tấn công bằng băng thông

Khi một cuộc tấn công DoS được phát động nó thường được phát hiện dựa trên sự thay đổi đáng kể về băng thông của hệ thống mạng. Ví dụ, một hệ thống mạng bình thường có thể có 80% lưu lượng là của giao thực TCP, 20% lưu lượng còn lại là của UDP. Thống kê này nếu có thay đổi rõ rệt có thể là dấu hiệu của một cuộc tấn cống DoS. Ví dụ như, sâu Slammer sẽ làm tăng lưu lượng UDP, trong khi sâu Welchi sẽ tạo ra ICMP flood. Việc phân tán lưu lượng gây ra bởi các sâu này gây tác hại lên bộ định tuyến, tường lửa, hoặc hạ tầng mạng. Hệ thống cần phải có các công cụ giám sát và điều phối băng thông nhằm giảm thiểu tác hại của tấn công dạng này.

2/. Ngăn chặn tấn công qua cơ chế SYN/ACK

SYN flood là một trong những cách tấn công DoS cổ nhất còn tồn tại cho đến thời điểm hiện tại, nhưng tác hại của nó gây ra thì không giảm. Điểm căn bản để phòng ngừa cách tấn công DoS này là khả năng kiểm soát được số lượng yêu cầu SYN/ACK trong cơ chế kết nối bắt tay 3 bước của giao thức TCP tới hệ thống mạng.

3/. Phát hiện và ngăn chặn tấn công tới hạn số kết nối

Bản thân các máy chủ chỉ có thể đáp ứng được một số lượng nhất định các kết nối tới nó cùng một lúc. Ngay bản thân tường lửa (đặc biệt với các tường lửa có tính năng duyệt trạng thái), thì các kết nối luôn được gắn liền với bảng trạng thái có giới hạn dung lượng. Đa phần các cuộc tấn công đều sinh ra số lượng các kết nối ảo thông qua việc giả mạo. Để phòng ngừa tấn công dạng này, hệ thống cần phân tích và chống được việc giả mạo, và kiểm soát được số lượng kết nối từ một nguồn cụ thể tới máy chủ..

4/. Phát hiện và ngăn chặn tấn công tới hạn tốc độ thiết lập kết nối

Một trong những điểm mà các máy chủ thường bị lợi dụng là khả năng các bộ đệm giới hạn dành cho tốc độ thiết lập kết nối, dẫn đến quá tải khi phải chịu sự thay đổi đột ngột về số lượng kết nối. Ở đây, việc áp dụng bộ lọc để giới hạn số lượng kết nối có một vai trò rất quan trọng. Một bộ lọc sẽ xác định ngưỡng tốc độ kết nối cho từng thành phần của mạng.

Trong một hệ thống, để đối phó với các cuộc tấn công từ chối dịch vụ, thì thành phần IPS được coi là quan trọng nhất. Các cuộc tấn công từ chối dịch vụ chủ yếu nhằm vào khả năng xử lý của hệ thống mạng mà đầu tiên là các thiết bị an ninh mạng. Năng lực xử lý của IPS là một trong những đặc điểm cần chú ý, đặc biệt là sự ổn định trong việc xử lý đồng thời các loại lưu lượng hỗn tạp với kích thước gói tin

3.2.2. Cách phòng chống tấn công hệ điều hành bằng kĩ thuật mật mã.

1/. Phòng chống bằng phương pháp mã hóa: sử dụng các tài khoản đăng nhập phức tạp; mã hóa tài khoản người dùng.
Các tệp tin, dữ liệu quan trọng của hệ thống nên bảo vệ bằng kĩ thuật mã hóa, giấu tin, nén tin, vv…

2/. Lỗi chủ yếu được tìm thấy trên các ứng dụng chạy trên hệ điều hành phổ biến hiện nay là Windows, trên các chương trình Webserver, DNS, hay SQL database. Cập nhật các bản vá là một trong những yêu cầu quan trọng cho việc phòng ngừa các điểm yếu của ứng dụng.

3**/.** Dùng phương pháp mật mã để xác thực thực thể kết nối
Ví dụ:

Để phòng chống tấn công ARP (ARP chuyển từ địa chỉ IP sang địa chỉ MAC trong liên lạc mạng) có thể dùng chữ kí số để xác thực thực thể kết nối, chống giả mạo.

3.3. TẤN CÔNG CƠ SỞ DỮ LIỆU

3.3.1. Một số dạng tấn công cơ sở dữ liệu

Ví dụ: tấn công cơ sở dữ liệu bằng phương pháp tiêm vào SQL(SQL Injection)

1/. SQL Injection là gì?

SQL injection là một kĩ thuật cho phép những kẻ tấn công lợi dụng lỗ hổng trong việc kiểm tra dữ liệu nhập trong các ứng dụng web và các thông báo lỗi của hệ quản trị cơ sở dữ liệu để "tiêm vào" (inject) và thi hành các câu lệnh SQL bất hợp pháp (không được người phát triển ứng dụng lường trước). Hậu quả của nó rất tai hại vì nó cho phép những kẻ tấn công có thể thực hiện các thao tác xóa, hiệu chỉnh, … do có toàn quyền trên cơ sở dữ liệu của ứng dụng, thậm chí là server mà ứng dụng đó đang chạy. Lỗi này thường xảy ra trên các ứng dụng web có dữ liệu được quản lí bằng các hệ quản trị cơ sở dữ liệu như SQL Server, MySQL, Oracle, DB2, Sysbase.

2/. Các dạng tấn công bằng SQL Injection

Có bốn dạng thông thường bao gồm: vượt qua kiểm tra lúc đăng nhập (authorization bypass), sử dụng câu lện SELECT, sử dụng câu lệnh INSERT, sử dụng các thủ tục lưu trữ (stored-procedures).

a/. Dạng tấn công vượt qua kiểm tra đăng nhập

- Với dạng tấn công này, tin tặc có thể dễ dàng vượt qua các trang đăng nhập nhờ vào lỗi khi dùng các câu lệnh SQL thao tác trên cơ sở dữ liệu của ứng dụng web.
- Xét một ví dụ điển hình, thông thường để cho phép người dùng truy cập vào các trang web được bảo mật, hệ thống thường xây dựng trang đăng nhập để yêu cầu người dùng nhập thông tin về tên đăng nhập và mật khẩu. Sau khi người dùng nhập thông tin vào, hệ thống sẽ kiểm tra tên đăng nhập và mật khẩu có hợp lệ hay không để quyết định cho phép hay từ chối thực hiện tiếp.
- Trong trường hợp này, người ta có thể dùng hai trang, một trang HTML để hiển thị form nhập liệu và một trang ASP dùng để xử lí thông tin nhập từ phía người dùng.

Ví dụ:

Trang login.htm

```
<form action="ExecLogin.asp" method="post">
        Username: <input type="text" name="fUSRNAME"><br>
        Password: <input type="password" name="fPASSWORD"><br>
        <input type="submit">
</form>
```

Trang execlogin.asp

```
<%
        var vUsrName, vPassword;
        var Conn = Server.CreateObject("ADODB.Connection");
        Conn.ConnectionString = "Driver={Microsoft Access Driver (*.mdb)};DBQ="
        + Server.MapPath("database.mdb");
        Conn.Open();
        var objRS, strSQL;
        vUsrName = "" + Request.Form("fUSRNAME");
        vPassword = "" +Request.Form("fPASSWORD");
        strSQL = "SELECT * FROM T_USERS WHERE USR_NAME='" +
        vUsrName + "' and USR_PASSWORD='" + vPassword + "'";
        objRS = Server.CreateObject("ADODB.Recordset");
        objRS = Conn.Execute(strSQL);
        if (objRS.EOF)
          Response.Write("Invalid login.");
        else
          Response.Write("You are logged in as " + objRS("USR_NAME"));
        Conn.Close();
%>
```

Thoạt nhìn, đoạn mã trong trang **execlogin.asp** dường như không chứa bất cứ một lỗ hổng về an toàn nào. Người dùng không thể đăng nhập mà không có tên đăng nhập và mật khẩu hợp lệ. Tuy nhiên, đoạn mã này thực sự không an toàn và là tiền đề cho một lỗi SQL injection. Đặc biệt, chỗ sơ hở nằm ở chỗ dữ liệu nhập vào từ người dùng được dùng để xây dựng trực tiếp câu lệnh SQL. Chính điều này cho phép những kẻ tấn công có thể điều khiển câu truy vấn sẽ được thực hiện. Ví dụ, nếu người dùng nhập chuỗi sau vào trong cả 2 ô nhập liệu username/password của trang login.htm là: ' OR ' ' = ' '. Lúc này, câu truy vấn sẽ được gọi thực hiện là:

```
SELECT * FROM T_USERS WHERE USR_NAME =" OR "=" and
USR_PASSWORD= " OR "="
```

- Câu truy vấn này là hợp lệ và sẽ trả về tất cả các bản ghi của T_USERS và đoạn mã tiếp theo xử lí người dùng đăng nhập bất hợp pháp này như là người dùng đăng nhập hợp lệ.

b/. Dạng tấn công sử dụng câu lệnh SELECT

Dạng tấn công này phức tạp hơn. Để thực hiện được kiểu tấn công này, kẻ tấn công phải có khả năng hiểu và lợi dụng các sơ hở trong các thông báo lỗi từ hệ thống để dò tìm các điểm yếu khởi đầu cho việc tấn công.

Xét một ví dụ rất thường gặp trong các website về tin tức. Thông thường, sẽ có một trang nhận ID của tin cần hiển thị rồi sau đó truy vấn nội dung của tin có ID này. Ví dụ: http://www.myhost.com/shownews.asp?ID=123. Mã nguồn cho chức năng này thường được viết khá đơn giản theo dạng

```
<%
    var vNewsID, objRS, strSQL;
    vNewsID = Request("ID");
    strSQL = "SELECT * FROM T_NEWS WHERE NEWS_ID =" + vNewsID;
    objRS = Server.CreateObject("ADODB.Recordset");
    objRS = Conn.Execute(strSQL);
    Conn.Close();
%>
```

Thông thường, đoạn mã này hiển thị nội dung của tin có ID trùng với ID đã chỉ định và hầu như không thấy có lỗi. Tuy nhiên, giống như ví dụ đăng nhập ở trước, đoạn mã này để lộ sơ hở cho một lỗi SQL injection khác. Kẻ tấn công có thể thay thế một ID hợp lệ bằng cách gán ID cho một giá trị khác, và từ đó khởi đầu cho một cuộc tấn công bất hợp pháp.

Ví dụ như: 0 OR 1=1 (nghĩa là, http://www.myhost.com/shownews.asp?ID=0 or 1=1).

Câu truy vấn SQL lúc này sẽ trả về tất cả các article từ bảng dữ liệu vì nó sẽ thực hiện câu lệnh:

```
SELECT * FROM T_NEWS WHERE NEWS_ID=0 or 1=1
```

Một trường hợp khác, ví dụ như trang tìm kiếm. Trang này cho phép người dùng nhập vào các thông tin tìm kiếm như Họ, Tên, … Đoạn mã thường gặp là:

```
<%
    var vAuthorName, objRS, strSQL;
    vAuthorName = Request("fAUTHOR_NAME");
    strSQL = "SELECT * FROM T_AUTHORS WHERE AUTHOR_NAME =' " +
    vAuthorName + " ' ";
    objRS = Server.CreateObject("ADODB.Recordset");
    objRS = Conn.Execute(strSQL);
    ...
    Conn.Close();
%>
```

Tương tự như trên, tin tặc có thể lợi dụng sơ hở trong câu truy vấn SQL để nhập vào trường tên tác giả bằng chuỗi giá trị:

```
' UNION SELECT ALL SELECT OtherField FROM OtherTable WHERE ' '=' (*)
```

Lúc này, ngoài câu truy vấn đầu không thành công, chương trình sẽ thực hiện thêm lệnh tiếp theo sau từ khóa UNION nữa.

Tất nhiên các ví dụ nói trên, dường như không có gì nguy hiểm, nhưng hãy thử tưởng tượng kẻ tấn công có thể xóa toàn bộ cơ sở dữ liệu bằng cách chèn vào các đoạn lệnh nguy hiểm như lệnh DROPTABLE.

Ví dụ như: ' DROP TABLE T_AUTHORS --

Chúng ta sẽ thắc mắc là làm sao biết được ứng dụng web bị lỗi dạng này được. Rất đơn giản, hãy nhập vào chuỗi (*) như trên, nếu hệ thống báo lỗi về cú pháp dạng: Invalid object name “OtherTable”; ta có thể biết chắc là hệ thống đã thực hiện câu SELECT sau từ khóa UNION, vì như vậy mới có thể trả về lỗi mà ta đã cố tình tạo ra trong câu lệnh SELECT.

Cũng sẽ có thắc mắc là làm thế nào có thể biết được tên của các bảng dữ liệu mà thực hiện các thao tác phá hoại khi ứng dụng web bị lỗi SQL injection. Cũng rất đơn giản, bởi vì trong SQL Server, có hai đối tượng là sysobjects và syscolumns cho phép liệt kê tất cả các tên bảng và cột có trong hệ thống. Ta chỉ cần chỉnh lại câu lệnh SELECT, ví dụ như:
' UNION SELECT name FROM sysobjects WHERE xtype = 'U' là có thể liệt kê được tên tất cả các bảng dữ liệu.

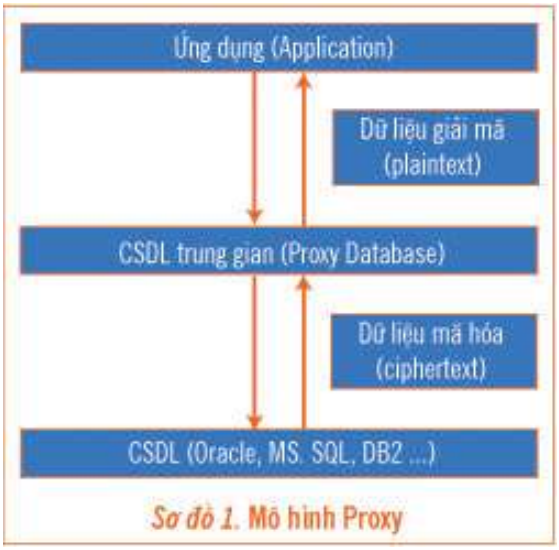
c/. Dạng tấn công sử dụng câu lệnh INSERT

Thông thường các ứng dụng web cho phép người dùng đăng kí một tài khoản để tham gia. Chức năng không thể thiếu là sau khi đăng kí thành công, người dùng có thể xem và hiệu chỉnh thông tin của mình. SQL injection có thể được dùng khi hệ thống không kiểm tra tính hợp lệ của thông tin nhập vào.

d/. Dạng tấn công sử dụng thủ tục lưu trữ (stored-procedures)

Việc tấn công bằng stored-procedures sẽ gây tác hại rất lớn nếu ứng dụng được thực thi với quyền quản trị hệ thống;.

3.3.2. Phòng chống tấn công CSDL bằng kĩ thuật mã hóa



Sơ đồ 1. Mô hình Proxy

1/. Giải pháp đơn giản nhất bảo vệ dữ liệu trong CSDL ở mức độ tập tin, chống lại sự truy cập trái phép vào các tập tin CSDL là hình thức mã hóa. Tuy nhiên, mã hóa dữ liệu ở mức độ này là giải pháp mang tính "được ăn cả, ngã về không", giải pháp này không cung cấp mức độ bảo mật truy cập đến CSDL ở mức độ bảng (table), cột (column) và dòng (row). Một điểm yếu nữa của giải pháp này là bất cứ ai với quyền truy xuất CSDL đều có thể truy cập vào tất cả dữ liệu trong CSDL. Điều này phát sinh một nguy cơ nghiêm trọng, cho phép các đối tượng với quyền quản trị (admin) truy cập tất cả các dữ liệu nhạy cảm. Thêm vào đó, giải pháp này bị hạn chế vì không cho phép phân quyền khác nhau cho người sử dụng CSDL.

2/. Giải pháp thứ hai, đối nghịch với giải pháp mã hóa cấp tập tin nêu trên, giải quyết vấn đề mã hóa ở mức ứng dụng. Giải pháp này xử lý mã hóa dữ liệu trước khi truyền dữ liệu vào CSDL. Những vấn đề về quản lý khóa và quyền truy cập được hỗ trợ bởi ứng dụng. Truy vấn dữ liệu đến CSDL sẽ trả kết quả dữ liệu ở dạng mã hóa và dữ liệu này sẽ được giải mã bởi ứng dụng. Giải pháp này giải quyết được vấn đề phân tách quyền an ninh và hỗ trợ các chính sách an ninh dựa trên vai trò (Role Based Access Control – RBAC).

Tuy nhiên, xử lý mã hóa trên tầng ứng dụng đòi hỏi sự thay đổi toàn diện kiến trúc của ứng dụng, thậm chí đòi hỏi ứng dụng phải được viết lại. Đây là một vấn đề đáng kể cho các công ty có nhiều ứng dụng chạy trên nhiều nền CSDL khác nhau.

Từ những phân tích hai giải pháp nêu trên, có thể dễ dàng nhận thấy một giải pháp bảo mật CSDL tối ưu cần hỗ trợ các yếu tố chính sau:

- Hỗ trợ mã hóa tại các mức dữ liệu cấp bảng, cột, hàng.
- Hỗ trợ chính sách an ninh phân quyền truy cập đến mức dữ liệu cột, hỗ trợ RBAC.
- Cơ chế mã hóa không ảnh hưởng đến các ứng dụng hiện tại.

Hai mô hình thỏa mãn các yêu cầu trên, đặc biệt là yêu cầu thứ ba.

1/. Xây dựng tầng CSDL trung gian

Trong mô hình này, một CSDL trung gian (proxy) được xây dựng giữa ứng dụng và CSDL gốc (Sơ đồ 1). CSDL trung gian này có vai trò mã hóa dữ liệu trước khi cập nhật vào CSDL gốc, đồng thời giải mã dữ liệu trước khi cung cấp cho ứng dụng. CSDL trung gian đồng thời cung cấp thêm các chức năng quản lý khóa, xác thực người dùng và cấp phép truy cập.

Giải pháp này cho phép tạo thêm nhiều chức năng về bảo mật cho CSDL. Tuy nhiên, mô hình CSDL trung gian đòi hỏi xây dựng một ứng dụng CSDL tái tạo tất cả các chức năng của CSDL gốc.

Hiện tại, trên thị trường sản phẩm mã hóa CSDL, Secure.Data của công ty Protegrity (www. Protegrity.com) sử dụng mô hình proxy nêu trên.

2/. Sử dụng cơ chế sẵn có trong CSDL

Mô hình này giải quyết các vấn đề mã hóa cột dựa trên các cơ chế sau:

- Các hàm Stored Procedure trong CSDL cho chức năng mã hóa và giải mã.
- Sử dụng cơ chế View trong CSDL tạo các bảng ảo, thay thế các bảng thật đã được mã hóa.

Trong mô hình này, dữ liệu trong các bảng gốc sẽ được mã hóa, tên của bảng gốc được thay đổi. Một bảng ảo (View) được tạo ra mang tên của bảng gốc, ứng dụng sẽ truy cập đến bảng ảo này.

3.4. TẤN CÔNG MÁY TÍNH

3.4.1. Một số dạng tấn công máy tính

1/. Lỗ hổng bảo mật vật lý

Một lỗ hổng vật lý của máy tính sẽ hoàn toàn bị khai thác ngay cả khi phương pháp nhận dạng phức tạp nhất, phương pháp mã hóa bảo mật nhất. Một chương trình theo dõi các thao tác trên bàn phím (keylogger), cả phần mềm lẫn phần cứng được cài đặt, khóa PGP bí mật của chúng ta sẽ bị lộ, do đó mọi dữ liệu mã hóa và tài khoản bị tổn hại. Bất chấp mật khẩu của chúng ta dài và bảo mật đến đâu thì lỗ hổng bảo mật vật lí là một trong những trường hợp nguy hiểm nhất.

2/. Sự chia sẻ không chủ đích

Một mật khẩu thường được chia sẻ cho bạn bè, ông chủ, gia đình trong nhiều hoàn cảnh khác nhau. Một điểm lợi đó là sự thuận tiện cho hai hay nhiều người để biết một dữ liệu tài khoản nhất định để tiếp cận được một tài nguyên nào đó. Các mật khẩu có thể được chia sẻ trong các cuộc nói chuyện với các đồng nghiệp trong việc thảo luận chính sách mật khẩu mới nhất của công ty, hay cách họ chọn mật khẩu, cách họ duy trì chúng. Một trong các phương pháp nguy hiểm và dễ dàng để đạt được các dữ liệu nhạy cảm bằng cách hỏi họ, bằng phương pháp trực tiếp hay gián tiếp, cái này gọi là social engineering. (Social engineering có thể tóm gọn như sau: để có được tài khoản của người khác ngoài cách sử dụng máy móc hoặc phần mềm, thì có thể sử dụng con người. ví dụ trên diễn đàn có thể đăng lý do đang sửa chữa, nâng cấp mong chúng ta cung cấp lại tài khoản, hoặc chúng ta nhận được tin nhắn trúng tiền thưởng mong hãy cung cấp tài khoản, v.v…)

3/. Bẻ khóa

Trong các trường hợp tấn công ngan hàng, các tập tin mật khẩu được mã hóa của công ty có thể được phơi bày trước những kẻ tấn công. Nếu xảy ra, những kẻ tấn công bắt đầu bẻ khóa tập tin này, sử dụng tất cả các khả năng kết hợp với ý tưởng tìm ra những mật khẩu yếu nhất để có thể có những đặc quyền sau này. Trong trường hợp công ty lo ngại mật khẩu bị tổn hại, ngay lập tức công ty thông báo cho mọi nhân viên thay đổi mật khẩu của họ, do đó dù cho các mật khẩu yếu nhất bị lộ, thì nó không còn hiệu lực nữa. Tuy nhiên, nếu công ty không lo mật khẩu bị lộ, họ thường xuyên tự bẻ khóa như cách những kẻ tấn công đã làm để lọc ra các mật khẩu yếu.

3.4.2. Phòng chống

1/. Sử dụng mật khẩu phức tạp, mã hóa dữ liệu, các file thông tin người dùng quan trọng trong máy tính.

2/. Sử dụng nhiều phần mềm mã hóa cùng lúc trên máy tính để bảo vệ cho từng loại dữ liệu chuyên biệt

3**/.** Tuyệt đối không sử dụng máy tính đang chứa những thông tin bí mật để lướt web một cách quá thoải mái.

4**/.** Thiết lập các chính sách bảo mật tổng thể, quản trị dễ dàng, đáp ứng các yêu cầu của ISO/IEC27001, ISMS, ITIL …

5/. Xây dựng hệ thống tường lửa vững chắc.

3.5. TẤN CÔNG PHẦN MỀM

3.5.1. Tấn công phần mềm.

Lợi dụng sơ hở của những phần mềm ứng dụng để tấn công làm cho chương trình thực thi bị sai hoặc lỗi không thể thi hành.

1/. Các lỗ hổng của Adobe

Ngoài Microsoft, Adobe cũng là một hãng sản xuất phần mềm hoạt động trên các máy tính cài Windows. Mọi người có Flash, Acrobat Reader và Shockwave và chúng được phần mềm mã độc sử dụng như một cơ chế để phát tán các thứ độc hại cho người dùng (tương tự chương trình Adobe hoạt động trên HĐH khác nhưng đích nhắm tới các máy PC Windows lại chiếm ưu thế hơn).

Nguy hiểm xảy ra đối với người dùng khi họ sử dụng các phiên bản của chương trình không được cập nhật hay phiên bản hiện thời có chứa các lỗ hổng chưa được vá và sẽ bị lợi dụng như các lỗ hổng an ninh.

Cơ chế hoạt động của chúng là lừa cho người dùng kích vào một trang web quảng cáo Flash hoặc tài liệu PDF bị nhiễm mã độc tự động mở ra khi ghé thăm vào trang quảng cáo.

2/. Điểm yếu của Firefox

Mối đe dọa: Phần mở rộng (add-on) của Firefox là một mối đe dọa bảo mật tiềm ẩn, tuy không đáng sợ như plug-in ActiveX của IE nhưng vẫn có nguy cơ cao. Nhiều cuộc tấn công web nhắm vào Firefox, phá hủy các add-on và cấu trúc hỗ trợ cho chương trình.

Cơ chế: Hầu hết mối nguy hiểm đến từ add-on đều giả vờ là hợp pháp. Chẳng hạn như chúng giả vờ là chương trình Adobe Flash Player và yêu cầu người dùng cập nhật. Điều đó đồng nghĩa với mã độc sẽ thâm nhập vào máy tính của nạn nhân. Hoặc thông qua các tập tin hỗ trợ, chỉnh sửa các chương trình và viết lại truy cập tới các tập tin khác theo mục đích của tin tặc.

3/. Sự đầu độc DNS

Mối đe dọa: Các máy chủ DNS có nhiệm vụ dịch các địa chỉ Internet thành tên miền thân thiện với người dùng. Tuy nhiên, thông tin cung cấp bởi các máy chủ DNS có thể bị tấn công hoặc bị định hướng sai. Điều này cho phép kẻ tấn công gửi cho người dùng bất cứ website nào mà chúng chọn.

Cơ chế: Các cuộc tấn công DNS phổ biến nhất khai thác lỗ hổng trong phần mềm máy chủ DNS để cho phép làm giả dữ liệu phân giải tên miền gửi tới khách hàng. Điển hình là vụ đầu độc DNS vào năm 2008 khi nhà nghiên cứu máy tính Dan Kaminsky chứng minh, làm thế nào các tên miền có thể chuyển hướng đối với phiên bản BIND hiện nay. BIND là phần mềm được sử dụng trên hầu hết máy chủ thực hiện phân giải DNS. Kết quả cuối cùng là tin tặc có thể đánh cắp toàn bộ tên miền- bao gồm cả tên miền con của chúng, các máy chủ mail, các bán ghi SPF và mọi thứ khác có thể có trong tài nguyên DNS.

3.5.2. Phòng chống tấn công phần mềm

1/. Luôn cập nhật các bản vá lỗi mới nhất từ nhà sản xuất

2/. Mã hóa, cất giấu các tập tin dữ liệu quan trọng.

3/. Mã hóa phần mềm, xác thực phần mềm.

4/. Khai thác tối ưu các tính năng bảo mật của mọi thành phần hệ thống.

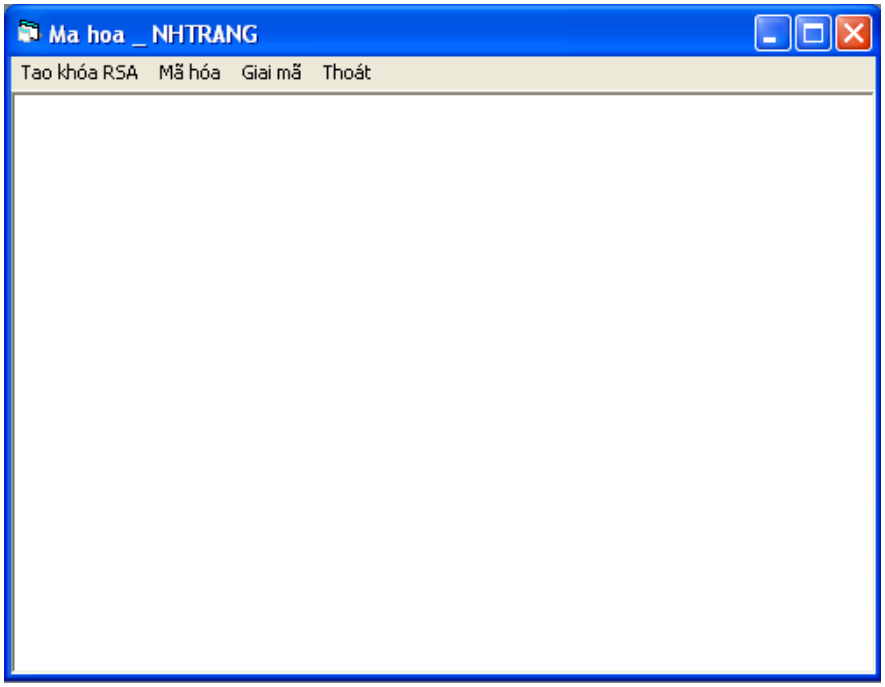
5/. Cung cấp khả năng tự hồi phục đối với các tấn công

6/. Xây dựng hệ thống tường lửa an toàn.

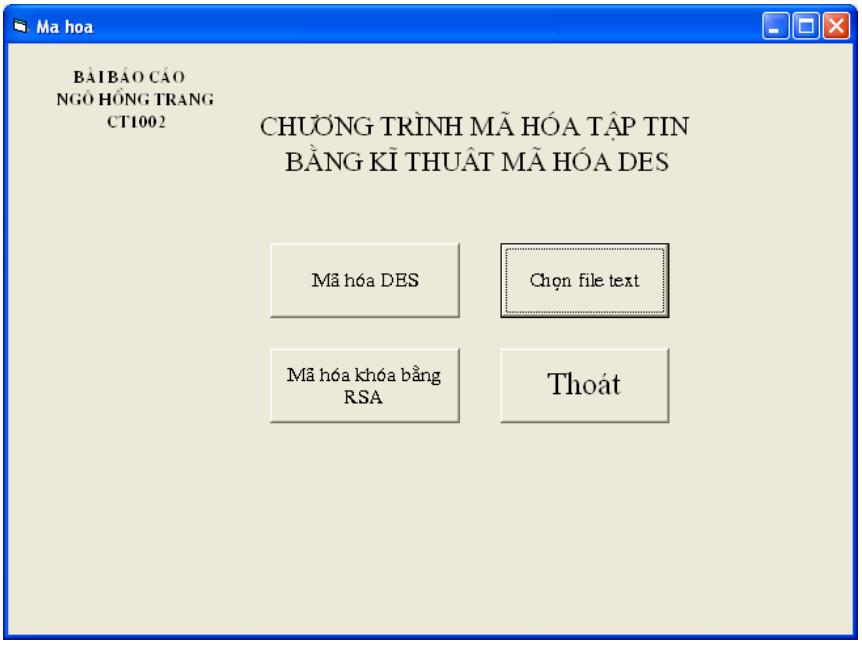
Chương 4. CHƯƠNG TRÌNH THỬ NGHIỆM

4.1. Giao diện chương trình

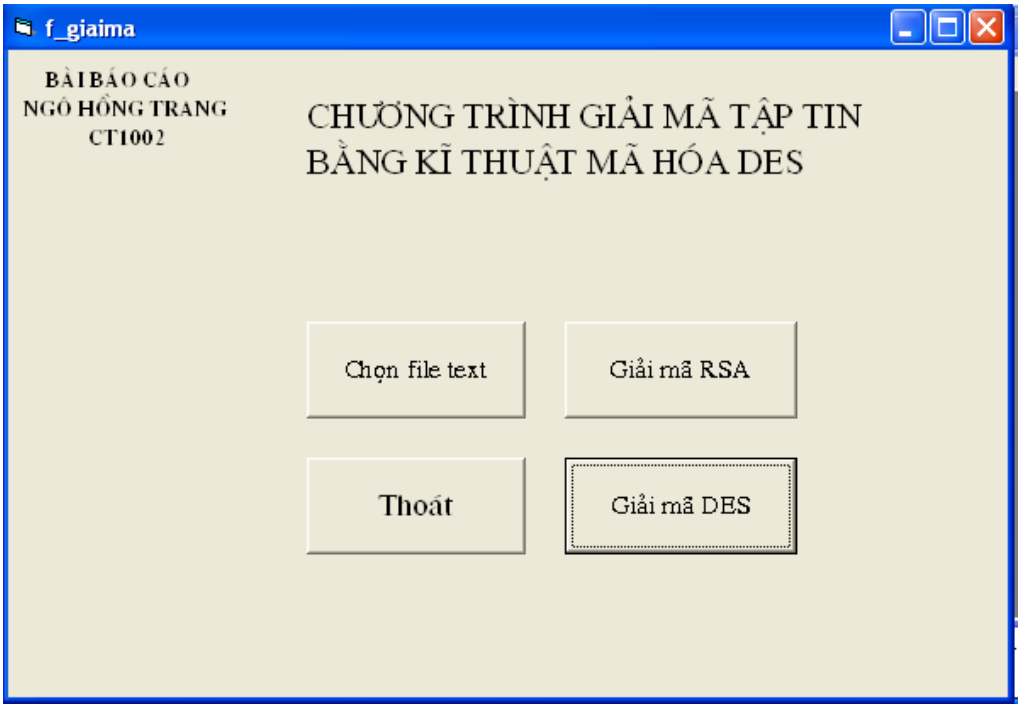
1/. Giao diện chính



2/. Giao diện chương trình mã hóa tập tin



3/. Giao diện chương trình giải mã tập tin text



4.2. Hướng dẫn chạy chương trình

1/. Cài đặt chương trình: Chạy file setup.exe và thực hiện các bước cài đặt

Mở chương trình: Start → programe →Baomat_NHTrang → Baomat_NHTrang

2/. Các nút chức năng

- ***Trong giao diện chương trình chính***

Tạo khóa cho RSA: nút này tạo ra khóa cho RSA để tham gia quá trình mã hóa tập tin khóa. Khi nhấn nút này thì 1 khóa bí mật và một khóa công khai của RSA được tạo ra.

Mã hóa: mở ra giao diện chương trình mã hóa tập tin

Giải mã: mở ra giao diện chương trình giải mã tập tin.

Thoát: đóng toàn bộ chương trình, kết thúc làm việc.

- ***Trong giao diện chương trình mã hóa tập tin***

Chọn file text: chỉ đường dẫn đến tập text cần mã hóa.

Mã hóa DES: mã hóa tập tin text bằng hệ mã hóa DES.

Mã hóa RSA: mã hóa tập tin khóa bí mật của hệ mã DES

Thoát: đóng cửa sổ chương trình mã hóa tập tin

- ***Trong giao diện chương trình giải mã tập tin***

Chọn file text: chỉ đường dẫn đến tập text cần giải mã.

Giải mã RSA: giải mã tập tin khóa bí mật của hệ mã DES

Giải mã DES: giải mã tập tin text đã được mã hóa bằng hệ mã hóa DES.

Thoát: đóng cửa sổ chương trình giải mã tin

- ***Trong giao diện chương trình Nén dữ liệu***

Chọn file text: chỉ đường dẫn đến tập text cần nén hoặc giải nén.

Nén dữ liệu: nén tập tin text, trong khi nén có sử dụng mã hóa DES byte

Giải nén: thực hiện chức năng giải nén tập tin đã nén.

4.3. Môi trường chạy ứng dụng

Hệ điều hành Window XP, Windows 7,…

Phần mềm viết ứng dụng: Microsoft Visual Basic 6.0

Sử dụng thư viện API Personal Edition và PKI Toolkit Trial Edition.

KẾT LUẬN

Trong suốt quá trình làm đồ án tốt nghiệp được sự giúp đỡ nhiệt tình của các thầy, cô khoa Công nghệ thông tin cùng các bạn trong lớp, đặc biệt là thầy Trịnh Nhật Tiến đã giúp đỡ em hoàn thành đề tài "Tìm hiểu một số dạng tấn công và phòng chống bằng kĩ thuật mật mã" đúng thời hạn. Do thời gian trình độ còn hạn chế và thời gian có hạn do đó bài luận văn của em không tránh khỏi sai sót. Em rất mong nhận được sự đóng góp ý kiến của thầy cô và các bạn để bài luận văn được hoàn chỉnh hơn.

NHỮNG KẾT QUẢ ĐẠT ĐƯỢC

1/. Tìm hiểu và nghiên cứu lý thuyết:

Một số dạng tấn công hệ thống thông tin (thông qua mạng máy tính, hệ điều hành, cơ sở dữ liệu….)

- Một số kĩ thuật mật mã.
- Nghiên cứu phương pháp phòng chống tấn công bằng kĩ thuật mật mã

2/. Thử nghiệm Chương trình DEMO phòng chống tấn công.

TÀI LIỆU THAM KHẢO

1/**.** Giáo trình An toàn và bảo mật thông tin của PGS.TS Trinh Nhật Tiến.

2/. Tham khảo tài liệu tại các trang web như:

http://vi.wikipedia.org/

http://www.caulacbovb.com/

http://www.cryptosys.net/