TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN CÔNG NGHỆ TRI THỨC

TRẦN HỒNG NGỌC – TRƯƠNG THỊ MỸ TRANG

NGHIÊN CỨU CÁC PHƯƠNG PHÁP MÃ HOÁ – GIẤU TIN ĐA TẦNG VÀ ỨNG DỤNG

LUẬN VĂN CỬ NHÂN TIN HỌC

TP. HCM, 2004

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN CÔNG NGHỆ TRI THỨC

TRƯƠNG THỊ MỸ TRANG - 0012694
TRẦN HỒNG NGỌC - 0012746

NGHIÊN CỨU CÁC PHƯƠNG PHÁP MÃ HOÁ – GIẤU TIN ĐA TẦNG VÀ ỨNG DỤNG

LUẬN VĂN CỬ NHÂN TIN HỌC

GIÁO VIÊN HƯỚNG DẪN
T.S NGUYỄN ĐÌNH THÚC
Th.S PHẠM PHẠM TUYẾT TRINH

NIÊN KHÓA 2000 - 2004

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

LỜI CẢM ƠN

Chúng em xin chân thành cám ơn Khoa Công Nghệ Thông Tin, trường Đại Học Khoa Học Tự Nhiên TpHCM đã tạo điều kiện tốt cho chúng em thực hiện đề tài luận văn tốt nghiệp này.

Chúng em xin chân thành cám ơn Thầy Nguyễn Đình Thúc và Cô Phạm Phạm Tuyết Trinh đã tận tình hướng dẫn, chỉ bảo và đóng góp ý kiến cho chúng em trong suốt thời gian thực hiện đề tài.

Chúng em xin chân thành cám ơn quý Thầy Cô trong Khoa đã tận tình giảng dạy, trang bị cho chúng em những kiến thức quý báu trong những năm học vừa qua.

Chúng con xin nói lên lòng biết ơn sâu sắc đối với Ông Bà, Cha Mẹ đã chăm sóc, nuôi dạy chúng con thành người.

Xin chân thành cám ơn các anh chị và bạn bè đã ủng hộ, giúp đỡ và động viên chúng em trong thời gian học tập và nghiên cứu.

Mặc dù chúng em đã cố gắng hoàn thành luận văn trong phạm vi và khả năng cho phép nhưng chắc chắn sẽ không tránh khỏi những thiếu sót. Chúng em kính mong nhận được sự cảm thông và tận tình chỉ bảo của quý Thầy Cô và các bạn

.

Sinh viên

Trần Hồng Ngọc – Trương Thị Mỹ Trang

Tháng 07/ 2004

MỤC LỤC

DANH SÁCH CÁC HÌNH VẼ

Chương 1. Giới thiệu

Trong những năm gần đây, sự phát triển nhanh chóng của Internet và các công cụ xử lý multimedia đã mang lại cho chúng ta nhiều thuận lợi trong việc lưu trữ dữ liệu, trao đổi thông tin, sao chép dữ liệu v.v…Tuy nhiên, bên cạnh các thuận lợi đó, sự phát triển này cũng tạo ra nhiều thử thách trong vấn đề tìm ra giải pháp bảo mật dữ liệu cũng như việc chứng nhận quyền sở hữu của các cá nhân. Những thử thách này đã thu hút sự chú ý của nhiều nhà nghiên cứu trong lĩnh vực công nghệ thông tin và toán: đó chính là bảo mật:

- Ø Làm sao để bảo mật dữ liệu?
- Ø Làm sao chứng nhận một dữ liệu nào đó thuộc quyền sở hữu của người này hay người kia?
- Ø Làm sao để người nhận biết được thông tin mà họ nhận được là chính xác?
- Ø Làm sao để tin tức truyền đi không bị đánh cắp?

Hiện đã có nhiều giải pháp được đề xuất như: sử dụng mật khẩu, mã hoá thông tin, steganography, ẩn dữ liệu (watermarking) v.v….và bên cạnh các phương pháp bảo mật mới ngày càng phức tạp thì cũng xuất hiện nhiều dạng tấn công khác nhau và ngày càng tinh vi hơn. Do đó, vấn đề làm sao đưa ra một giải pháp hiệu quả theo thời gian và sự phát triển mạnh mẽ của khoa học kỹ thuật và các kỹ thuật phần cứng là không dễ.

Trong giới hạn của luận văn, chúng tôi sẽ trình bày sơ nét về các giải pháp này để chúng ta có cái nhìn tổng quát về bảo mật thông tin. Đồng thời, chúng tôi cũng đề xuất một số ứng dụng. Bố cục luận văn gồm 6 chương và ba phụ lục:

Chương 1. Giới thiệu - Trình bày khái quát về luận văn và giới hạn mục tiêu của đề tài.

Chương 2. Một số hệ mã hóa - Trình bày một số khái niệm cơ bản, hệ mã khoá công khai RSA, ECC, hệ mã đối xứng RC6 và so sánh giữa RSA và ECC.

Chương 3. Hàm băm - Giới thiệu một số phương pháp băm hỗ trợ tăng tốc độ xử lý cho việc mã hoá dữ liệu trong ứng dụng tạo chữ ký điện tử.

Chương 4. Giấu dữ liệu – WaterMarking - Giới thiệu sơ lược về kỹ thuật giấu dữ liệu và một số phương pháp giấu dữ liệu dựa trên phép biến đổi Wavelet.

Chương 5. Một số ứng dụng – Mô tả một số ứng dụng được đề xuất dựa trên các kiến thức đã trình bày ở các chương trên.

Chương 6. Kết luận và hướng phát triển - Đánh giá các kết quả đạt được và đưa ra hướng phát triển cho luận văn.

Phụ lục 1: Biến đổi Wavelet - Trình bày sơ nét về phép biến đổi Wavelet là cơ sở của các phương pháp giấu dữ liệu được trình bày trong Chương 4.

Phụ lục 2: Kết quả thử nghiệm hàm băm Tiger và Whirlpool.

Chương 2. Một số hệ thống mã hoá

2.1. Các khái niệm cơ bản

2.1.1. Số nguyên tố

Định nghĩa 2.1:

Gọi Z là tập các số nguyên {0, 1, -1, 2, -2, ...} và N = {n ∈ Z, n ≥ 0}; N* = { n ∈ Z, n ≥ 1}. Cho a, b ∈ Z, nếu ∃c ∈ Z : b = ac ta nói a là ước số của b hay b là bội số của a, ký hiệu a|b.

Định lý 2.1:

Với a, b, c, x, y ∈ Z:

a|a, 1|a;

a|b, b|c ⇒ a|c

a|b, a|c ⇒ a|(bx + cy)

a|b, b|a ⇒ a = ± b

a|b ⇒ (-a)|b, a|(-b), (-a)|(-b)

Định nghĩa 2.2:

Cho a ∈ Z, b ∈ N*, tồn tại duy nhất q ∈ Z, và r ∈ N sao cho: a = bq + r, 0 ≤ r ≤ b. Khi đó: q được ký hiệu là a/b và r được ký hiệu là a % b (hay a mod b).

Định nghĩa 2.3:

- Một số nguyên c ∈ Z được gọi là ước chung của a,b ∈ Z nếu c|a và c|b.
- Một ước chung d ∈ Z của a, b ∈ Z, được gọi là ước chung lớn nhất của a,b ∈ Z , ký hiệu d = gcd(a,b) hay d = a ∧ b, nếu mọi bội số của mọi số chia của a, b; (c ∈ Z, c|a, c|b ⇒ c|d).

Định nghĩa 2.4:

- Một số nguyên d ∈ Z được gọi là bội chung của a, b ∈ Z nếu a|d và b|d.
- Một bội chung d ∈ N của a, b ∈ Z, được gọi là bội chung nhỏ nhất của a, b ∈ Z, ký hiệu d = lcm(a,b) hay d = a ∨ b, nếu mọi bội số của a, b đều chia hết cho d; (c ∈ Z, a|c, b|c ⇒ d|c).

Định nghĩa 2.5:

Hai số nguyên tố a,b ∈ Z được gọi là nguyên tố cùng nhau, ký hiệu a ⊥ b, nếu a ∧ b=1.

4 Ghi chú:

$1 \wedge 1 = 1 \Rightarrow 1 \perp 1$.

a, b $\in$ N và a ≥ 2, b ≥ 2, a $\perp$ b, thì a $\neq$ b. Thực vậy, nếu a = b, ta có a $\wedge$ a $\neq$ a $\neq 1$.

Định nghĩa 2.6:

Một số nguyên a ≥ 2 được gọi là số nguyên tố nếu nó chỉ có ước số dương là 1 và a; ngược lại, a được gọi là hợp số. Vì thế, 1 là hợp số. Tập tất cả các số nguyên tố ký hiệu là P.

4 Ghi chú:

§ a,b $\in$ P, a $\neq$ b $\Rightarrow$ a $\perp$ b; vì a và b chỉ có ước chung dương là 1.

§ Nếu a $\in$ P, $\forall$ b $\in$ Z và không là bội số của a thì b nguyên tố cùng nhau với a. Thực vậy, nếu c là ước chung dương của a, b và c $\neq 1$, ta có c = a vì a là số nguyên tố. Thì b chia hết cho c là vô lý.

§ Mỗi số nguyên lớn hơn hay bằng 2 có ít nhất một ước số là số nguyên tố: ước chung nguyên tố nhỏ nhất cũng sẽ lớn hơn hay bằng 2.

Định lý 2.2:

$\forall$ a, b $\in$ N*, a > b, ta có: a $\wedge$ b = b $\wedge$ (a % b)

Thuật toán 2.1: (Thuật toán Euclid tính ước chung lớn nhất của 2 số nguyên dương)

Cho a,b $\in$ N, a > b > 1. Đặt a = r_0, b = r_1.

Từ định nghĩa 1.2, ta có:

$r_0 = r_1q_1 + r_2$, $0 \leq r_2 < r_1$;

$r_1 = r_2q_1 + r_3$, $0 \leq r_3 < r_2$;

……

$r_{k-2} = r_{k-1}q_{k-1} + r_k$, $0 \leq r_k < r_{k-1}$; $2 \leq k \leq n$;

$r_{n-1} = r_nq_n + r_{n+1}$, $0 = r_{n+1} < r_n$;

$\Rightarrow a = r_0 > b = r_1 > r_2 > \ldots > r_n > r_{n+1} = 0$.

Vậy,

r_n = gcd(a, b);

Theo định lý 2.2:

gcd(a,b) = gcd(r_0,r_1) = gcd(r_1,r_2) = … = gcd(r_{n-1},r_n)

= gcd($r_nq_n + r_{n+1}$, r_n) gcd(r_nq_n, r_n) = r_n.

Định lý 2.3: (Định lý Eulid mở rộng)

Với a,b $\in$ N, a > b $\geq$ 1; ta có:

$\exists$ x, y $\in$ Z: ax + by = 1;

Nếu a $\perp$ b, $\exists$ x, y $\in$ Z: ax + by = 1;

a $\perp$ b $\Leftrightarrow \exists$ x, y $\in$ Z: ax + by = 1.

Định nghĩa 2.8: (Định nghĩa phép đồng dư)

Cho x,y ∈ Z, m ∈ N*:

x được gọi là đồng dư của y mod m, ký hiệu:

$$x \equiv y(\text{mod } m).$$

nếu m là số chia của x – y; m được gọi là môđun của phép đồng dư.

Tập các số nguyên modulo m, ký hiệu Z_m, là tập:

$$Z_m = \{0, 1, 2, \dots, m-1\}$$

Nếu x ∈ Z_m, số nguyên x mod m của Z_m được gọi là rút gọn modulo của x theo m.

Định nghĩa 2.9:

Một số nguyên a ∈ Z_m được gọi là modulo m khả nghịch nếu tồn tại x ∈ Z_m: ax = 1 (mod m). Nếu tồn tại số x này, thì x là duy nhất, và được gọi nghịch đảo của a modulo m, ký hiệu a^{-1} (mod m), hay đơn giản hơn a^{-1}.

Định nghĩa 2.10: (Định nghĩa hàm phi-Euler)

Cho n ≥ 1, đặt φ(n) là tập các số nguyên trong khoảng [1,n] nguyên tố cùng nhau với n. Hàm φ như thế được gọi là hàm phi-Euler

Định lý 2.3:

Cho k = card(Z_m*) = φ(m), m ≥ 2, Z_m* = {x_1, x_2, …, x_k}, xZ_m* = {xy; y ∈ Z_m*}, ∀x ∈ Z_m*, và u = $x_1x_2\dots x_k$. Ta có:

xZ_m* = Z_m*, ∀x ∈ Z_m*.

$u = x^k u \text{ (mod } m)$, $x^k = 1 \text{ (mod } m)$.

(Định lý Euler) $x \perp m \Rightarrow x^{\varphi(m)} = 1 \text{ (mod } m)$.

(Định lý Fermat) $p \in P, x \perp p \Rightarrow x^{p-1} = 1 \text{ (mod } p)$.

$p \in P, n \in Z \Rightarrow n^p = n \text{ (mod } p)$.

$p \in P, n \in Z, r = s \text{ (mod } \varphi(m)) \Rightarrow nr = ns \text{ (mod } p)$.

Nếu p,q là hai số nguyên tố khác nhau và n = pq, thì φ(m) = (p-1)(q-1).

Định lý 2.4: (Định lý số dư Trung Hoa)

Nếu các số nguyên $n_1,\dots,n_k$ đôi một nguyên tố cùng nhau và $n = n_1\dots n_k$ thì $x \equiv a_1(\text{mod } n_1), \dots, x \equiv a_k \text{ (mod } n_k)$, có duy nhất nghiệm trong Z_n.

Ánh xạ $f: Z_n \to Z_{n-1} \times \dots \times Z_{nk}$, xác định bởi

$$f(x) = (x \bmod n_1, \dots, x \bmod n_k), x \in Z_n$$

Một số phép tính nhanh trên các số nguyên:

- *Phép toán mod trên trường Z_m*

  Ta có : x = y (mod m)

Khi x, y, m là những số nguyên lớn, 0 ≤ x,y ≤ m, b ≥ 2 và $y = \sum_{0 \le i \le I} y_i b^i$ là biểu diễn cơ sở b của y, ta có:

$$(x * y) \bmod m = (y_0 * x + \sum_{0 \le i \le I} y_i (x * b^i) \bmod m)) \bmod m$$

$$(x * b^i) \bmod m = (b * ((x * b^i) \bmod m)) \bmod m, 0 \le i \le I$$

Vì thế ta có thể sử dụng thuật giải sau để tính tích modulo P = (x*y)mod m

Thuật giải: (trường hợp tổng quát)
Đặt x = x mod m, y = y mod m, và P = (y_0 * x) mod m
Cho i từ 1 đến I, do:

a. x = (b * x) mod m;
b. Tính x = (y_i * x) mod m, và P = (P + x) mod m

Return P

q *Phép lũy thừa mod*

Định nghĩa 1.11: Cho x ∈ Z_m và một số nguyên p ∈ N* có biểu diễn nhị phân p = $\sum_{0\le i\le I} p_i 2^i$. Việc tính giá trị y = x^p mod m được gọi là phép lũy thừa mod. Ta có:

$$x^p = x^{p_0} * (x^2)^{p_1} * (x^4)^{p_2} * ... * (x^{2^i})^{p^i}$$

Thuật giải:
Đầu vào: x ∈ Z_m, p = $\sum_{0\le i\le I} p_i 2^i$
Đầu ra: y = xp mod m

y = 1. Nếu p = 0, return y
A = x. Nếu p_0 = 1 thì y = x
Cho I chạy từ 1 đến I, do:

c. $A = A^2$ mod m
d. Nếu pi = 1 thì y = (A*y) mod m

Return y

2.1.2. Mã hóa khóa bí mật (Private-Key Encryption):

Các thuật toán mã hoá khoá bí mật dùng một khoá bí mật đơn để mã hóa và giải mã dữ liệu. Cần phải giữ an toàn cho khoá từ việc truy cập bởi các tác nhân không được phân quyền bởi vì bất kỳ người nào có khoá đều có thể giải mã dữ liệu. Mã hoá khoá bí mật cũng được gọi là mã hoá khoá đối xứng bởi vì chỉ cùng một khoá dùng cho cả mã hoá và giải mã. Thuật toán mã hoá khoá bí mật tốc độ cực kỳ nhanh (so với các thuật toán khoá công) và thích hợp tốt với việc thực thi biến đổi mật mã trên các dòng dữ liệu lớn.

Điển hình, các thuật toán khoá bí mật được gọi là mật mã khối (block cipher) được dùng để mã hoá một khối dữ liệu tại một thời điểm. Các phương pháp mã khối như RC2, DES, Tripple DES, và Rijindael,… biến đổi một khối n byte đầu vào thành một khối byte được mã hoá ở đầu ra. Nếu muốn mã hoá hay giải mã một chuỗi byte cần phải biến đổi nó thành từng khối, bởi vì kích thước n thì nhỏ (n = 8 byte cho RC2, DES và Tripple DES n = 16; n = 24; hay

n = 32 byte đối với Rijindael), các giá trị lớn hơn n phải được mã hoá một khối tại một thời điểm.

Các loại mã hóa khối dùng kiểu xích được gọi là xích khối mật mã (CBC_Cipher Block Chaining) dùng một khoá và một vectơ khởi tạo (IV_Initial Vector) để thực thi biến đổi mật mã trên dữ liệu. Đối với khoá bí mật K được cho, một mật mã khối đơn giản không dùng IV sẽ mã hoá cùng hai khối đầu vào của văn bản gốc giống nhau (plaintext) sẽ cho cùng một khối đầu ra văn bản mật mã (ciphertext). Nếu các khối được nhân đôi trong chuỗi dữ liệu văn bản gốc, thì các khối được mã hoá được nhân đôi trong chuỗi văn bản mật mã. Nếu một người dùng không được phân quyền biết bất cứ điều gì về cấu trúc của khối văn bản gốc, họ có thể dùng thông tin đó để giải mã khối văn bản mật mã được biết và có thể phục hồi khoá. Để chống lại điều này, thông tin từ khối trước đó được trộn vào trong tiến trình mã hoá khối tiếp theo. Vì vậy, đầu ra của hai khối văn bản gốc giống nhau là khác nhau bởi vì kỹ thuật này dùng khối trước đó để mã hoá khối tiếp theo, một IV được dùng để mã hóa khối đầu tiên của dữ liệu. Dùng hệ thống này, các phần đầu (header) thông điệp phổ biến có thể bị những người không có quyền biết đến thì họ cũng không thể dùng công nghệ đảo (reverse engineer) để tìm ra khoá.

Chỉ có một cách để xâm phạm vào dữ liệu được mã hoá bằng phương pháp mật mã này là thực hiện vét cạn với mọi khoá có thể có. Phụ thuộc vào kích thước khoá được dùng để thực hiện mã hoá, loại tìm kiếm này sử dụng rất nhiều thời gian thậm chí dùng các máy tính nhanh nhất và do đó không khả thi. Các kích thước khoá lớn hơn, thì khó giải mã hơn. Mặc dù việc mã hoá về mặt lý thuyết không có nghĩa là không khả thi việc lấy dữ liệu đã mã hóa, nó không đưa ra các chi phí thời gian phải trả cho việc thực hiện giải mã này. Nếu mất nhiều tháng để thực hiện mọi cách tìm kiếm dữ liệu thì nó cũng chỉ có ý nghĩa như vài ngày vì đều không có kết quả. Do đó, phương pháp vét cạn là không thể thực hiện.

Sự bất lợi của khóa bí mật là nó cho hai nhóm người thoả thuận về khoá và IV và truyền thông các giá trị của họ. Cũng vậy, khoá phải được giữ bí mật đối với những người không quyền. Vì những vấn đề này mà mã hoá khoá bí mật thường được dùng chung với mã hóa khóa công để truyền thông cá nhân các giá trị của khóa và IV. Một số thuật toán mã hóa phổ biến như DES (Data Encrypion Standard), AES (Advanced Encryption Standard), Blowfish, IDEA, RC4,…

Giả sử A và B là hai nhóm người muốn truyền thông trên một kênh truyền không an toàn, họ có thể dùng mã hóa khoá bí mật như sau: Cả A và B thỏa thuận với nhau dùng một thuật toán cụ thể (ví dụ Rijindael) với một khóa và IV cụ thể. A soạn một thông điệp và tạo một dòng truyền trên mạng để gửi đi. Kế đó, A mã hóa văn bản dùng khóa và IV và gửi nó thông qua mạng A không gửi khóa và IV cho B. B nhận văn bản mã hóa và giải mã nó bằng khóa và IV đã thỏa thuận trước. Nếu việc truyền thông bị chặn thì người chặn không

thể phục hồi văn bản gốc do không biết khóa và IV. Trong kịch bản này, khóa phải được giữ bí mật còn IV thì không cần phải giữ bí mật. Trong kịch bản thế giới thực, A hay B phát sinh ra khoá bí mật và dùng mã hóa khóa công (bất đối xứng) để truyền khóa bí mật (bất đối xứng) .

Các hệ thống mã khóa bí mật tuy được sử dụng một thời gian dài và ít phức tạp về mặt toán học hơn các phương pháp mã hóa khóa công khai. Tuy nhiên vẫn còn một số hạn chế làm giới hạn khả năng của các hệ thống mã hóa khóa bí mật. Hai vấn đề gây hạn chế chính là:

- *Vấn đề phân phối khóa:* trước đây các hệ thống mã hóa khóa bí mật chủ yếu chỉ sử dụng trong quân đội và các tổ chức ngoại giao nên không có trở ngại gì khi thay đổi khoá hay truyền khóa, nhưng ngày nay nhu cầu nhiều người muốn truyền tin an toàn mà không cần biết nhau qua Internet dẫn đến việc thiết lập kênh truyền an toàn giữa hai người bất kỳ không khả thi.
- *Vấn đề quản lý khóa:* trong hệ thống có n người sử dụng, giả sử mỗi người muốn liên lạc an toàn với tất cả mọi người thì cần n(n-1)/2 khóa. Trong khi việc sử dụng cho các mục đích thương mại ngày nay tăng lên rất nhanh, việc lưu trữ khóa trở nên khó khăn.

Giới thiệu phương pháp DIFFIE-HELLMAN trong truyền khóa

Đây là giải pháp đầu tiên cho vấn đề phân phối khóa của hệ thống mã hóa khóa đối xứng. Nghi thức thực hiện như sau:

(1) A, B công khai chọn một nhóm nhân tuần hoàn Γ và một phần tử g $\in$ Γ

A chọn một số ngẫu nhiên a, và gửi ga đến B

B chọn một số ngẫu nhiên b, và gửi gb đến A

Sau khi nhận gb, A tính (gb)a

Tương tự, B tính (ga)b

A, B cùng chia xẻ một phần chung là gab đóng vai trò như khóa bí mật giữa họ. Nếu một kẻ tấn công biết được g, ga, gb thì cũng rất khó xác định gab. Bài toán này có quan hệ chặt chẽ đến bài toán logarit rời rạc.

2.1.3. Mã khóa công khai (Public-Key Encyption):

2.1.3.1. Giới thiệu

Mã hóa khóa công dùng khóa cá nhân được giữ bí mật đối với những người không quyền và một khóa công được mọi người biết đến. Cả khóa công và khóa bí mật có quan hệ toán học với nhau; dữ liệu được mã hóa với khóa công chỉ có thể được giải mã với khóa bí mật và dữ liệu được ký với khoá bí mật có thể được kiểm tra với khóa công. Khóa công sẵn có với bất kỳ ai. Cả hai khóa là duy nhất với phiên truyền. Các thuật toán mật mã khóa công cũng được

biết đến như các thuật toán bất đối xứng bởi vì một khóa được yêu cầu mã hóa dữ liệu trong khi khóa kia để giải mã dữ liệu.

Các thuật toán mật mã khóa công dùng kích thước vùng nhớ cố định trong khi các thuật toán mật mã khóa bí mật dùng vùng nhớ có chiều dài biến đổi. Các thuật toán khoá công không thể dùng dữ liệu dạng xích với các chuỗi dữ liệu theo cách các thuật toán khoá bí mật làm vì chỉ các khối lượng dữ liệu nhỏ có thể được mã hóa. Do đó, các thao tác bất đối xứng không dùng cùng mô hình dòng dữ liệu như các thao tác đối xứng.

Hai nhóm (A và B) có thể dùng mã hóa khóa công như sau: Đầu tiên A phát sinh cặp khóa khóa công khai – khoá bí mật. Nếu B muốn gửi cho A một thông điệp được mã hóa, B yêu cầu A khoá công khai. A gửi cho B khóa công khai của A trên mạng không an toàn và B dùng khóa này để mã hóa thông điệp (nếu B nhận khóa của A trên một kênh truyền không an toàn như mạng công cộng , B phải xác nhận với A rằng B nhận đúng bản sao khóa công khai của A). B gửi thông điệp được mã hóa cho A và A giải mã bằng khóa bí mật của A.

Trong quá trình truyền khoá công khai của A, một tác nhân không quyền có thể chặn khóa lại. Ngoài ra, cùng tác nhân này có thể chặn thông điệp mã hóa từ B. Nhưng tác nhân này không thể giải mã thông điệp với khóa công khai. Thông điệp chỉ có thể giải mã với khóa bí mật của A mà khóa này không được truyền. A không dùng khóa bí mật của A để mã hóa thông điệp hồi âm đến B vì bất cứ ai với khóa công khai có thể giải mã thông điệp được. Nếu A muốn gửi thông điệp lại cho B, A sẽ yêu cầu khóa công khai của B và mã hóa thông điệp dùng khóa công khai đó. Kế đó, B giải mã thông điệp dùng khóa bí mật tương ứng của B.

Trong thực tế, A và B dùng mã hóa khóa công khai (bất đối xứng) để truyền khóa bí mật (đối xứng) và dùng mã hóa khóa bí mật cho phần còn lại của phiên.

Mã hóa khoá công khai có không gian khóa hay phạm vi các giá trị có thể cho khóa lớn hơn, và do đó ít bị ảnh hưởng do tấn công vét cạn hơn khi thử mọi khóa có thể. Khóa công khai dễ phân phối bởi vì nó không phải bảo mật. Các thuật toán khóa công khai có thể được dùng để tạo chữ ký số để xác nhận định danh người gửi dữ liệu. Tuy nhiên, các thuật toán khóa công khai thì rất chậm (so với các thuật toán khóa bí mật) và thường không phù hợp khi mã hoá khối lượng dữ liệu lớn. Điển hình, mã hóa khóa công được dùng mã hóa khóa và IV được dùng bởi thuật toán khóa bí mật. Sau khi khóa và IV được truyền, kế đó mã hóa khóa bí mật được dùng cho phần còn lại của phiên.

RSA là một minh hoạ nổi tiếng của phương pháp mã hóa công khai. Tuy nhiên, RSA rất chậm trong khi yêu cầu bảo mật ngày nay không chỉ ứng dụng trên mạng có dây mà còn áp dụng cho mạng không dây và các thiết bị cầm tay bị hạn chế nhiều về khả năng lưu trữ, năng lượng, tốc độ nên một phương pháp

khóa công mới (ECC) đã ra đời vá đáp ứng các yêu cầu này. Cả hai phương pháp sẽ được trình bày chi tiết ở phần sau.

2.1.3.2. Phân loại hệ thống mã hóa khóa công:

Hiện nay, người ta chia các hệ thống mật mã khóa công khai chuẩn công nghiệp thành 3 loại dựa trên cơ sở toán học và độ khó để giải quyết bài toán tương ứng:

- *Bài toán phân tích ra thừa số nguyên tố (Integer Factorization Problem – IFP)*

Cho trước n là tích của hai số nguyên tố lớn, tìm hai số đó.

n =pq => Tìm p,q?

Các hệ thống dựa trên IFP gồm RSA, Rabin Williams,…Nhiều hệ thống chứng tỏ là an toàn, nghĩa là rất khó khi phân tích n thành các thừa số nguyên tố.

- *Bài toán logarit rời rạc (Discrete Logarithm Problem – DLP)*

Cho n là số nguyên tố, G = {i: $1 \le i \le n-1$}, a là một phần tử sinh của G nghĩa là bậc của a bằng n-1. Vậy ta có G = {a^i: $0 \le i \le n-2$}.

Với mỗi phần tử y thuộc G, có duy nhất một phần tử x thuộc G sao cho

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

phần tử x này được gọi là logarit rời rạc của y, với cơ số a.

Và ta có phép biến đổi logarit rời rạc G vào G, dùng để mã hóa thông tin. Để giải được bài toán logarit rời rạc thì cần tìm được x ($0 \le x \le n-1$). Tuy nhiên, hiện tại vẫn chưa có phương pháp nào giải bài toán logarit rời rạc hiệu quả.

Một ví dụ về hệ thống dựa trên DLP là hệ thống DSA mà chính phủ hoa kỳ đang sử dụng.

- *ECC (Elliptic Curve Cryptography): giải quyết bài toán logarit rời rạc trên đường cong Elip.*

2.1.4. Chữ ký điện tử

2.1.4.1. Giới thiệu:

Các thuật toán khóa công được dùng để tạo các chữ ký điện tử là một ứng dụng chứng thực quan trọng. Các chữ ký điện tử chứng thực định danh người gửi và bảo vệ sự toàn vẹn dữ liệu. Dùng khóa công khai được phát sinh

bởi A, người nhận dữ liệu của A có thể xác nhận rằng A đã gửi bằng cách so sánh chữ ký điện tử với dữ liệu và khoá công của A.

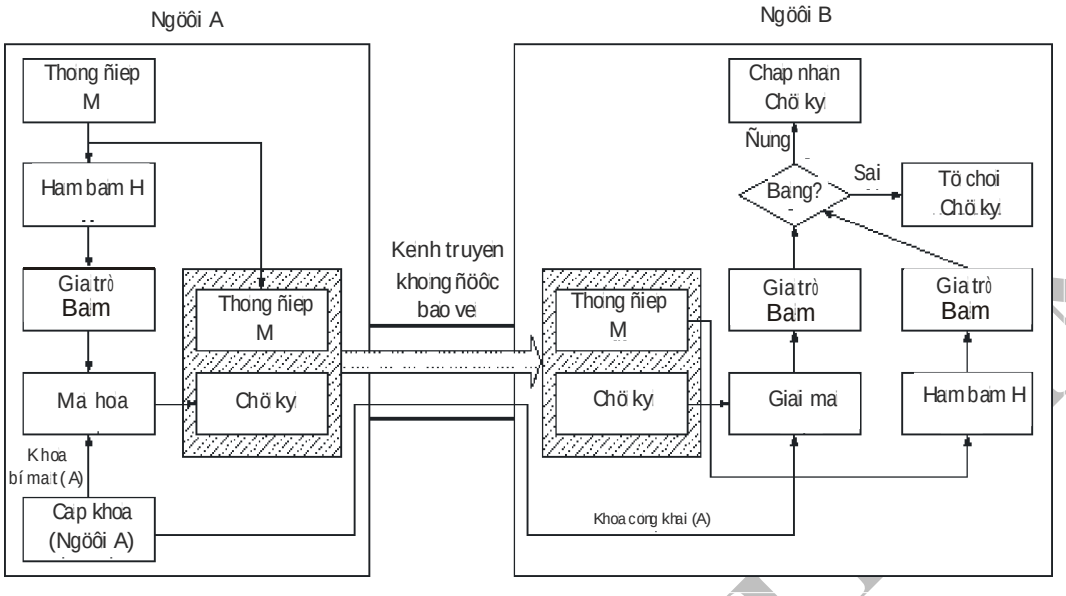
Một phương pháp chữ ký gồm 2 thành phần: thuật toán dùng để tạo ra chữ ký điện tử và thuật toán để xác nhận chữ ký điện tử. Và một phương pháp chữ ký điện tử là một bộ năm (P, K, A, S, V) thỏa các điều kiện sau:

- P là tập hữu hạn các thông điệp.
- A là tập hợp hữu hạn các chữ ký có thể sử dụng.
- Không gian khóa K là tập hợp hữu hạn các khóa có thể sử dụng.
- Với mỗi khóa k ∈ K, tồn tại thuật toán chữ ký và $sig_k \in S$, và thuật toán xác nhận chữ ký tương ứng $ver_k \in V$. Với mỗi thuật toán $sig_k : P \rightarrow A$, và P x A → {true, false} là các hàm thỏa điều kiện:

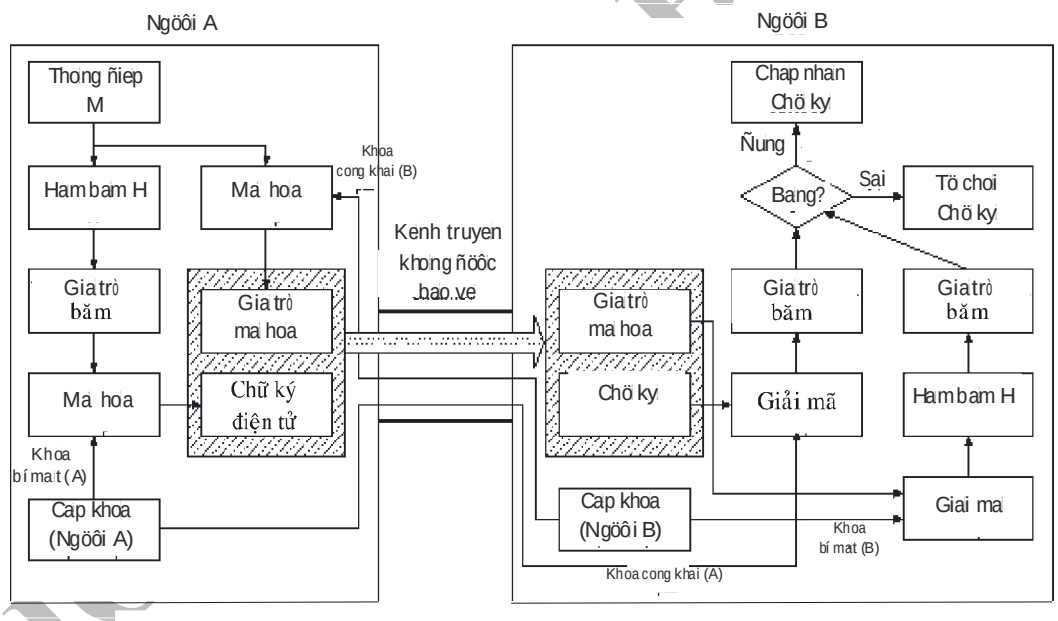
$$\forall x \in P, \forall y \in A : ver(x, y) = \begin{cases} \text{true} & \text{neu } y = sig(x) \\ \text{false} & \text{neu } y \neq sig(x) \end{cases}$$

Chuẩn DSS của NIST bắt đầu năm 1994 qui định các giải thuật chữ ký được chấp nhận, chiều dài khóa, chiều dài thông điệp, … đảm bảo tính bảo mật cao áp dụng vào các lĩnh vực kinh tế thương mại, ngân hàng, an ninh, quân đội, chính phủ, … DSS cũng công nhận 3 giải thuật chữ ký: DSA (Digital Signature Algorithm), RSA, ECDSA (Elipptic Curve Digital Signature Algorithm).

Để dùng mã hóa khóa công khai cho việc ký số một thông điệp. Đầu tiên A áp dụng thuật toán băm cho thông điệp để tạo mã khoá thông điệp. Mã khóa thông điệp là thể hiện duy nhất và mang tính cô động của dữ liệu. Kế đó A mã hóa mã khóa thông điệp với khóa bí mật của A để tạo ra chữ ký cá nhân. Khi nhận thông điệp và chữ ký, B giải mã chữ ký dùng khóa công khai của A để phục hồi mã khóa thông điệp, và băm thông điệp dùng cùng hàm băm mà A dùng. Nếu mã khóa thông điệp mà B làm trùng khớp mã khóa thông điệp A gửi. B đảm bảo là thông điệp này đến từ chủ nhân khóa bí mật và thông điệp không bị sửa đổi. Lưu ý là chữ ký có thể được kiểm chứng bởi bất cứ người nào do khóa công khai của người gửi thì phổ biến và nó được bao gồm trong định dạng của chữ ký số. Phương pháp này không giữ bí mật cho thông điệp đối với những thông điệp bí mật thì thông điệp này phải được mã hóa (Hình 2.1). Khi gửi đến B tùy theo mức độ bảo mật của thông điệp mà ta sẽ gửi bản rõ hay bản mã thông điệp.



Hình 2.1. Chữ ký điện tử được gửi cùng bản rõ thông điệp



Hình 2.2. Chữ ký điện tử được gửi cùng bản mã của thông điệp

2.1.4.2. Các đặc điểm của chữ ký điện tử:

§ *Tính xác nhận*: một chữ ký điện tử đảm bảo rằng chính người ký là người tạo ra nó.

§ *Tính an toàn* : không thể làm giả chữ ký nếu không biết thông tin bí mật để tạo chữ ký.

§ *Không thể dùng lại*: một chữ ký điện tử không thể dùng cho một tài liệu khác.
§ *Không thể phủ nhận*: một khi đã ký người ký không thể phủ nhận chữ ký đó.
§ *Tính hiệu quả*: ký và xác nhận nhanh chóng, dễ dàng.

2.2. Mã hóa đối xứng RC6

2.2.1. Giới thiệu RC6

RC6 là một cải tiến đáng kể của hệ mã RC5 (RC5 được công bố vào 12/1994), được thiết kế bởi Ronald Rivest (MIT, USA), cùng với Matt Robshaw, Ray Sidney, Yiqun Lisa Yin trong phòng nghiên cứu RSA nhằm đáp ứng những yêu cầu mới của AES (Advanced Encryption Standard) của NIST năm 1998. RC6 là một trong năm phương pháp được chọn đạt chuẩn AES của NIST ngày 2/10/2000.

RC6 là một thuật toán đơn giản, cài đặt nhanh hơn so với những phương pháp AES khác. RC6 cũng dựa trên các vòng lặp với phép toán quay vòng phụ thuộc dữ liệu đầu vào. RC6 mã hóa một lần 4 khối w bit thay vì 2 khối như RC5 để thỏa mãn yêu cầu xử lý khối dữ liệu đầu vào dài 128 bit (4 words x 32 bits) chuẩn của AES, và sử dụng các phép tính tích các số nguyên cũng như phép cộng số nguyên tố.

2.2.2. Thuật toán RC6

Một phiên bản RC6 được cụ thể một cách chính xác với tên RC6-w/r/b ở đó kích thước mỗi từ là w bits (chiều dài khối dữ liệu), mã hóa bao gồm một số vòng mã hóa không âm r, b là chiều dài khóa mã hóa theo byte. RC6-w/r/b hoạt động trên các đơn vị 4 từ w-bit dùng sáu phép tính cơ bản sau.

a + b : phép cộng mod 2^w.
a – b : phép trừ mod 2^w.
a ⊕ b: phép XOR trên từng bit của các từ w bit.
a . b : nhân mod 2^w.
a >>> b : quay trái từ a một lượng là lg(w) bit của từ b.
a <<< b : quay phải từ a một lượng là lg(w) bit của từ b.

2.2.2.1. Lập khóa:

Lập khóa của RC6-w/r/b cũng giống với lập khóa của RC5-w/r/b, chỉ khác là đối với RC6-w/r/b, nhiều từ xuất phát từ khóa người dùng cung cấp để dùng trong mã hóa và giải mã. Người dùng cung cấp một khóa b bytes ($0 \le b \le 255$), các byte zero được thêm vào để cho chiều dài khóa bằng với một số khác 0 các từ đầy đủ (có nghĩa là mỗi từ đủ 4 byte). Các byte khoá này kế đó được đưa vào một mảng c từ w-bit dạng little-endian L[0],…, L[c-1]. Do đó,

byte đầu tiên của khóa được lưu trữ như là byte vị trí thấp trong L[0],… và L[c-1] được làm đầy với các byte 0 nếu cần thiết (Chú ý nếu b = 0 thì c = 1 và L[0] = 0). 2r + 4 từ (mỗi từ w bit) được dẫn xuất từ khóa này và lưu trữ trong mảng S[0, 1, …, 2r + 3]. Mảng này được dùng cho cả mã hóa và giải mã. Các giá trị của hằng số P_{32} = 0xB7E15163, Q_{32} = 0x9E3779B9.

Thủ tục thực hiện:

```
S[0] = Pw
For i=1 to 2r+3 do
        S[i] = S[i-1] + Qw

A = B = i = j = 0

v = 3 x max{c, 3r + 4}
For s=1 to v do
{
        A = S[i] = (S[i] + A + B) <<<3
        B = L[j] = (L[j] +A + B) <<< (A + B)
        i = (i + 1) mod (2r + 4)
        j = (j + 1) mod c
}
```

2.2.2.2. Mã hóa và giải mã :

RC6 làm việc với 4 thanh ghi A, B, C, D mỗi thanh ghi dài w-bit chứa đầu vào là bản rõ, và đầu ra là bản mã ở cuối quá trình mã hóa. Sau khi phát sinh khóa mở rộng, dữ liệu được chia thành 4 khối cùng chiều dài w-bit ký hiệu A, B, C, D. Thực hiện r vòng mã hóa trên các khối này, mỗi vòng sử dụng 2 khóa mở rộng. Sau mỗi vòng lặp, vị trí các khối được biến đổi từ (A, B, C, D) thành (B, C, D, A). Cuối cùng, kết quả là (A, B, C, D). Quá trình giải mã làm ngược lại do RC6 là phương pháp mã đối xứng.

Thủ tục thực hiện :

- ***Mã hóa***

```
B = B + S[0]
D = D + S[1]

For i=1 to r do
{
        t = ( B x ( 2B + 1 )) <<< lg(w)
        u = ( D x ( 2D + 1 )) <<< lg(w)
        A = (( A ⊕ t) <<< u) + S[ 2i ]
        C = (( C ⊕ u) <<< t) + S[ 2i+1 ]
```

(A, B, C, D) = (B, C, D, A)
}
A = A + S[2r + 2]
C = C + S[2r + 3]

- ***Giải mã***

C = C – S [2r + 3]
A = A – S [2r + 2]

For i = r downto 1 do
{
(A, B, C, D) = (D, A, B, C)
u = (D x (2D + 1)) <<< lg(w)
t = (B x (2B + 1)) <<< lg(w)
C = ((C – S [2i + 1]) >>> t) ⊕ u
A = ((A – S[2i]) >>> u) ⊕ t
}

D = D – S[1]
B = B – S[0]

2.2.3. Nghi thức RC6

Để sử dụng hệ thống mã hoá khóa đối xứng RC6 ta làm theo các bước sau:

§ Người dùng cung cấp khóa (gọi là khóa người dùng) gồm b byte.
§ Lập khóa vòng S, thêm các byte 0 để tổng số byte của khóa người dùng là bội số của w-bit/8

Đầu vào : Khoá người dùng cung cấp b byte được đưa vào mảng c từ L[0], L[1],…, L[c-1].
Số vòng r.
Đầu ra: Khóa vòng w-bit S[0,…, r+3].

- ***Mã hóa:*** đưa bản rõ vào và mã hóa kết quả được bản mã.

Đầu vào : Văn bản rõ chứa trong 4 thanh ghi w-bit A, B, C, D
Số vòng r.
Khoá vòng w-bit S[0, 1, …, 2r+3].
Đầu ra : Văn bản mã được lưu trữ trong 4 thanh ghi A, B, C, D.

- ***Giải mã:*** làm ngược lại để được bản rõ.

Đầu vào : Bản mã được lưu trữ trong 4 thanh ghi đầu vào A, B, C, D.

Số vòng r.
Khóa vòng w-bit S[0, 1, …, 2r+3].

Đầu ra : Bản rõ lưu trữ trong A, B, C, D.

2.2.4. Đánh giá RC6

RC6 là giải thuật đơn giản dễ cài đặt nhất trong các phương pháp AES và tốc độ thực thi cao trên các bộ vi xử lý 32 bit; tuy nhiên, hiệu năng không bằng 4 giải thuật AES còn lại (Mars, Rijindael, Twofish, Sherpen) khi hoạt động trên bộ vi xử lý 8-bit và một số bộ xử lý 64 bit (do không hỗ trợ tốt phép nhân 64 bit). Về độ an toàn bảo mật thì không chênh lệch so với 4 giải thuật AES còn lại.

2.3. Phương pháp mã hóa khóa công RSA

2.3.1. Giới thiệu

RSA là hệ thống mật mã dùng khóa công được thiết kế bởi các giáo sư ở viện công nghệ MIT (USA): Ronald Rivest, Adi Sharmir, Leonard Adleman năm 1977 cho RSA Data Security inc. RSA là 3 chữ cái đầu của tên 3 giáo sư (Rivest, Sharmir, Adleman). Hệ thống mã hóa này dùng cho mã hóa, giải mã và mô hình chữ ký điện tử.

Thuật toán RSA được thiết kế dựa trên độ khó của bài toán phân tích ra thừa số nguyên tố trên tập số nguyên Z_n. Cho số nguyên dương n = p.q với p,q là 2 số nguyên tố. Nếu biết được n muốn tìm p, q thì phải giải bài toán phân tích ra thừa số nguyên tố đòi hỏi phải thực hiện một số lượng các phép tính vô cùng lớn nếu n,p,q đủ lớn (ít nhất là 100 ký số); hiện tại, điều này là không khả thi. Tuy nhiên, trong tương lai với sự ra đời của các công nghệ máy tính cực mạnh thì bài toán phân tích ra thừa số nguyên tố vẫn có thể giải được.

2.3.2. Thuật toán RSA

Thuật toán RSA chủ yếu dựa trên 3 số nguyên dương <d, e, n> với
e : số mũ công khai
d : số mũ cá nhân
n : hằng số của phương pháp

Cặp <e, n> được gọi là khóa công khai, còn cặp <d, n> được gọi là khóa cá nhân.

<u>Cơ sở toán học của hệ mã RSA là định lý Euler</u>: "Nếu n và a là 2 số nguyên tố cùng nhau, ta có: $a^{\varphi(n)} = 1 (\text{mod } n)$*; trong đó,* $\varphi(n)$ *là số số nguyên trong đoạn [1, n] và nguyên tố cùng nhau với n "*

Ta có : Nếu n là số nguyên tố, $\varphi(n) = n - 1$.
Nếu n là tích của 2 số nguyên tố p và q : n=p.q thì
$\varphi(n) = (p - 1)(q - 1)$

Giả sử thông điệp số m là một số nguyên trong đoạn [2, n – 1]; m sẽ được mã hóa bằng khóa công khai <e, n> thành số nguyên c:

$$c = m^e \pmod n$$

Nếu số nguyên d, trong khóa cá nhân <d, n> thỏa điều kiện

$$e.d = 1 \pmod{\varphi(n)}$$

thì tồn tại một số nguyên k sao cho: $e.d = k.\varphi(n) + 1$.

Và $c^d \pmod n = (m^e \pmod n)^d \pmod n$
$= m^{ed} \pmod n$
$= (m^{k.\varphi(n)} \pmod n)\ (m \pmod n)$
$= (1^k \pmod n)\ (m \pmod n)$
$= m.$

Có thể tìm lại bản thông điệp m từ khóa cá nhân <d, n> và bản mã c theo công thức :

$$m = c^d \pmod n$$

2.3.3. Nghi thức RSA

1. Tạo ngẫu nhiên 2 số nguyên tố p,q phân biệt, gần nhau, và rất lớn (có số ký số ít nhất là 100). Và tính :
   $n = p.q$
   $\varphi(n) = (p - 1)(q - 1)$
2. Chọn ngẫu nhiên một số e, $1 < e < \varphi(n)$, sao cho e nguyên tố cùng nhau với $\varphi(n)$.
3. Tính số nghịch đảo d của e đối với $\varphi(n)$: $1 < d < \varphi(n)$, $ed = 1 \pmod{\varphi(n)}$ (dùng thuật toán Eulid mở rộng).

- ***Mã hóa:*** B muốn gửi thông điệp m cho A với yêu cầu cần bảo mật thông tin.
  - B yêu cầu A gửi khóa công khai <e, n>.
  - B dùng khóa công khai <e, n> này , mã hóa thông điệp m thành c theo công thức $c = m^e \pmod n$.
  - B gửi c cho A.
- ***Giải mã***: dùng khóa cá nhân <d, n>, A tính $m = cd \pmod n$, để có nguyên bản m từ bản mã c.

2.3.4. Đánh giá RSA

RSA đơn giản, dễ hiểu, dễ cài đặt.

Hiệu suất hoạt động: RSA chạy chậm do việc phát sinh khóa công khai - khoá bí mật hay quá trình mã hóa - giải mã tốn nhiều thời gian vì phải tính toán trên các số nguyên dương cực lớn, có chiều dài vượt quá khả năng chứa của thanh ghi nên phải giả lập, thực hiện lại nhiều lần và sử dụng nhiều đến bộ xử lý. Do đó, RSA không được sử dụng vào mục đích mã hóa các khối lượng dữ liệu lớn mà chỉ ứng dụng trong chữ ký điện tử để mã hóa thông điệp ngắn đã qua hàm băm, giải thuật trao đổi khóa bí mật (khóa dùng cho các hệ thống mã hóa đối xứng – hay khóa riêng), hay chỉ mã hóa một lượng dữ liệu nhỏ.

Tính bảo mật: như đã trình bày ở trên, độ an toàn của RSA dựa trên bài toán phân tích ra thừa số nguyên tố. Do đó, chiều dài số càng lớn thì càng khó phân tích ra thừa số nguyên tố.

2.4. Hệ mã hóa ECC (Elliptic Curve Cryptography)

2.4.1. Giới thiệu

ECC đầu tiên được đề xuất độc lập nhau bởi Victor Miller và Neal Koblitz năm 1985 và được xem như là một lựa chọn cùng với các thuật toán khóa công khác (RSA, DSA, DH). ECC có cùng mức độ bảo mật nhưng với kích thước khóa nhỏ hơn làm cho việc tính toán ít hơn, sử dụng năng lượng thấp hơn, cũng như tiết kiệm băng thông và bộ nhớ hơn, chính những đặc điểm này mà ECC trở nên thu hút đối với mạng không dây và các thiết bị cầm tay, các máy chủ Web an toàn, thẻ thông minh, điện thoại tế bào, và máy nhắn tin,… mà các phương pháp mã hóa khóa công khai khác không thể đáp ứng được.

ECC được thiết kế dựa trên độ khó trong việc giải bài toán logarit rời rạc trên đường cong Elip (ECDLP_ Elliptic Curve Discrete Logarithm Problem). ECC được dùng trong truyền khóa, chữ ký điện tử, chứng thực, truyền thông điệp, …

2.4.2. Một số khái niệm

Các tính toán trên số thực thường rất chậm và tạo ra kết quả không chính xác do việc làm tròn số thực. Các ứng dụng trên mã hóa cần tính toán nhanh và chính xác nên các trường hữu hạn được quan tâm nhiều.

Một trong những điểm khác biệt quan trọng giữa đường cong Elip phân bố trên trường hữu hạn và trên trường số thực là số hữu hạn các điểm của nó, là một thuộc tính mong muốn trong mục tiêu mã hóa. Do các đường cong này

chứa một số điểm rời rạc, nên khi nối các điểm lại với nhau thì đồ thị này không rõ ràng là đường cong. Các quy luật đại số khi tính toán có thể được chuyển về các đường cong Elip trên trường hữu hạn. Ngoài ra, các tính toán trên trường hữu hạn không chứa lỗi làm tròn – một đặc tính quan trọng yêu cầu cho một hệ thống mã hóa.

2.4.2.1. Trường hữu hạn

- **Khái niệm**

Một trường hữu hạn bao gồm một tập hữu hạn các đối tượng gọi là các phần tử trường cùng với việc định nghĩa hai phép toán: phép cộng và phép nhân. Trường hữu hạn có q phần tử nếu và chỉ nếu q là lũy thừa của một số nguyên tố và tương ứng với mỗi q sẽ có một trường. Trường hữu hạn chứa q phần tử được ký hiệu là Fq.

Có hai loại trường hữu hạn Fq được dùng:

(i) Fp với q = p, p là số nguyên tố lẻ và được gọi là các trường hữu hạn số nguyên tố.

(ii) Trường hữu hạn F_2^m với $q = 2^m$ với m bất kỳ ($m \geq 1$) gọi là các trường nhị phân hữu hạn.

- **Các trường hữu hạn**

Ø *Trường hữu hạn Fp*

Trường hữu hạn Fp là trường hữu hạn số nguyên tố chứa p phần tử. Chỉ có một trường Fp cho mỗi số nguyên tố lẻ p, nhưng có nhiều cách khác nhau biểu diễn các phần tử của Fp, luận văn này sử dụng cách biểu diễn Fp là tập các số nguyên : {0, 1, .., p – 1} với phép cộng và phép nhân được định nghĩa như sau:

- Phép cộng: Nếu a, b$\in$ Fp thì $a + b \equiv r \pmod p$.
- Phép nhân: Nếu a, b$\in$ Fp thì $a.b \equiv s \pmod p$.

Ngoài ra, ta còn định nghĩa nghịch đảo của phép cộng và phép nhân giữa các phần tử trường như sau:

- Phép cộng nghịch đảo: Nếu a $\in$ Fp, thì phép cộng nghịch đảo (- a) của a trong trường Fp là nghiệm duy nhất cho phương trình $a + x \equiv 0 \pmod p$.
- Phép nhân nghịch đảo: Nếu a $\in$ Fp, $a \neq 0$ thì phép nhân nghịch đảo a^{-1} của a trong trường Fp, là nghiệm duy nhất cho phương trình $a.x \equiv 1 \pmod p$ (dùng thuật toán Eulid mở rộng để tính a^{-1}).

Phép chia và phép trừ được biểu diễn qua phép cộng nghịch đảo và phép nhân nghịch đảo: (a – b) mod p là a + (– b) mod p và a / b mod p là $a.(b^{-1}) \bmod p$. Trường Fp được dùng nên thỏa :

$[\log_2 p] \in \{112, 128, 160, 192, 224, 256, 384, 521\}$

Ø *Trường hữu hạn* F_2^m

Trường hữu hạn F_2^m là trường nhị phân hữu hạn chứa 2^m phần tử. Chỉ có một trường hữu hạn F_2^m duy nhất cho mỗi lũy thừa 2^m với $m \geq 1$. Có nhiều cách biểu diễn các phần tử của trường F_2^m. Trong luận văn này, trường F_2^m được biểu diễn dưới dạng tập các đa thức nhị phân bậc nhỏ hơn bằng $(m – 1)$:

$$\{a_{m-1}x_{m-1} + a_{m-2}x_{m-2} + \ldots + a_1x + a_0, a_i \in \{0, 1\}\}$$

với phép cộng và phép nhân được định nghĩa dưới dạng một đa thức nhị phân tối giản $f(x)$ bậc m là một đa thức rút gọn như sau:

§ Phép cộng: Nếu $a = a_{m-1}x_{m-1} + a_{m-2}x_{m-2} + \ldots + a_1x + a_0$, $b_{m-1}x_{m-1} + b_{m-2}x_{m-2} + \ldots + b_1x + b_0 \in F_2^m$, thì $a + b = r$ trong trường F_2^m, với $r = r_{m-1}x_{m-1} + r_{m-2}x_{m-2} + \ldots + r_1x + r_0$ với $r_i \equiv a_i + b_i \pmod 2$.

§ Phép nhân: Nếu $a = a_{m-1}x_{m-1} + a_{m-2}x_{m-2} + \ldots + a_1x + a_0$, $b_{m-1}x_{m-1} + b_{m-2}x_{m-2} + \ldots + b_1x + b_0 \in F_2^m$, thì $a.b = s$ trong trường F_2^m, với $s = s_{m-1}x_{m-1} + s_{m-2}x_{m-2} + \ldots + s_1x + s_0$ là phần dư khi đa thức $a.b / f(x)$ với tất cả hệ số được biểu diễn dạng mod 2.

Để định nghĩa phép trừ và phép chia, ta cần định nghĩa nghịch đảo của phép cộng và phép nhân:

§ Phép cộng nghịch đảo: Nếu $a \in F_2^m$, thì nghịch đảo $(-a)$ của a trong trường F_2^m là nghiệm duy nhất của $a + x = 0 \pmod p$ trong trường F_2^m.

§ Phép nhân nghịch đảo: Nếu $a \in F_2^m$, $a \neq 0$ thì nhân nghịch đảo a^{-1} của a trong trường F_2^m là nghiệm duy nhất của $a.x \equiv 1 \pmod p$ trong trường F_2^m.

Phép chia và phép trừ được biểu diễn dưới dạng phép cộng nghịch đảo và phép nhân nghịch đảo: $(a – b)$ trong trường F_2^m là $a + (– b)$ trong trường F_2^m và a / b trong trường F_2^m là $a.(b^{-1})$ trong trường F_2^m.

Trong thực tiễn thường dùng F_2^m với $m \in \{113, 131, 163, 193, 233, 239, 283, 409, 571\}$ và phép cộng và phép nhân được biểu diễn dùng một trong những đa thức nhị phân tối giản bậc m như trong bảng sau [4]:

Trường	Đa thức tối giản
F_2^{113}	$f(x) = x^{113} + x^9 + 1$
F_2^{131}	$f(x) = x^{131} + x^8 + x^3 + x^2 + 1$
F_2^{163}	$f(x) = x^{163} + x^7 + x^6 + x^3 + 1$
F_2^{193}	$f(x) = x^{193} + x^{15} + 1$

F_2^{233}	$f(x) = x^{233} + x^{74} + 1$
F_2^{239}	$f(x) = x^{239} + x^{36} + 1$ hay $f(x) = x^{239} + x^{158} + 1$
F_2^{283}	$f(x) = x^{283} + x^{12} + x^7 + x^5 + 1$
F_2^{409}	$f(x) = x^{409} + x^{87} + 1$
F_2^{571}	$f(x) = x^{571} + x^{10} + x^5 + x^2 + 1$

Bảng 1: biểu diễn trường F_2^m

- **Nhận xét về trường hữu hạn Fp và F2m**

Fp	F2m
Dựa trên các tính toán số học (mod) cũ	Tính toán nhanh
Dễ nâng cấp từ RSA	Hiệu quả khi cài đặt cho phần cứng
Chậm khi cài đặt cho phần cứng	

2.4.2.2. Một số đặc tính Elip trên trường hữu hạn

- **Một số khái niệm**

Cho đường cong Elip có phương trình theo *công thức Weierstrass dài*:

Y2 + a1XY + a3Y = X3 + a2X2 + a4X + a6, với ai ∈ P (1)

Gọi E là một Elip được định nghĩa trên trường hữu hạn Fq bất kỳ (trường Fq có q phần tử).

§ *Bậc của một điểm trên Elip:* Nếu P ∈ E(Fq) là một điểm của Elip thì bậc của P là số dương n nhỏ nhất, sao cho nP = *O* với *O* là điểm trung hòa của Elip (hay còn gọi là điểm ở vô cùng).

§ *Số điểm của một đường cong Elip trên Fq* được gọi là bậc của Elip trên trường Fq, ký hiệu #E(Fq) hay viết tắt là #(E).

§ *Đinh lý Hasse về số điểm trên đường cong E:*

$$q + 1 - 2\sqrt{q} \le \#E(Fq) \le q + 1 + 2\sqrt{q}$$

§ *Biệt số (discrimant) và lượng bất biến j (j-invariant) của Elip:*

Gọi E là Elip được định nghĩa bằng công thức Weiertrass dài, định nghĩa các thông số sau:

$$d_2 = a_1^2 + 4a_2$$
$$d_4 = 2a_4 + a_1 a_3$$
$$d_6 = a_3^2 + 4a_6$$
$$d_8 = a_1^2 a_6 + 4a_2 a_6 - a_1 a_3 a_4 + a_2 a_3^2 - a_4^2$$
$$c_4 = d_2^2 + 24 d_4$$
$$\Delta = -d_2^2 d_8 - 8 d_4^3 - 27 d_6^2 + 9 d_2 d_4 d_6$$
$$j(E) = c_4^3 / \Delta$$

Δ được gọi là biệt số của công thức Weierstrass, và j(E) được gọi là lượng bất biến j của E nếu $\Delta \neq 0$.

- **Các lớp Elip đặc biệt**

Có vô số lớp Elip, nhưng chúng ta cần xem xét một số lớp Elip đặc biệt vì chúng có liên quan đến tính bảo mật của hệ thống.

§ *Lớp Elip kỳ dị (supersingular):* các Elip kỳ dị có một số tính chất đặc biệt cho trường F_2^m như sau: một Elip trên trường hữu hạn có q phần tử (q = #E) được nói là kỳ dị nếu t^2 = 0, q, 2q, 3q, 4q với t = q + 1 - #E(Fq), $|t| \leq 2\sqrt{q}$ (theo định lý Hasse). Một Elip E là kỳ dị nếu j(E) = 0.

§ *Các lớp Elip không kỳ dị (non-supersingular):* một Elip không kỳ dị là một Elip có lượng bất biến j ≠ 0. Ưu điểm của một Elip không kỳ dị so với Elip kỳ dị là nó có thể cho cùng mức độ bảo mật, nhưng với trường nhỏ hơn.

2.4.2.3. Khảo sát đường cong Elip

- **Đường cong Elip trên Fp**

Đặt Fp là trường hữu hạn số nguyên tố với p là số nguyên tố lẻ, và a,b ∈ Fp thỏa $4a^3 + 27b^2 \neq 0$ (mod p). Đường cong elip E(Fp) trên trường Fp được định nghĩa bởi các thông số a, b ∈ Fp bao gồm một tập hợp các nghiệm hay các điểm P = (x, y), với x, y ∈ Fp, của phương trình:

$$y^2 = x^3 + ax + b \pmod p$$

với *O* gọi là điểm ở vô cùng. Phương trình $y^2 = x^3 + ax + b$ (mod p) được gọi là phương trình xác định của E(Fp) với P(xp, yp) được cho trước. Số điểm trên E(Fp) ký hiệu là #E(Fp).

Theo đinh lý Hasse, ta có số điểm trên đường cong E trên Fp*:*

$$p + 1 - 2\sqrt{p} \le \#E(Fp) \le p + 1 + 2\sqrt{p}$$

*Các luật cộng điểm trên E(F*p*):*

1. $O + O = O$
2. $(x, y) + O = O + (x, y) = (x, y)$, $\forall (x, y) \in E(Fp)$
3. $(x, y) + (x, -y) = O$, $\forall (x, y) \in E(Fp)$ với $-(x, y) = (x, -y)$
4. $x_3 \equiv \lambda^2 - x_1 - x_2 \pmod p$, $y_3 \equiv \lambda(x1 - x3) - y1 \pmod p$ và
   $\lambda = \frac{y_2 - y_1}{x_2 - x_1} \pmod p$
5. $(x1, y1) \in E(F_p)$ là điểm với $y1 \neq 0$ thì $(x1,y1) + (x2,y2) = (x3,y3)$ với $x_3 \equiv \lambda^2 - 2.x_1$, $y_3 \equiv \lambda(x1 - x3) - y_1 \pmod p$ và
   $\lambda = \frac{3.x_1^2 + a}{2.y_1} \pmod p$

Tập các điểm trên E(Fp) hình thành nhóm hoán vị, *nhóm Abel* nghĩa là P1 + P2 = P2 + P1, ∀P1, P2 ∈ E(Fp). Chú ý tính hoán vị có thể luôn luôn tính được một cách hiệu quả qua các phép số học đơn giản.

Các hệ mã ECC dựa trên *phép nhân vô hướng* của các điểm đường cong Elip. Cho trước số nguyên k và một điểm P ∈ E(Fp), phép nhân vô hướng là quá trình cộng P vào chính nó k lần. Kết quả phép nhân vô hướng được ký hiệu: k × P hay kP. Phép nhân vô hướng giữa các điểm của đường cong Elip được tính toán một cách hiệu quả dùng tính hoán vị cùng với thuật toán "gấp đôi và cộng" (double – and – add) hay một trong những biến của nó.

q Đường cong Elip trên F_2^m

Đặt F_2^m là trường nhị phân hữu hạn, và đặt a, b ∈ F_2^m, b≠0 trong trường F_2^m thì đường cong Elip không kỳ dị $E(F_2^m)$ trên trường F_2^m được định nghĩa bằng các thông số a, b ∈ F_2^m là tập các nghiệm hay các điểm P = (x, y) với x, y ∈ F_2^m của phương trình:

$y^2 + x.y = x^3 + a.x^2 + b$ trong trường F_2^m

O gọi là điểm ở vô cùng. Số điểm trên $E(F_2^m)$ được ký hiệu $\#E(F_2^m)$

Theo đinh lý Hasse, ta có số điểm trên đường cong E trên F_2^m*:*

$$2^m + 1 - 2\sqrt{2^m} \le \#E(F_2^m) \le 2^m + 1 + 2\sqrt{2^m}$$

Các luật cộng các điểm trên E:

1. $O + O = O$
2. $(x, y) + O = O + (x, y) = (x, y)$, $\forall (x, y) \in E(F2m)$
3. $(x, y) + (x, x+y) = O$, $\forall (x, y) \in E(F2m)$ với $-(x, y) = (x, x+y)$

4. Đặt $(x_1, y_1), (x_2, y_2) \in E(F2m)$ là 2 điểm $x_1 \neq x_2$ thì $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$ nếu $x_3 = \lambda^2 + \lambda + x_1 + x_2 + a$ trong trường F2m và $y_3 = \lambda (x_1 + x_3) + x_3 + y_1$ và $\lambda \equiv \frac{y_1 + y_2}{x_1 + x_2}$ trong trường F2m
5. Đặt $(x_1, y_1) \in E(F2m)$ là một điểm với $x_1 \neq 0$ thì $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$ nếu $x_3 = \lambda^2 + \lambda + a$ trong trường F2m và $y3 = x_1^2 + (\lambda + 1).x_3$ trong trường F2m và $\lambda = x_1 + \frac{y_1}{x_1}$ trong trường F2m

Tập các điểm trên $E(F_2^m)$ hình thành nhóm giao hoán Abel. Luật giao hoán dùng phép số học đơn giản để tính. Các hệ mã ECC dựa trên phép nhân vô hướng giữa các điểm đường cong Elip. Cho trước số nguyên k và một điểm $P \in E(F2m)$, phép nhân vô hướng là tiến trình thêm P vào chính nó k lần. Kết quả của phép nhân vô hướng ký hiệu $k \times P$ hay kP.

2.4.3. Các thành tố mật mã trong ECC

2.4.3.1. Các thông số miền đường cong Elip

Hoạt động của các hệ mã khóa công bao gồm các phép tính số học trên đường cong elip dựa trên một trường hữu hạn được xác định bởi các tham số miền đường cong Elip.

q Các thông số miền đường cong Elip trên trường Fp

Các thông số miền đường cong Elip trên trường Fp là bộ sáu

$$T = (p, a, b, G, n, h)$$

gồm một số nguyên tố lẻ p tạo thành trường Fp, hai thành phần $a, b \in Fp$ hình thành đường cong Elip được định nghĩa bởi phương trình

$$E: y^2 = x^3 + ax + b \pmod p$$

Điểm cơ sở $G = (x_G, y_G)$ trên E(Fp), số nguyên tố n cho biết bậc của G, và số nguyên h là phần bù đại số $h = \#E(Fp) / h$. Các thông số miền đường cong Elip trên trường Fp xác định chính xác đường cong Elip và điểm cơ sở. Việc này cần thiết để định nghĩa các hệ mã khoá công khai dựa trên ECC.

Phát sinh các thông số miền đường cong Elip trên trường F_p

Đầu vào: Một số nguyên $t \in \{56, 64, 80, 96, 112, 128, 192, 256\}$
Đầu ra: Các thông số miền đường cong trên trường *F*p
$T = (p, a, b, G, n, h)$
Thủ tục thực hiện:

1. Chọn một số nguyên tố p sao cho $[\log_2 p] = 2t$ nếu $t \neq 256$, và $[\log_2 p] = 521$ nếu $t = 256$ để xác định trường hữu hạn *F*p

2. Chọn a, b ∈ F_p để xác định đường cong elip E(F_p) qua phương trình:

$$E: y^2 = x^3 + a.x + b \pmod p$$

Một điểm cơ sở G = (x_G, y_G) trên E(F_p), số nguyên tố n là bậc của G, và số nguyên h là phần bù đại số h = #E(F_p) / n, với những ràng buộc sau:

- § $4.a^3 + 27.b^2 \neq 0 \pmod p$
- § #E(Fp) ≠ p
- § $p^B \neq 1 \pmod p$ với $1 \leq B \leq 20$
- § h ≤ 4

3. Return T = (p, a, b, G, n, h)

q Các thông số miền đường cong Elip trên trường F_2^m

Các thông số trên đường cong Elip trên trường F2m là bộ bảy

T = (m, f(x), a, b, G, n, h)

gồm số nguyên m tạo thành trường F2m, một đa thức nhị phân tối giản f(x) bậc m đại diện F2m, hai thành tố a,b ∈ F2m tạo thành đường cong Elip E(F2m) qua phương trình:

$y^2 + x.y = x^3 + a.x^2 + b$ trong trường F2m

một điểm cơ sở G = (x_G, y_G) trên E(F2m), một số nguyên tố n chỉ bậc của G, và một số nguyên h là phần bù đại số h = #E(F2m) / n

Phát sinh các thông số miền đường cong Elip trên trường F_2^m

Đầu vào: Một số nguyên t ∈ {56, 64, 80, 96, 112, 128, 192, 256}
Đầu ra: Các thông số miền đường cong Elip trên trường F_2^m:
T = (m, f(x), a, b, G, n, h)
Thủ tục thực hiện:

1. Gọi t' là số nguyên nhỏ nhất lớn hơn t trong tập {64, 80, 96, 112, 128, 192, 256, 512}. Chọn m ∈ {113, 131, 163, 193, 233, 239, 283, 409, 571} sao cho 2t<m<2t' để xác định trường hữu hạn F_2^m.
2. Chọn một đa thức tối giản f(x) có bậc m từ bảng 1 làm đại diện cho trường F_2^m
3. Chọn a, b ∈ F_2^m để xác định đường cong E(F_2^m) qua phương trình:

$$E : y^2 + x.y = x^3 + a.x^2 + b \text{ trong trường } F_2^m$$

Một điểm cơ sở G = (x_G, y_G) trên đường cong E(F_2^m), một số nguyên tố n chỉ bậc của G, và một số nguyên h là phần bù đại số h=#E(F_2^m)/n, thỏa các ràng buộc sau:

- § b ≠ 0 trên trường F_2^m
- § #E(F_2^m) ≠ 2^m
- § $2^{mB} \neq 1 \pmod n$ với $1 \leq B \leq 20$

§ $h \leq 4$

4. Return T = (m, f(x), a, b, G, n, h)

2.4.3.2. Cặp khóa đường cong Elip

Cho các thông số miền đường cong T trên trường Fp hay F2m thì một cặp khóa (d, Q) liên quan đến T bao gồm khóa bí mật đường cong Elip d là một số nguyên thuộc khoảng [1, n-1] và khóa công khai đường cong Elip Q = (x_Q, y_Q) là điểm thỏa Q = dG.

Phát sinh cặp khoá đường cong Elip

Đầu vào: Các thông số miền đường cong có giá trị T = (p, a, b, G, n, h) hay T = (m, f(x), a, b, G, n, h)

Đầu ra: Một cặp khóa đường cong (d, Q) liên quan đến T

Thủ tục thực hiện phát sinh cặp khóa đường cong Elip:

1. Chọn ngẫu nhiên hay giả ngẫu nhiên một số nguyên d trong khoảng [1, n-1]
2. Tính Q = dG
3. Kết xuất (d, Q)

Kiểm tra tính hợp lệ của khóa công khai Q

1. $Q \neq O$
2. Nếu xét T trên trường F_p thì kiểm tra x_Q, y_Q là những số nguyên trong khoảng [1, p–1] và:

   $y_Q^2 = x_Q^3 + a.x_Q + b \pmod p$

   Nếu xét T trên trường F_2^m thì kiểm tra x_Q, y_Q là những đa thức nhị phân có bậc tối đa là (m – 1) , và:

   $y_Q^2 + x_Q.y_Q = x_Q^3 + a.\ x_Q^2 + b$ trong trường F_2^m
3. $nQ = O$
4. Nếu có một điều kiện không thỏa thì khóa công khai không hợp lệ.

2.4.4. Các lược đồ trong ECC

Có 5 loai dữ liệu dùng cho dữ liệu trong các lược đồ trong ECC

§ Chuỗi bit

§ Điểm EC

§ Chuỗi 8 bit: một chuỗi bit được thêm các bit 0 sao cho kết quả chiều dài của nó là bội số của 8, sau đó chia chuỗi kết quả thành các chuỗi dài 8 bit.

§ Số nguyên

§ Các thành phần trường: bao gồm các thành phần thuộc trường Fp hay F2m.

2.4.4.1. Lược đồ chữ ký điện tử dựa trên ECC

Lược đồ chữ ký điện tử được thiết kế cho 2 bên - người ký U và người xác nhận V – khi U muốn gửi một thông điệp M cần chứng thực và V muốn xác thực đó của thông điệp M. Lược đồ chữ ký điện tử được mô tả dưới hành động ký gửi, hành động xác nhận, hành động thiết lập và triển khai khóa.

Thuật toán chữ ký điện tử (Elliptic Curve Digital Signature Algorithm _ ECDSA)

- **Thiết lập lược đồ**

1. U dùng hàm băm khi phát sinh chữ ký. Gọi Hash là hàm băm được chọn và hashlen là chiều dài chuỗi giá trị băm được sinh ra khi sử dụng hàm băm Hash.
2. U thiết lập các thông số miền đường cong Elip T = (p,a,b,G,n,h) hay T = (m, f(x), a, b, G, n, h) với mức độ bảo mật mong muốn.
3. V đạt được hàm băm và các thông số trong trạng thái chứng thực được thiết lập bởi U.

- **Triển khai khóa**

1. U thiết lập cặp khóa (d_U, Q_U) liên quan đến T dùng với lược đồ chữ ký.
2. V nhận được khóa công khai Q_U trong trạng thái chứng thực được xác nhận có giá trị.

- **Hành động ký**

U ký thông điệp dùng ECDSA bằng cách dùng các khóa và các thông số được thiết lập trong suốt thủ tục thiết lập và thủ tục triển khai khóa:

Đầu vào: Thông điệp M dạng chuỗi 8 bit
Đầu ra: Chữ ký S = (r, s) trên M bao gồm cặp số nguyên r và s hay xuất "không hợp lệ"
Thủ tục thực hiện:

1. Chọn một cặp khóa (k, R) với R = (x_R, y_R) liên quan đến các thông số T được thiết lập trong thủ tục thiết lập bằng cách dùng việc phát sinh cặp khóa.
2. Biến đổi thành phần trường x_R thành $\overline{x_R}$ (dạng số nguyên của thành phần trường x_R) (xem cách chuyển đổi ở phụ lục B).
3. Đặt r = $\overline{x_R}$ (mod n). Nếu r = 0, quay về bước 1.

4. Tính giá trị băm dùng hàm băm Hash đã chọn H = Hash(M). Nếu hàm băm báo "không hợp lệ", xuất "không hợp lệ" và ngừng.
5. Phát sinh một số nguyên e từ H như sau:
   5.1. Biến đổi chuỗi H thành chuỗi bit $\overline{H}$.
   5.2. Đặt $\overline{E} = \overline{H}$ nếu $[\log_2 n] \geq 8$(hashlen), và đặt $\overline{E}$ bằng $[\log_2 n]$ bits bên trái của $\overline{H}$ nếu $[\log_2 n] < 8$(hashlen).
   5.3. Biến đổi chuỗi bit $\overline{E}$ thành chuỗi octet E
   5.4. Chuyển chuỗi octet E thành số nguyên e.
6. Tính:
   $s = k^{-1}.(e + r.\, d_U) \pmod n$
   Nếu s = 0, quay về bước 1
7. Kết xuất S = (r, s).

q **Hành động xác nhận**

V xác nhận thông điệp từ U dùng ECDSA bằng cách dùng các khóa và các thông số được thiết lập trong thủ tục thiết lập và thủ tục triển khai khóa:

Đầu vào: Chuỗi octet thông điệp M và chữ ký của U là S = (r, s)
Đầu ra: Chữ ký trên M có giá trị hay không
Thủ tục thực hiện:

1. Nếu r và s đều là số nguyên trong khoảng [1, n-1], xuất "có giá trị" và ngừng.
2. Tính giá trị băm dùng hàm băm Hash đã chọn H = Hash(M). Nếu hàm băm báo "không hợp lệ", xuất "không hợp lệ" và ngừng.
3. Phát sinh một số nguyên e từ H như sau:
   3.1. Biến đổi chuỗi H thành chuỗi bit $\overline{H}$.
   3.2. Đặt $\overline{E} = \overline{H}$ nếu $[\log_2 n] \geq 8$(hashlen), và đặt $\overline{E}$ bằng $[\log_2 n]$ bits bên trái của $\overline{H}$ nếu $[\log_2 n] < 8$(hashlen).
   3.3. Biến đổi chuỗi bit $\overline{E}$ thành chuỗi octet E
   3.4. Chuyển chuỗi octet E thành số nguyên e.
4. Tính
   $u_1 = e.s^{-1} \pmod n$
   $u_2 = r.s^{-1} \pmod n$
5. Tính
   $R = (x_R, y_R) = u_1 G + u_2 Q_U$
   Nếu R = *O*, xuất "không hợp lệ" và dừng.
6. Biến đổi thành phần trường x_R thành một số nguyên $\overline{x_R}$.
7. Đặt $v = \overline{x_R} \pmod n$
8. So sánh v và r nếu v = r thì xuất "hợp lệ" , nếu v ≠ r thì xuất "không hợp lệ".

2.4.5. Đánh giá ECC

q **Ưu điểm các hệ thống dựa trên ECC**

§ Được đánh giá là an toàn hơn các hệ thống mã hóa khóa công khác dựa trên DLP/IFP [2].
§ Thực thi nhanh
§ Băng thông nhỏ
§ Các thông số ngắn hơn
§ Tạo cặp khóa đơn giản.
§ Các hệ thống ECC có thể chống lại các tấn công trên các hệ thống mã hóa dựa trên DLP/IFP

q **Khuyết điểm các hệ thống dựa trên ECC**

§ Có nhiều tùy chọn cài đặt.
§ Vấn đề toán học được đánh giá là mới nên nghiên cứu chưa hoàn tất

2.5. So sánh RSA và ECC

Để đánh giá độ an toàn của một hệ thống mã hóa thực tế đã chứng minh là không nhất thiết chúng ta phải tìm cách giải được bài toán mà hệ thống được thiết kế dựa trên đó, chúng ta chủ yếu đánh giá trong trường hợp tổng quát do các tham số hệ thống được chọn lựa không rơi vào các trường hợp đặc biệt dễ bị tấn công. Do đó để đánh giá một hệ thống chúng ta có thể tập trung trên các thuật giải tốt nhất được biết để đánh giá độ an toàn.

q **Kích thước khóa**

Đối với RSA, tuy ECC có kích thước khóa nhỏ hơn nhưng ECC vẫn có cùng mức độ bảo mật với RSA.

Hướng dẫn của NIST về kích thước khóa công khai đối với chuẩn AES			
Kích thước khóa ECC (Bits)	Kích thước khóa RSA (Bits)	Tỷ lệ kích thước khóa	Kích thước khóa AES (Bits)
163	1024	1:06	-
256	3072	1:12	128
384	7680	1:20	192
512	15360	1:30	256

Bảng 2.1. Bảng so sánh về kích thước khóa công khai giữa ECC, RSA và AES [7]

- **Thời gian thực thi**

ECC có kích thước khóa ngắn hơn RSA dẫn đến việc thực hiện các phép tính cần thiết ít hơn, thời gian mã hóa nhanh hơn, và cần ít transistor cho việc cài đặt phần cứng hơn. Trong khi đó, RSA với việc thực hiện các phép tính trên các số nguyên dương cực lớn với kích thước khóa dài làm cho thời gian phát sinh khóa, mã hóa và giải mã lâu hơn rất nhiều so với ECC, chưa tính đến chi phí cho những phép tính phải sử dụng nhiều năng lượng, bộ xử lý, không gian lưu trữ do chiều dài bit của một số nguyên dương thường vượt qua khả năng chứa của thanh ghi nên phải dùng đến bộ đệm và bộ nhớ phụ để lưu trữ và chuyển đổi.

Ví dụ: ECC 115-bit sử dụng 11000 transistor trong khi RSA 512-bit khi thực thi dùng 50000 transistor và được xem là có độ bảo mật như nhau.

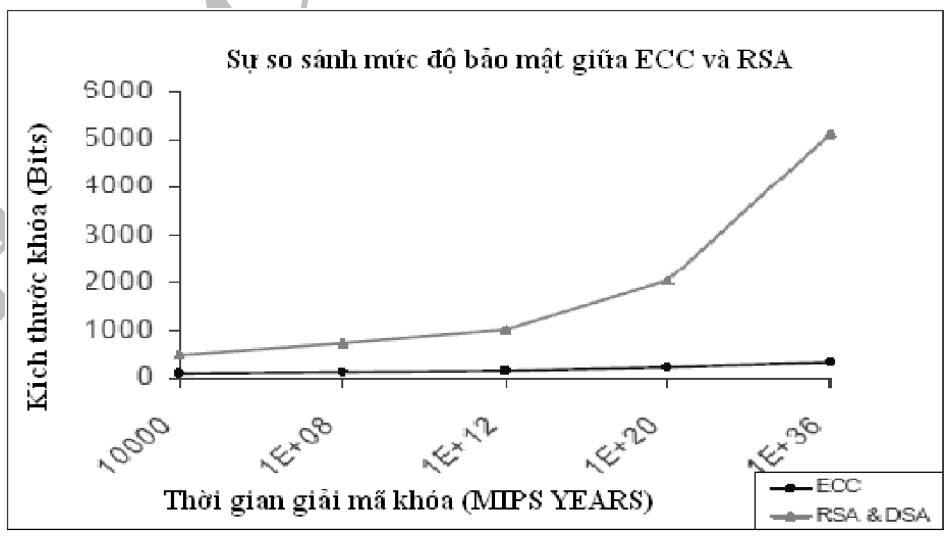
Ngoài ra, đặc biệt trong nhiều trường hợp của trường F_2^m cho phép thực hiện tính toán rất nhanh, trong trường hợp này ECC nhanh hơn cỡ 10 lần so với RSA hay DSA.

- **Tài nguyên sử dụng**

Các thiết bị ECC yêu cầu ít không gian lưu trữ, ít năng lượng, ít bộ nhớ, và yêu cầu băng thông ít hơn (do khóa nhỏ hơn tạo chữ ký ngắn hơn) những hệ thống khóa công khác.

- **Độ an toàn**

Cùng một mức độ bảo mật (cùng một thời gian giải mã khóa) nhưng kích thước khóa của RSA dài hơn ECC (Hình 3).



Hình 2.3. So sánh mức độ bảo mật giữa ECC và RSA

Cho đến năm 2002, thuật toán hiệu quả nhất để giải bài toán logarit rời rạc trên trường các số nguyên mod p bằng phương pháp sàng số có độ phức tạp mũ là:

$$O(\exp(c+o(1)))(\ln p)^{1/3} (\ln (\ln p))^{2/3}) \text{ với c là hằng số}$$

Tuy nhiên để giải bài toán logarit rời rạc trên đường cong Elip (ECDLP) đến năm 2002 vẫn chưa có thuật toán tổng quát độ phức tạp mũ con nào được biết. Thuật toán trong trường hợp tổng quát nhanh nhất hiện nay là thuật toán Pollard's có độ phức tạp tính số lượng phép toán trên nhóm là $O(0.88\sqrt{n})$, với n là ước số nguyên tố lớn nhất của bậc Elip. Nói chung, thuật toán tổng quát cho ECDLP là hàm mũ hoàn toàn.

Chương 3. Hàm băm

Do tốc độ của các hệ mã công khai rất chậm, vì thế, chữ ký điện tử thường không được ký trực tiếp trên toàn văn bản mà thông qua bản tóm tắt. Bản tóm tắt thường được xây dựng dựa trên một hàm băm. Chương này khảo sát các phương pháp băm và ứng dụng nó trong tạo chữ ký điện tử

Các hàm băm mật mã nhận chuỗi đầu vào có chiều dài bất kỳ (thường rất lớn) và ánh xạ chuỗi này thành chuỗi đầu ra có chiều dài cố định thường ngắn, kết xuất này gọi là giá trị băm (hash value), bản tóm tắt hay mã khóa thông điệp (message disgest). Thuật ngữ hàm băm (Hash Function) có nghĩa là nén một chuỗi có chiều dài bất kỳ thành chuỗi có chiều dài cố định. Đối với các ứng dụng bảo mật cần phân biệt giữa hàm băm có khoá (MAC – Message Authentication Code) và hàm băm không khoá (MDC - Manipulation Detection Codes).

Hàm băm được dùng trong chữ ký điện tử. Một đặc tính cơ bản của hàm băm là việc tạo mã khóa thông điệp rất dễ nhưng việc phá mã để chuyển ngược mã thông điệp thành bản rõ ban đầu rất khó nếu không muốn nói là không thể.

Lý do để sử dụng hàm băm có thể so sánh với việc mỗi người có một dấu tay duy nhất để phân biệt, người ta cũng muốn dùng hàm băm để tạo ra giá trị duy nhất cho mỗi thông điệp. Ví dụ, gởi thông điệp "mua 500 cổ phần ABC" đến người môi giới chứng khoán. Trước khi gởi, cho thông điệp qua một hàm băm để tạo ra một mã khóa thông điệp rồi gởi cả thông điệp và giá trị băm đến người môi giới chứng khoán. Dọc đường, một kẻ gian chặn thông điệp và thay đổi số cổ phiếu thành 5000. Khi người môi giới chứng khoán nhận thông điệp và chạy cùng hàm băm trên thông điệp. Do thông điệp đã bị thay đổi, giá trị băm mà người môi giới chứng khoán thu được không giống với giá trị băm được gởi kèm theo thông điệp, và người môi giới chứng khoán biết được thông điệp đã bị thay đổi.

Như đã được đề cập, các hàm băm được dùng để ký các văn bản bằng kỹ thuật số. Một khi mã khóa thông điệp được tạo ra, nó có thể được mã hóa chung với thông điệp ban đầu bằng cách dùng khóa công khai của người nhận. Bản mã sau cùng có thể được truyền đi một cách an toàn. Chỉ người nhận mới có thể giải mã thông điệp bằng khóa cá nhân của anh ta. Hàm băm còn có một đặc tính là chúng sinh ra những mã khóa thông điệp rất khó chuyển dạng ngược lại. Do đó, mã khóa thông điệp có thể truyền công khai hầu như không sợ ai sẽ dùng nó để chuyển ngược lại thành thông điệp ban đầu.

3.1. Tính chất của hàm băm

3.1.1. Hàm băm một chiều (OWHF - One-Way Hash Function)

Khái niệm hàm băm một chiều được giới thiệu bởi Diffie và Hellman. Còn khái niệm đầu tiên không hình thức được giới thiệu bởi Merkle.

***Định nghĩa 1*:** Một hàm băm một chiều là một hàm băm h thỏa các điều kiện sau:

1. Đối số X có thể có chiều dài bất kỳ và kết quả h(X) có chiều dài cố định n bits.
2. Hàm băm phải là một chiều có nghĩa là cho Y là ảnh của X thì không có khả năng tính toán để tìm một thông điệp X sao cho h(X) = Y (tính chống nghịch ảnh) và cho X và h(X) thì không thể tìm được thông điệp X' ≠ X để h(X') = h(X) (tính chống trùng ảnh).

Chiều dài n của kết quả băm thường từ 64 đến 128 bit. Một hàm có tính chất chống nghịch ảnh được biết như hàm một chiều (tính chất chống nghịch ảnh là đặc thù cho các hàm băm). Chống trùng ảnh còn được gọi là sự chống xung đột yếu. Với một số ứng dụng (ví dụ: các hàm phát sinh ngẫu nhiên và thuật toán MAC trên cơ sở các hàm băm), phần lớn đầu vào của hàm băm có thể biết trước, tuy nhiên người ta yêu cầu là phần không biết còn lại của đầu vào khó phục hồi sau khi băm. Đặc tính này gọi là chống nghịch ảnh thành phần (partial preimage resistance).

3.1.2. Hàm băm chống xung đột (CRHF - Collision Resistant Hash Function)

Tầm quan trọng của việc chống xung đột đối với các hàm băm được dùng trong các lược đồ chữ ký điện tử đầu tiên được đưa ra bởi Yuval. Định nghĩa đầu tiên của CRHF được đề xuất bởi Damgard. Định nghĩa không hình thức được đề xuất bởi Merkle.

***Định nghĩa 2**:* Một hàm băm chống xung đột (CRHF) là một hàm h thoả các điều kiện sau:

1. Đối số X có thể có chiều dài bất kỳ và kết quả h(X) có chiều dài cố định n bits.
2. Hàm băm phải là hàm một chiều.
3. Hàm băm chống xung đột có nghĩa là sẽ không khả thi nếu cần tìm hai thông điệp khác nhau có cùng kết quả băm.

3.1.3. Các hàm băm lặp (Iterated Hash Function)

Đa số các hàm băm hiện nay đều dựa trên nguyên tắc nén với đầu vào có chiều dài cố định. Chúng xử lý các khối thông điệp theo cách tương tự như nhau. Lai và Messay gọi đây là hàm băm lặp. Dữ liệu đầu vào được thêm các bit/byte/… để chiều dài đầu vào là bội số của chiều dài khối. Tiếp theo dữ liệu đầu vào được chia thành t khối từ X_1 đến X_t. Kết quả băm được tính như sau:

$H_0 = IV$
$H_i = f(X_i, H_{i-1})$ $\quad i = 1, 2, \ldots, t$
$h(X) = g(H_t)$

Trong đó:

- IV (Initial Value) : giá trị khởi đầu.
- H_i (chaining variable) : biến xích.
- f: hàm nén hay hàm vòng.
- g: hàm biến đổi đầu ra.

Hai yếu tố có tầm quan trọng ảnh hưởng đến tính bảo mật của hàm băm là: qui luật thêm giá trị (thêm các bit hay byte,…) và chọn IV. Luật thêm giá trị này không được nhập nhằng (tức là không tồn tại hai thông điệp sau khi thêm cho hai giá trị giống nhau); cuối cùng người ta thêm chiều dài thông điệp. Còn IV nên được định nghĩa như phần mô tả của hàm băm được gọi là sự tăng cường thông điệp rút gọn (MD-strengthening = Message Disgest – strengthening) theo Merkle và Damgard.

Định lý 1 (Định lý Lai-Massey): Giả sử rằng việc thêm giá trị bit/byte/… bao gồm cả chiều dài của chuỗi đầu vào và giả sử thêm rằng thông điệp X ban đầu (chưa thêm giá trị) chứa ít nhất 2 khối thì khi tìm một nghịch ảnh thứ hai đối với h với giá trị IV cố định 2^n thao tác nếu và chỉ nếu tìm một nghịch ảnh thứ hai đối với f với H_{i-1} được chọn bất kỳ yêu cầu 2^n thao tác.

Định lý 2 (Định lý Damgard-Merkle): Đặt f là hàm băm chống xung đột ánh xạ l thành n bits (với l – n > 1). Nếu một luật thêm không nhập nhằng được sử dụng thì hàm thiết lập sau đây sẽ cho ra một hàm băm chống xung đột.

$H_1 = f(0^{n+1} \| X_1)$
$H_i = f(H_{i-1} \| 1 \| X_i)$ $\quad i = 2, 3, \ldots, t.$

3.2. Giới thiệu một số hàm băm

3.2.1. Hàm MD5

3.2.1.1. Giới thiệu

MD5 do giáo sư Ron Rivest đề xuất năm 1991. Thuật toán lấy dữ liệu đầu vào có chiều dài bất kỳ và cho ra kết quả thông điệp rút gọn dài 128-bit. Thuật toán được chứng minh là không khả thi trong việc tìm ra 2 thông điệp có cùng thông điệp rút gọn hoặc không thể tìm thông điệp ban đầu từ thông điệp rút gọn cho trước. Thuật toán MD5 được áp dụng cho hầu hết các ứng dụng chữ ký điện tử. Ở đó một tập tin lớn phải được "nén" trong trạng thái an toàn trước khi được mã hoá bằng một khoá cá nhân trong hệ mã hoá khoá công khai như RSA chẳng hạn. Thuật toán MD5 được thiết kế chạy khá nhanh trên các máy 32-bit và không yêu cầu bất kỳ bảng thay thế (S-box) lớn nào. Thuật toán MD5 là phiên bản mở rộng của thuật toán MD4. MD5 chạy khá chậm hơn so với MD4, nhưng được thiết kế cẩn thận hơn và có tính bảo mật cao hơn. MD4 đang phải đối đầu với những tấn công giải mã.

3.2.1.2. Thuật toán

Cũng như MD4, MD5 thực hiện trên đơn vị khối là chuỗi bit. Đầu vào là thông điệp b-bit (b Z, b>=0, b không nhất thiết phải là bội số của 8, và có thể lớn tuỳ ý); đầu ra là thông điệp rút gọn. Các bit của thông điệp được viết lại như sau: m_0, m_1, …., m_{b-1}

Tính toán thông điệp rút gọn của thông điệp gồm 5 bước sau:

1. **Thêm bit**: Thông điệp được thêm các bit để chiều dài khi mod 512 bằng 448 và thêm 64 bit để lưu chiều dài thông điệp để cho kích thước sau cùng của thông điệp này là bội số của 512. Bước này luôn được thực hiện dù chiều dài có bằng 448 khi mod 512 hay không. Thêm ít nhất là 1 bit và thêm nhiều nhất là 512 bit. Thông điệp ban đầu được thêm 1 bit '1' duy nhất và sau đó là thêm các bit '0' để chiều dài sau cùng của thông điệp là 448 khi mod 512.

2. **Thêm chiều dài**: b là chiều dài của thông điệp trước khi thêm các bit vào (ở bước 1) được lưu trữ trong 64 bit, b không thể lớn hơn 2^{64} bit; 64 bit lưu trữ b được chia làm 2 từ 32 bit được thêm vào thông điệp theo dạng từ có thứ tự thấp được lưu trước.Thông điệp sau khi qua bước 1 và thêm b vào sẽ có chiều dài chính xác là bội số của 512 bit (16 từ 32 bit). Đặt M[0..N-1] ký hiệu cho các từ của thông điệp kết quả với N là bội số của 16-từ.

3. **Khởi tạo vùng đệm MD:** Vùng đệm gồm 4 từ (A, B, C, D) được dùng để tính giá trị băm (hay giá trị rút gọn của thông điệp). Mỗi từ A, B, C, D là một thanh ghi 32-bit, các thanh ghi này được khởi tạo bởi các giá trị sau theo hệ số thập lục phân, các byte thứ tự thấp lưu trước:

   word A: 01 23 45 67
   word B: 89 ab cd ef
   word C: fe dc ba 98
   word D: 76 54 32 10

4. **Xử lý các thông điệp theo các khối 16-từ:** Trước hết cần định nghĩa 4 hàm phụ trợ có đầu vào là 3 từ 32-bit và cho kết quả là 1 từ 32-bit.

   F(X,Y,Z) = XY OR NOT(X) Z
   G(X,Y,Z) = XZ OR Y NOT(Z)
   H(X,Y,Z) = X XOR Y XOR Z
   I (X,Y,Z) = Y XOR (X OR NOT(Z))

Bước này dùng một bảng gồm 64 phần tử T[0..63] được khởi tạo từ hàm sine, T[i] ký hiệu cho phần tử thứ i của bảng và bằng abs(sin(i)) * 4294967296, i có đơn vị là radian.

```
static DWORD T[64] = {
//vòng 1
        0xd76aa478, 0xe8c7b756, 0x242070db, 0xc1bdceee,
        0xf57c0faf, 0x4787c62a, 0xa8304613, 0xfd469501,
        0x698098d8, 0x8b44f7af, 0xffff5bb1, 0x895cd7be,
        0x6b901122, 0xfd987193, 0xa679438e, 0x49b40821,
//vòng 2
        0xf61e2562, 0xc040b340, 0x265e5a51, 0xe9b6c7aa,
        0xd62f105d, 0x2441453 , 0xd8a1e681, 0xe7d3fbc8,
        0x21e1cde6, 0xc33707d6, 0xf4d50d87, 0x455a14ed,
        0xa9e3e905, 0xfcefa3f8, 0x676f02d9, 0x8d2a4c8a,
//vòng 3
        0xfffa3942, 0x8771f681, 0x6d9d6122, 0xfde5380c,
        0xa4beea44, 0x4bdecfa9, 0xf6bb4b60, 0xbebfbc70,
        0x289b7ec6, 0xeaa127fa, 0xd4ef3085, 0x4881d05 ,
        0xd9d4d039, 0xe6db99e5, 0x1fa27cf8, 0xc4ac5665,

//vòng 4
        0xf4292244, 0x432aff97, 0xab9423a7, 0xfc93a039,
        0x655b59c3, 0x8f0ccc92, 0xffeff47d, 0x85845dd1,
        0x6fa87e4f, 0xfe2ce6e0, 0xa3014314, 0x4e0811a1,
        0xf7537e82, 0xbd3af235, 0x2ad7d2bb, 0xeb86d391
};
```

Bước này được thực hiện như sau :

```
/* Xử lý mỗi khối 16-từ */
  For i = 0 to N/16-1 do

    /* Copy khối i vào X. */
    For j = 0 to 15 do
      Set X[j] to M[i*16+j].
    end for on j

    /* Lưu lại nội dung các thanh ghi */
    AA = A
    BB = B
    CC = C
    DD = D

    /* Vòng 1. */
    /* Đặt [abcd k s i] ký hiệu cho
        a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s). */
[ABCD  0  7  1]  [DABC  1 12  2]  [CDAB  2 17  3]  [BCDA  3 22  4]
[ABCD  4  7  5]  [DABC  5 12  6]  [CDAB  6 17  7]  [BCDA  7 22  8]
[ABCD  8  7  9]  [DABC  9 12 10]  [CDAB 10 17 11]  [BCDA 11 22 12]
[ABCD 12  7 13]  [DABC 13 12 14]  [CDAB 14 17 15]  [BCDA 15 22 16]

    /* Vòng 2. */
    /* Đặt [abcd k s i] ký hiệu cho
        a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s). */
[ABCD  1  5 17]  [DABC  6  9 18]  [CDAB 11 14 19]  [BCDA  0 20 20]
[ABCD  5  5 21]  [DABC 10  9 22]  [CDAB 15 14 23]  [BCDA  4 20 24]
[ABCD  9  5 25]  [DABC 14  9 26]  [CDAB  3 14 27]  [BCDA  8 20 28]
[ABCD 13  5 29]  [DABC  2  9 30]  [CDAB  7 14 31]  [BCDA 12 20 32]

    /* Vòng 3. */
    /* Đặt [abcd k s t] ký hiệu cho
        a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s). */
[ABCD  5  4 33]  [DABC  8 11 34]  [CDAB 11 16 35]  [BCDA 14 23 36]
[ABCD  1  4 37]  [DABC  4 11 38]  [CDAB  7 16 39]  [BCDA 10 23 40]
[ABCD 13  4 41]  [DABC  0 11 42]  [CDAB  3 16 43]  [BCDA  6 23 44]
[ABCD  9  4 45]  [DABC 12 11 46]  [CDAB 15 16 47]  [BCDA  2 23 48]

    /* Vòng 4. */
    /* Đặt [abcd k s t] ký hiệu cho
        a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s). */
[ABCD  0  6 49]  [DABC  7 10 50]  [CDAB 14 15 51]  [BCDA  5 21 52]
[ABCD 12  6 53]  [DABC  3 10 54]  [CDAB 10 15 55]  [BCDA  1 21 56]
[ABCD  8  6 57]  [DABC 15 10 58]  [CDAB  6 15 59]  [BCDA 13 21 60]
```

[ABCD 4 6 61] [DABC 11 10 62] [CDAB 2 15 63] [BCDA 9 21 64]

hay các vòng được thực hiện cụ thể như sau:

```
/* Vòng 1 */
 FF (a, b, c, d, x[ 0], S11, 0xd76aa478); /* 1 */
 FF (d, a, b, c, x[ 1], S12, 0xe8c7b756); /* 2 */
 FF (c, d, a, b, x[ 2], S13, 0x242070db); /* 3 */
 FF (b, c, d, a, x[ 3], S14, 0xc1bdceee); /* 4 */
 FF (a, b, c, d, x[ 4], S11, 0xf57c0faf); /* 5 */
 FF (d, a, b, c, x[ 5], S12, 0x4787c62a); /* 6 */
 FF (c, d, a, b, x[ 6], S13, 0xa8304613); /* 7 */
 FF (b, c, d, a, x[ 7], S14, 0xfd469501); /* 8 */
 FF (a, b, c, d, x[ 8], S11, 0x698098d8); /* 9 */
 FF (d, a, b, c, x[ 9], S12, 0x8b44f7af); /* 10 */
 FF (c, d, a, b, x[10], S13, 0xffff5bb1); /* 11 */
 FF (b, c, d, a, x[11], S14, 0x895cd7be); /* 12 */
 FF (a, b, c, d, x[12], S11, 0x6b901122); /* 13 */
 FF (d, a, b, c, x[13], S12, 0xfd987193); /* 14 */
 FF (c, d, a, b, x[14], S13, 0xa679438e); /* 15 */
 FF (b, c, d, a, x[15], S14, 0x49b40821); /* 16 */

/* Vòng 2 */
 GG (a, b, c, d, x[ 1], S21, 0xf61e2562); /* 17 */
 GG (d, a, b, c, x[ 6], S22, 0xc040b340); /* 18 */
 GG (c, d, a, b, x[11], S23, 0x265e5a51); /* 19 */
 GG (b, c, d, a, x[ 0], S24, 0xe9b6c7aa); /* 20 */
 GG (a, b, c, d, x[ 5], S21, 0xd62f105d); /* 21 */
 GG (d, a, b, c, x[10], S22,  0x2441453); /* 22 */
 GG (c, d, a, b, x[15], S23, 0xd8a1e681); /* 23 */
 GG (b, c, d, a, x[ 4], S24, 0xe7d3fbc8); /* 24 */
 GG (a, b, c, d, x[ 9], S21, 0x21e1cde6); /* 25 */
 GG (d, a, b, c, x[14], S22, 0xc33707d6); /* 26 */
 GG (c, d, a, b, x[ 3], S23, 0xf4d50d87); /* 27 */
 GG (b, c, d, a, x[ 8], S24, 0x455a14ed); /* 28 */
 GG (a, b, c, d, x[13], S21, 0xa9e3e905); /* 29 */
 GG (d, a, b, c, x[ 2], S22, 0xfcefa3f8); /* 30 */
 GG (c, d, a, b, x[ 7], S23, 0x676f02d9); /* 31 */
 GG (b, c, d, a, x[12], S24, 0x8d2a4c8a); /* 32 */

 /* Vòng 3 */
 HH (a, b, c, d, x[ 5], S31, 0xfffa3942); /* 33 */
 HH (d, a, b, c, x[ 8], S32, 0x8771f681); /* 34 */
 HH (c, d, a, b, x[11], S33, 0x6d9d6122); /* 35 */
 HH (b, c, d, a, x[14], S34, 0xfde5380c); /* 36 */
 HH (a, b, c, d, x[ 1], S31, 0xa4beea44); /* 37 */
 HH (d, a, b, c, x[ 4], S32, 0x4bdecfa9); /* 38 */
```

HH (c, d, a, b, x[7], S33, 0xf6bb4b60); /* 39 */
HH (b, c, d, a, x[10], S34, 0xbebfbc70); /* 40 */
HH (a, b, c, d, x[13], S31, 0x289b7ec6); /* 41 */
HH (d, a, b, c, x[0], S32, 0xeaa127fa); /* 42 */
HH (c, d, a, b, x[3], S33, 0xd4ef3085); /* 43 */
HH (b, c, d, a, x[6], S34, 0x4881d05); /* 44 */
HH (a, b, c, d, x[9], S31, 0xd9d4d039); /* 45 */
HH (d, a, b, c, x[12], S32, 0xe6db99e5); /* 46 */
HH (c, d, a, b, x[15], S33, 0x1fa27cf8); /* 47 */
HH (b, c, d, a, x[2], S34, 0xc4ac5665); /* 48 */

/* Vòng 4 */
II (a, b, c, d, x[0], S41, 0xf4292244); /* 49 */
II (d, a, b, c, x[7], S42, 0x432aff97); /* 50 */
II (c, d, a, b, x[14], S43, 0xab9423a7); /* 51 */
II (b, c, d, a, x[5], S44, 0xfc93a039); /* 52 */
II (a, b, c, d, x[12], S41, 0x655b59c3); /* 53 */
II (d, a, b, c, x[3], S42, 0x8f0ccc92); /* 54 */
II (c, d, a, b, x[10], S43, 0xffeff47d); /* 55 */
II (b, c, d, a, x[1], S44, 0x85845dd1); /* 56 */
II (a, b, c, d, x[8], S41, 0x6fa87e4f); /* 57 */
II (d, a, b, c, x[15], S42, 0xfe2ce6e0); /* 58 */
II (c, d, a, b, x[6], S43, 0xa3014314); /* 59 */
II (b, c, d, a, x[13], S44, 0x4e0811a1); /* 60 */
II (a, b, c, d, x[4], S41, 0xf7537e82); /* 61 */
II (d, a, b, c, x[11], S42, 0xbd3af235); /* 62 */
II (c, d, a, b, x[2], S43, 0x2ad7d2bb); /* 63 */
II (b, c, d, a, x[9], S44, 0xeb86d391); /* 64 */

A = A + AA
B = B + BB
C = C + CC
D = D + DD

//hết vòng lặp theo i

5. **Xuất kết quả:** Giá trị băm được lưu trong 4 thanh ghi A, B, C, D có nghĩa là chuỗi byte kết quả bắt đầu với với byte thấp của A và kết thúc với byte cao của D.

3.2.1.3. Phân biệt MD5 với MD4

MD5 khác MD4 ở những điểm sau:

1. MD5 có thêm vòng thứ tư nhằm tăng độ an toàn.

2. Mỗi bước có một hằng số phụ T[i] phân biệt còn MD4 sử dụng 3 hằng số 0, 0x5A827999, 0x6ED9EBA1 cho từng chu kỳ biến đổi.
3. Hàm G trong vòng 2 thay đổi từ (XY OR XZ OR YZ) thành (XZ OR Y NOT(Z)) nhằm làm cho g ít tính đối xứng hơn.
4. Kết quả mỗi bước cộng với kết quả của bước trước cho thấy kết quả của mỗi bước phụ thuộc vào bước trước đó. Điều này giúp thuật toán lan truyền nhanh hơn.
5. Các hằng số xoay vòng trong mỗi vòng vừa được tối ưu, để đạt được một hiệu ứng lan truyền nhanh hơn. Các hằng số xoay vòng trong các vòng khác nhau thì phân biệt nhau.

3.2.2. SHA-1

3.2.2.1. Giới thiệu

SHA_1 dựa trên các nguyên lý tương tự với những nguyên lý mà giáo sư Ronald L.Rivest của MIT khi thiết kế thuật toán băm MD4, SHA_1 được đề xuất vào tháng 4 năm 1995.

Khi nhập vào một thông điệp có chiều dài bất kỳ nhỏ hơn 2^{64} bit, SHA_1 cho ra kết quả là một thông điệp rút gọn (hay giá trị băm) dài 160 bits.

SHA_1 được gọi là an toàn vì không thể tìm ra thông điệp liên quan đến thông điệp rút gọn hay tìm ra hai thông điệp khác nhau nhưng có cùng thông điệp rút gọn. Bất kỳ thay đổi nào của thông điệp, với xác suất cao, kết quả vẫn cho ra các thông điệp rút gọn khác nhau.

3.2.2.2. Các hàm và các hằng số được dùng trong thuật toán

Một chuỗi các hàm logic f(0), f(1),…, f(79) được dùng trong thuật toán SHA_1. Mỗi hàm f(t), $0 \le t \le 79$, hoạt động trên các từ 32-bit B, C, D và cho ra một từ 32-bit. Hàm f(t, B, C, D) được định nghĩa như sau:

Cho các từ B, C, D.

f(t;B,C,D) = (B AND C) OR ((NOT B) AND D)	($0 \le t \le 19$).
f(t;B,C,D) = B XOR C XOR D	($20 \le t \le 39$).
f(t;B,C,D) = (B AND C) OR (B AND D) OR (C AND D)	($40 \le t \le 59$).
f(t;B,C,D) = B XOR C XOR D	($60 \le t \le 79$).

Các từ hằng số K(0), K(1),…K(79) được dùng trong SHA_1. Trong hệ thập lục phân, những hằng này được cho bởi các giá trị như sau:

K(t) = 5A827999	($0 \le t \le 19$).
K(t) = 6ED9EBA1	($20 \le t \le 39$).
K(t) = 8F1BBCDC	($40 \le t \le 59$).
K(t) = CA62C1D6	($60 \le t \le 79$).

3.2.2.3. Tính giá trị băm

Có hai phương pháp được đề xuất trong việc tính giá trị băm của thuật toán SHA_1 và cả hai cho ra cùng giá trị băm. Mặc dù thuật toán thứ hai lưu trữ 64 từ 32-bit, nó có thể kéo dài thời gian hoạt động do việc tăng tính phức tạp trong tính toán địa chỉ cho {W[t]} trong bước c. Ngoài ra còn có những phương pháp tính toán khác cũng cho các kết quả giống nhau.

- **Phương pháp 1** (thêm bit vào thông điệp)

SHA_1 được dùng để tính thông điệp rút gọn cho đầu vào là một thông điệp hay một tập tin dữ liệu có chiều dài $l < 2^{64}$. Thông điệp hay tập tin dữ liệu được xem như một chuỗi bit. Nếu số bit của thông điệp là bội số của 8 (để gọn hơn biểu diễn thông điệp ở hệ lục phân). Mục đích của việc thêm các bit vào thông điệp để làm cho chiều dài tổng cộng của thông điệp là bội số của 512. SHA_1 liên tục xử lý các khối 512 bit để tính thông điệp rút gọn.

Việc thêm bit được thực hiện như sau: một bit ‘1’ được thêm vào tiếp theo là một chuỗi bit ‘0’ và một giá trị nguyên 64 bit chứa chiều dài của thông điệp ban đầu được thêm vào thông điệp sao cho chiều dài sau cùng của thông điệp là bội số của 512 bit. Chiều dài thông điệp được lưu trong 2 từ 32-bit, nếu chiều dài $l < 2^{32}$ thì từ đầu tiên chứa các bit ‘0’. Thông điệp sau khi thêm được xử lý như những khối 512-bit. Thông điệp sau khi thêm chứa 16*n từ 32-bit với n>0, thông điệp gồm n khối 16-từ M(0), M(1),…M(n).

Việc tính toán dùng 2 vùng đệm , mỗi vùng đệm chứa 5 từ 32 bit và một chuỗi 80 từ 32-bit. Các từ của vùng đệm 5 từ đầu ký hiệu là A, B, C, D, E. Các từ của vùng đệm trong 5 từ thứ hai ký hiệu là H0, H1, H2, H3, H4. Các từ của chuỗi 80-từ được ký hiệu là W(0), W(1), W(2),….,W(79). Một vùng đệm TEMP chứa duy nhất 1 từ được dùng.

Để tạo ra thông điệp rút gọn , các khối 16-từ M(1),M(2),…,M(n) được xử lý theo thứ tự. Việc xử lý mỗi M(i) bao gồm 80 bước. Trước khi xử lý bất kỳ khối nào , H_i được khởi động theo hệ thệp lục phân như sau:

H0 = 67452301
H1 = EFCDAB89
H2 = 98BADCFE
H3 = 10325476
H4 = C3D2E1F0.

Bây giờ M(1), M(2), …, M(n) được xử lý như sau:

1. Chia M(i) thành 16 từ W(0), W(1), …, W(15), trong đó W(0) là từ bên trái nhất.
2. For t = 16 to 79

W(t) = S^1(W(t-3) XOR W(t-8) XOR W(t-14) XOR W(t-16)).

3. Đặt A = H0, B = H1, C = H2, D = H3, E = H4.
4. For t = 0 to 79 do
   TEMP = S^5(A) + f(t;B,C,D) + E + W(t) + K(t);
   E = D; D = C; C = S^30(B); B = A; A = TEMP;
5. H0 = H0 + A,
   H1 = H1 + B,
   H2 = H2 + C,
   H3 = H3 + D,
   H4 = H4 + E.

Sau khi xử lý thông điệp là một chuỗi dài 160-bit được biểu diễn bằng 5 từ: H0 H1 H2 H3 H4.

- **Phương pháp 2**

Phương pháp ở trên giả sử chuỗi W(0),…, W(79) được lưu như một mảng 80 từ 32-bit. Điều này thì có hiệu quả theo quan điểm tối ưu thời gian thực thi bởi vì các địa chỉ của W(t-3), … , W(t-16) ở bước b dễ dàng tính được. Nếu không gian lưu trữ được chú trọng thì {W(t)} được biểu diễn như một hàng đợi xoay vòng. Hàng đợi này bao gồm một mảng 160 từ 32-bit W(0), … , W(15). Trong trường hợp này, theo hệ thập lục phân ta đặt MASK = 0000000F.

Kế đó xử lý M(i) như sau:

1. Chia M(i) thành 16-từ W[0], … , W[15] với W[0] là từ bên trái nhất.
2. Đặt A = H0, B = H1, C = H2, D = H3, E = H4.
3. For t = 0 to 79 do
   s = t AND MASK;
   if (t >= 16) W[s] = S^1(W[(s + 13) AND MASK] XOR W[(s +8) AND MASK] XOR W[(s + 2) AND MASK] XOR W[s]);
   TEMP = S^5(A) + f(t;B,C,D) + E + W[s] + K(t);
   E = D; D = C; C = S^30(B); B = A; A = TEMP;
4. Đặt
   H0 = H0 + A,
   H1 = H1 + B,
   H2 = H2 + C,
   H3 = H3 + D,
   H4 = H4 + E.

3.2.3. Tiger

3.2.3.1. Giới thiệu

Các hàm băm được đề cập ở trên được thiết kế trên các bộ vi xử lý 32-bit, thế hệ sắp tới của các bộ vi xử lý có dùng các từ 64-bit (word), và bao gồm

luôn cả một loạt các máy DEC Alpha cũng như các bộ vi xử lý thế hệ kế tiếp từ Intel, HP, và IBM. Dường như sẽ hợp lý nếu giả sử việc dùng các vi xử lý trong các ứng dụng nhúng (đa số các hệ thống sẽ dùng các bộ vi xử lý 64-bit trong vòng 5 năm tới). Tuy nhiên, trên các vi xử lý, các họ hàm băm trên không thể được cài đặt một cách hiệu quả.

Ví dụ: họ MD dùng các phép toán cộng và xoay vòng, vì vậy một thanh ghi 64-bit chỉ có thể xử lý một giá trị 32-bit tại một thời điểm, việc này làm giảm đi tốc độ khoảng 2 lần. Hơn nữa cấu trúc Alpha không có hỗ trợ bất kỳ phép xoay vòng nào trên 64-bit hay 32-bit.

Tiger, ra đời năm 1996 bởi Ross Anderson và Eli Biham, giải quyết các vấn đề nêu trên, nó nhanh như SHA-1 trên bộ vi xử lý 32-bit và nhanh hơn khoảng 3 lần trên vi xử lý 64-bit (DEC Alpha), và được cho là nhanh hơn SHA-1 trên các bộ vi xử lý 16-bit vì SHA-1 được thiết kế để chạy trên các máy 32-bit, trong khi yêu cầu là hàm băm thực hiện trên nhiều kích thước khác nhau của từ. Hàm băm Tiger có tính một chiều, giải quyết vấn đề xung đột, giải quyết vấn đề bội số (chống lại các kiểu tấn công dùng bội số hay các hệ số của các giá trị băm để tìm ra được khoá dùng trong việc băm thông điệp).

Hành động chính của hàm băm là tra cứu bảng trong 4 S-box , mỗi hoạt động đầu vào 8-bit và đầu ra 64-bit. Trên các máy 32-bit có thể cài đặt hai lần tra cứu bảng, với khoảng cách được tính chỉ một lần. Những phép tính khác như cộng hay trừ 64-bit, nhân 64-bit với hằng số nhỏ (5, 7, và 9), các phép tính shift và logic 64-bit như XOR và NOT. Tất cả các phép toán này chậm hơn gần hai lần trên các máy 32-bit; ngoài ra phép shift và nhân với hằng số nhỏ, chậm hơn 4 hay 5 lần (các bộ vi xử lý Alpha có những lời hướng dẫn đặc biệt cho phép nhân với các hằng số là nên có dạng 4±1 hay 8±1).

Vì khả năng tương thích, chúng ta chấp nhận cấu trúc bên ngoài của họ MD4: thông điệp được thêm một bit ‘1’ duy nhất được theo sau bởi chuỗi bit ‘0’ và cuối cùng là chiều dài thông điệp lưu trữ trong 64-bit tiếp theo. Kết quả được chia thành n khối 512-bit.

Kích thước của giá trị băm hay của trạng thái trung gian là 3 từ (192 bit), giá trị này được chọn vì :

- Dùng các từ 64-bit, kích thước là bội số của 64.
- Do tương thích với ứng dụng dùng SHA-1, kích thước băm nên ít nhất là 160 bit tăng tính bảo mật.
- Các biện pháp tấn công các hàm băm trên các giá trị băm trung gian hơn là giá trị băm cuối cùng. Điển hình, những kẻ tấn công chọn ra hai giá trị xung đột nhau trong một khối trung gian và hai giá trị này cứ được nhân lên thành sự xung đột toàn bộ hàm. Tuy nhiên, các tấn công này sẽ không hiệu quả nếu các giá trị băm trung gian lớn hơn.

Tiger với toàn bộ 192 bit ở đầu ra được ký hiệu là Tiger/192. Có hai loại hàm Tiger ngắn hơn dùng thay thế cho các hàm băm khác trong các ứng dụng hiện tại là:

- Tiger/160: giá trị băm là 160 bit đầu trong kết quả của Tiger/192 và được dùng vì tính tương thích với SHA và SHA-1.
- Tiger/128: Giá trị băm là 128 bit trong kết quả của Tiger/192, và được dùng cho tương thích với MD4, MD5, RIPE_MD, các hàm băm thuộc họ Snefru và các hàm băm khác.

Tính hiệu quả của hàm một phần dựa trên tính song song trong thiết kế của nó. Trong họ MD và Snefru, mỗi phép tính phụ thuộc trên kết quả của phép tính trước đó; vì vậy, các bộ vi xử lý không thể dùng một cách hiệu quả do cơ chế đường ống. Trong mỗi vòng của Tiger, 8 hành động tra cứu bảng được thực hiện song song, vì vậy trình biên dịch có thể tận dụng cơ chế đường ống tốt nhất. Thiết kế cũng cho phép việc cài đặt phần cứng hiệu quả. Tiger yêu cầu kích thước vùng nhớ ít hơn kích thước của 4 S-box, nếu vùng nhớ được yêu cầu này được định vị trong cache của vi xử lý thì việc tính toán sẽ chạy nhanh khoảng 2 lần (được đo trên máy DEC Alpha). Kích thước của 4 S-box là 4*256*8 = 8096 Bytes = 8Kbytes tương đương với kích thước của cache trên đa số các máy. Nếu 8 S-box được dùng, 16 Kbytes sẽ được yêu cầu gấp hai lần kích thước cache trên Alpha.

3.2.3.2. Đặc tả

Trên Tiger tất cả các phép tính được thực hiện trên các từ 64-bit theo dạng little endian hay bù 2. Dùng 3 thanh ghi 64-bit được gọi là a,b và c để chứa các giá trị băm trung gian. Các thanh ghi này được khởi tạo với giá trị h0 như sau:

A = 0x0123456789ABCDEF
B = 0xFEDCBA9876543210
C = 0xF096A5B4C3B2E187

Mỗi khối thông điệp 512-bit liên tiếp được chia thành 8 từ 64-bit x0, x1,…., x7, và việc tính toán sau thực hiện việc cập nhật h_i thành h_{i+1}.

Việc tính toán này bao gồm 3 lần thông qua hàm *pass*, và cứ giữa hai hàm *pass* là hàm *key schedule*, hàm này có tác dụng biến đổi dạng dữ liệu đầu vào ngăn cản kẻ tấn công lợi dụng tính thưa của dữ liệu trong 3 vòng lặp qua hàm *pass* trên. Cuối cùng hàm *feedforward* được dùng, ở đó giá trị của a,b,c được kết hợp với các giá trị khởi đầu của nó để cho giá trị băm h_{i+1}:

```
save_abc
pass(a, b, c, 5)
key_schedule
pass(c, a, b, 7)
key_schedule
```

pass(b, c, a, 9)
feedforward

Trong đó:

- **save_abc** lưu lại giá trị h_i

  aa = a
  bb = b
  cc = c
  dd = d

- **pass(a, b, c, mul)** là:

  round(a, b, c, x0, mul);
  round(b, c, a, x1, mul);
  round(c, a, b, x2, mul);
  round(a, b, c, x3, mul);
  round(b, c, a, x4, mul);
  round(c, a, b, x5, mul);
  round(a, b, c, x6, mul);
  round(b, c, a, x7, mul);

trong đó:

c ^= x;
a -= t1[c_0] ^ t2[c_2] ^ t3[c_4] ^ t4[c_6] ;
b += t4[c_1] ^ t3[c_3] ^ t2[c_5] ^ t1[c_7] ;
b *= mul;

Ở đây c_i là byte thứ i của c ($0 \le i \le 7$) và t1 đến t4 là 4 bảng S_box được trình bày trong tập tin "sboxes.h" đính kèm.

- **key_schedule**

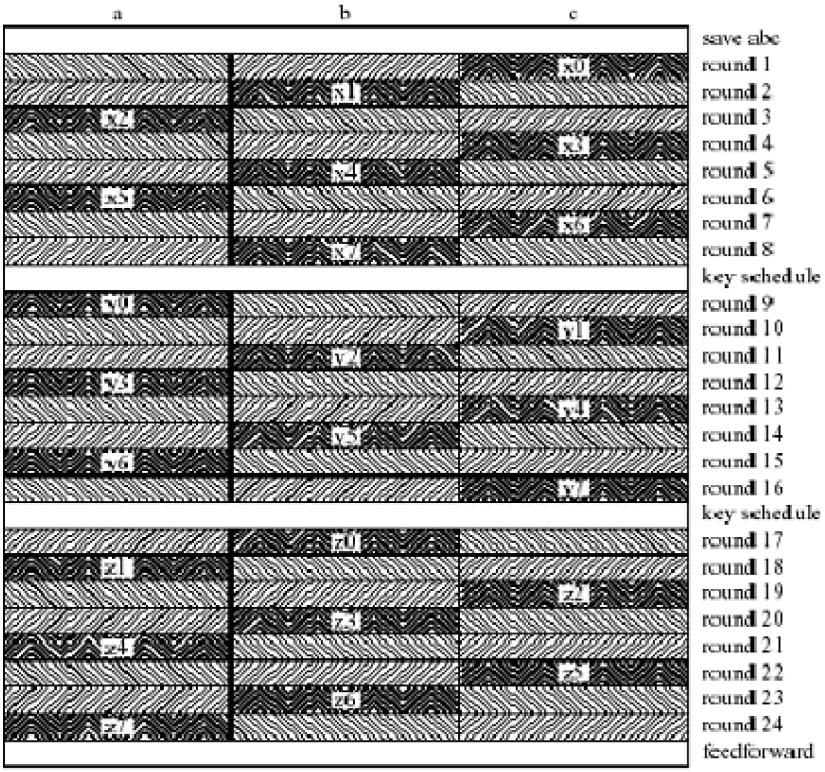
```
x0 -= x7 ^ 0xA5A5A5A5A5A5A5A5;
x1 ^= x0;
x2 += x1;
x3 -= x2 ^ ((~x1)<<19);
x4 ^= x3;
x5 += x4;
x6 -= x5 ^ ((~x4)>>23);
x7 ^= x6;
x0 += x7;
x1 -= x0 ^ ((~x7)<<19);
x2 ^= x1;
x3 += x2;
x4 -= x3 ^ ((~x2)>>23);
x5 ^= x4;
x6 += x5;
```

x7 -=x6 ^ 0x0123456789ABCDEF;

- **feedforward**

  a ^= aa;
  b – = bb;
  c += cc;

Các thanh ghi a,b,c chứa các giá trị băm (trung gian) h_{i+1} dài 192 bit



Hình 3.1. Phát thảo chức năng nén của Tiger

Vùng màu đen là các thanh ghi bị ảnh hưởng mà các đường sọc trong các thanh ghi này chỉ đến các byte trong vùng màu trắng. Các biến y0,…,y7 và các biến z0,…z7 là ký hiệu của x0,…,x7 tương ứng trong hàm pass thứ 2 và thứ 3. Cuối cùng, giá trị trung gian h_n được xem như đầu ra Tiger/192.

3.2.3.3. Tính bảo mật

Tính chất phi tuyến từ các S_box từ 8 bit thành 64 bit (đầu vào 8 bit, đầu ra 64 bit). Điều này tốt hơn nhiều so với việc chỉ kết hợp phép tính cộng với phép XOR (dùng các bit mang theo _ carry bit), và nó ảnh hưởng đến các bit xuất, không chỉ các bit lân cận.

Có một sự lan truyền mạnh , trong mỗi bit thông điệp ảnh hưởng đến tất cả 3 thanh ghi sau 3 vòng _ nhanh hơn những hàm băm khác. Việc lan truyền trong các từ 64 bit (và các S_box 64 bit) thì nhanh hơn nhiều khi các từ ngắn hơn được dùng.

Như ghi chú ở trên, tất cả các biện pháp tấn công đều tấn công trên kết quả của MDx hay Snefru ở một trong các khối trung gian. Tăng giá trị của kết quả trung gian lên 192 bit giúp ngăn chặn các cuộc tấn công này.
Các bảng khoá bảo đảm rằng việc thay đổi một số lượng nhỏ bit trong thông điệp ảnh hưởng đến nhiều bit trong suốt quá trình thông qua hàm pass. Cùng với việc lan truyền mạnh, nó giúp cho Tiger chống lại các tấn công tương đương với các tấn công khác nhau của Dobbertin trên MD4 (đó việc thay đổi một số bit nào đó trong thông điệp ảnh hưởng đến đa số các bit trong nhiều vòng, và kế đó sự khác biệt nhỏ nào có thể được thực hiện để hủy bỏ trong hàm pass trước đó).

Phép nhân của thanh ghi b trong mỗi vòng cũng góp phần cho việc chống lại các tấn công như vậy, do nó bảo đảm rằng các bit được dùng như đầu vào đến các S_box trong các vòng trước được trộn với những S_box khác, và với cùng S_box với một đầu vào khác. Phép nhân này cũng chống lại các tấn công trên các hàm băm, bởi vì các hằng số là khác nhau ở mỗi vòng.

Hàm *feedforward* ngăn cản các tấn công "meet-in-the-middle", "birthday" tìm ra các ảnh trước đó của hàm băm (mặc dù sự phức tạp sẽ là 2^{96}).

3.3. Hàm băm Whirlpool

3.3.1. Giới thiệu

Hàm băm Whirlpool được công nhận cùng với phương pháp mã hoá AES là những nền tảng bảo mật mạnh mẽ tại Hội thảo về Bảo Mật NESSIE _ New European Schemes for Signatures, Integrity, and Encryption (các kế hoạch Châu Âu mới cho chữ ký, tính toàn vẹn, và mã hoá) tại Lund, Thụy Điển vào ngày 26/2/2003. Mục đích của dự án là nêu ra những thuật toán bảo mật mới cho thị trường Châu Âu. Vincent Rijmen, trưởng nhóm bảo mật ở văn phòng Cryptomathic.s Belgian ở Leuven, là một trong hai tác giả của thuật toán AES và hàm băm Whirlpool. Nếu AES được chọn vào danh sách mã hoá theo khối 128-bit thì Whirlpool được chọn trong danh sách các hàm băm chống xung đột. Whirlpool được xếp vào chuẩn ISO ISO/IEC 10118-3 cho các hàm băm.

Whirlpool là hàm băm 512-bit được thiết kế dựa trên nguyên lý hoạt động của AES và có thể là một lựa chọn cho thuật toán SHA_1, một chiều, chống xung đột thực hiện trên thông điệp có chiều dài ít hơn 2^{256} bit do Paulo S.L.M Barreto và Vincent Rijmen đề xuất năm 2001.

Whirlpool bao gồm việc áp dụng hàm nén nhiều lần, trên nền tảng mã hoá toàn bộ khối 512 bit thông điệp chạy bên dưới, dùng khoá 512 bit. Hàm *round* và *key schedule* được thiết kế theo chiến lược Wide Trail. Whirlpool thực thi trên bộ vi xử lý 8-bit và 64-bit thuận lợi đặc biệt do cấu trúc hàm; tuy nhiên, lại không hướng tới bất kỳ nền phần cứng cụ thể nào.

3.3.2. Các cơ sở và ký hiệu toán học

3.3.2.1. Trường Galois (sự biểu diễn nhị phân)

Ký hiệu trường Galois $GF(2^4)$ là $GF(2)[x] / p_4(x)$ với $p_4(x) = x^4 + x + 1$ và trường $GF(2^8)$ như $GF(2)[x] / p_8x$ với $p_8(x) = x^8 + x^4 + x^3 + x^2 + 1$.

Đa thức $p_4(x)$ và $p_8(x)$ là các đa thức chính đầu tiên ở bậc 4 và 8, và được chọn sao cho $g(x) = x$ là phần tử sinh của $GF(2^4) / \{0\}$ và $GF(2^8) / \{0\}$ tương ứng. Các phần tử thuộc trường Galois được biểu diễn dưới dạng một đa thức $u = \sum_{i=o}^{m-1} u_i.x^i \in GF(2)[x]$, trong đó $u_i \in GF(2)$ với mọi $i = 0,\ldots, m-1$ sẽ được ghi chú giá trị số $\sum_{i=o}^{m-1} u_i.2^i$ hay được viết dưới dạng thập lục phân, ví dụ 13_x để ký hiệu cho $p_4(x)$.

3.3.2.2. Các lớp ma trận

M_{mxn} [$GF(2^8)$] ký hiệu cho tập các ma trận mxn dựa trên trường Galois $GF(2^8)$. $Cir(a_0, a_1, \ldots, a_{m-1})$ ký hiệu cho ma trận xoay vòng mxm mà dòng đầu tiên bao gồm $a_0, a_1, \ldots, a_{m-1}$, tức là :

$$Cir(a_0, a_1, \ldots, a_{m-1}) \equiv \begin{bmatrix} a_0, a_1, \ldots, a_{m-1} \\ a_{m-1}, a_0, \ldots, a_{m-2} \\ \ldots\ldots\ldots\ldots \\ a_1, a_2, \ldots, a_0 \end{bmatrix}$$

Hay đơn giản $Cir(a_0, a_1, \ldots, a_{m-1}) = c \Leftrightarrow c_{ij} = a_{(j-i) \bmod m}$, $0 \le i,j \le m-1$.

3.3.2.3. Mã MDS (MDS code - Maximal Distance Separable code)

Khoảng cách Hamming: giữa hai vectơ u và v từ không gian vectơ n chiều GF(2p)n bằng số tọa độ mà u và v khác nhau.

Trọng lượng Hamming wh(a): của phần tử a Є GF(2p)n là khoảng cách Hamming giữa a và vectơ không trong trường GF(2p)n tức là số thành phần khác không của a.

Mã tuyến tính [n,k,d]: trên trường GF(2p)n là không gian con k chiều của không gian vectơ (GF(2p))n trong đó khoảng cách Hamming giữa 2 vectơ không gian con phân biệt bất kỳ ít nhất là d (và d là số lớn nhất với đặc tính này).

Ma trận sinh G: cho mã [n,k,d] tuyến tính C là ma trận kxn mà các dòng của ma trận này hình thành nên nền tảng cho C. Một ma trận phát sinh ở dạng chuẩn hay phân cấp bậc, nếu nó có dạng G = [I_{kxk} $A_{kx(n-k)}$] trong đó I_{kxk} là ma trận định vị bậc k. Viết đơn giản là G = [IA] bỏ đi các chỉ mục khi các chiều của ma trận không liên quan đến thảo luận hay đã rõ trong ngữ cảnh đã cho.

Mã tuyến tính [n,k,d] tuân theo ràng buộc một đầu là d ≤. n-k+1. Mã gặp giới hạn này tức là d = n-k+1 gọi là mã có thể phân rã khoảng cách lớn nhất. Mã [n,k,d] tuyến tính C với ma trận phát sinh G = [I_{kxk} $A_{kx(n-k)}$] là MDS khi và chỉ khi mọi ma trận con vuông hình thành từ các dòng và các cột của A thì không kỳ dị.

3.3.2.4. Các thuộc tính mật mã

Tích của m biến Bool phân biệt được gọi là tích bậc m của các biến. Mỗi hàm Bool f : GF(2^n) " GF(2) có thể được viết như một tổng trên trường GF(2) của các tích bậc m phân biệt của các đối số của nó, 0 ≤ m ≤ n, và được gọi là dạng chuẩn đại số của f.

Bậc dạng chuẩn đại số của f, kí hiệu là *v*(f) là bậc lớn nhất của các số hạng xuất hiện trong dạng chuẩn đại số của f. Một hàm Bool tuyến tính là hàm Bool có bậc bằng 1, tức là dạng chuẩn đại số của nó chỉ bao gồm các đối số đơn. Cho trước α Є GF(2^n), ký hiệu hàm Bool tuyến tính là l_α: GF(2^n) " GF(2) bao gồm tổng các bit đối số được chọn bởi các bit của α :

$$l_\alpha(x) \equiv \bigoplus_{i=0}^{n-1} \alpha_i \cdot x_i.$$

Một ánh xạ S: GF(2^n) " GF(2^n), x # S[x] được gọi là S-box. Một S-box cũng có thể được xem như là một ánh xạ S : GF(2^n) " GF(2^n), do đó được mô tả dưới dạng các hàm Bool thành phần của nó s_i : GF(2^n) " GF(2), 0 ≤ i ≤ n-1 tức là S[x] = (s_0(x),…,s_{n-1}(x)).

Bậc của một S-box S, ký hiệu là ν_S là bậc nhỏ nhất trên tất cả các phép hợp các thành phần của S:

$$\nu_S \equiv \min_{\alpha \in GF(2)^n} \{\nu(l_\alpha \circ S)\}.$$

Tham số δ của một S-box S được định nghĩa như sau:

$$\delta_S \equiv 2^{-n} \cdot \max_{a \neq 0,\, b} \#\{c \in \mathrm{GF}(2^n) | S[c \oplus a] \oplus S[c] = b\}.$$

Giá trị 2^n. δ được gọi là hệ số đồng dạng phân biệt của S.

Hệ số tương quan (*correlation*) *c(f,g)* giữa 2 hàm Bool f và g được định nghĩa như sau:

$$c(f,g) \equiv 2^{1-n} \cdot \#\{x | f(x) = g(x)\} - 1.$$

Cực trị (*the extreme value*) của hệ số tương quan giữa các hàm tuyến tính của các bit đầu vào và các hàm tuyến tính của các bit đầu ra được gọi là độ dốc của S (*the bias of S*).

Tham số λ của S-box S được định nghĩa là giá trị tuyệt đối cũa độ dốc:

$$\lambda_S \equiv \max_{(i,j) \neq (0,0)} |c(l_i, l_j \circ S)|.$$

Số nhánh B (*the branch number B*) của ánh xạ tuyến tính θ : $GF(2^n)^k$ " $GF(2^n)^m$

$$B(\theta) \equiv \min_{a \neq 0} \{\mathrm{w_h}(a) + \mathrm{w_h}(\theta(a))\}.$$

Cho mã tuyến tính a[k+m,k,d] trên trường $GF(2^p)$ với ma trận phát sinh (*generator matrix*) $G = [I_{kxk} M_{kxm}]$, ánh xạ tuyến tính θ : $GF(2^n)^k$ " $GF(2^n)^m$ được định nghĩa bởi θ(a) = a.M có nhánh B(θ) = d; nếu mã là MDS, một ánh xạ như vậy được gọi là ánh xạ lan truyền tối ưu (*optimal diffusion mapping*).

3.3.2.5. Ký hiệu khác

Cho một dãy các hàm f_m, f_{m+1},…, f_{n-1}, f_n , nếu m ≤ n, chúng ta dùng ký hiệu

$$\bigcirc_{r=m}^{n} f_r \equiv f_m \circ f_{m+1} \circ \cdots \circ f_{n-1} \circ f_n,$$

và

$$\bigcirc_{m}^{r=n} f_r \equiv f_n \circ f_{n-1} \circ \cdots \circ f_{m+1} \circ f_m$$

nếu m > n cả hai biểu thức thay thế cho ánh xạ đơn vị.

3.3.3. Mô tả Whirlpool

Whirlpool nguyên thủy chính là hàm băm Merkle dựa trên mã hoá toàn bộ khối (dedicated block cipher), *W*, hoạt động trên một trạng thái băm (*state hash*) 512 bit dùng một trạng thái khoá mắc xích (*chained key state*), cả hai đều

xuất phát từ dữ liệu nhập. Sau đây là các ánh xạ và các hằng số thành phần thiết lập nên Whirlpool và đặc tả hàm băm Whirpool.

3.3.3.1. Nhập và xuất

Trạng thái băm (state hash) bên trong được xem như là một ma trận $M_{8x8}[GF(2^8)]$. Do đó, các khối dữ liệu 512 bit (được mô tả bên ngoài như các mảng byte bằng cách nhóm các bit lần lượt thành những đoạn 8 bit) phải được ánh xạ đến hay từ định dạng ma trận. Việc này được thực hiện bởi hàm

$\mu : GF(2^8)^{64}$ " $M^{8x8}[GF(2^8)]$

và nghịch đảo của nó là : $\mu(a) = b \Leftrightarrow b_{ij} = a_{8i+j}$, $0 \le i,j \le 7$

3.3.3.2. Lớp phi tuyến γ

Hàm $\gamma : M_{8x8}[GF(2^8)]$ " $M_{8x8}[GF(2^8)]$ bao gồm việc áp dụng song song của S-box $S : GF(2^8)$ " $GF(2^8)$, x # S[x] cho tất cả các byte của từng đối số :

$$\gamma(a) = b \Leftrightarrow bij = S[a_{ij}],\ 0 \le i,j \le 7$$

3.3.3.3. Hoán vị theo chu kỳ π

Hàm hoán vị $\pi : M_{8x8}[GF(2^8)]$ " $M_{8x8}[GF(2^8)]$ xoay theo chu kỳ mỗi cột của đối số (argument) của nó một cách độc lập, để các phần tử thuộc cột j được xoay j vị trí trên cột j (xoay theo dòng):

$$\pi(a) = \pi \Leftrightarrow b_{ij} = a_{(i-j)\bmod 8,j},\ 0 \le i,j \le 7$$

Mục đích của π là phát tán các byte của mỗi dòng giữa các dòng với nhau.

3.3.3.4. Lớp lan truyền tuyến tính θ

Lớp lan truyền tuyến tính $\theta: M_{8x8}[GF(2^8)]$ " $M_{8x8}[GF(2^8)]$ là một ánh xạ tuyến tính dựa trên mã MDS [n,k,d] với n = 16, k = 8, d = 9 với ma trận sinh $G_C = [IC]$ trong đó $C = Cir(01_x, 01_x, 04_x, 01_x, 08_x, 05_x, 02_x, 09_x)$ tức là :

$$C = \begin{bmatrix} 01_x & 01_x & 04_x & 01_x & 08_x & 05_x & 02_x & 09_x \\ 09_x & 01_x & 01_x & 04_x & 01_x & 08_x & 05_x & 02_x \\ 02_x & 09_x & 01_x & 01_x & 04_x & 01_x & 08_x & 05_x \\ 05_x & 02_x & 09_x & 01_x & 01_x & 04_x & 01_x & 08_x \\ 08_x & 05_x & 02_x & 09_x & 01_x & 01_x & 04_x & 01_x \\ 01_x & 08_x & 05_x & 02_x & 09_x & 01_x & 01_x & 04_x \\ 04_x & 01_x & 08_x & 05_x & 02_x & 09_x & 01_x & 01_x \\ 01_x & 04_x & 01_x & 08_x & 05_x & 02_x & 09_x & 01_x \end{bmatrix},$$

để $\theta(a) = b \Leftrightarrow b = a \,.\, C$. Kết quả của θ là trộn các byte trong mỗi dòng trạng thái.

3.3.3.5. Phép cộng khoá σ[k]

Phép cộng khoá quan hệ σ[k] : $M_{8x8}[GF(2^8)]$ " $M_{8x8}[GF(2^8)]$ bao gồm phép XOR của ma trận khoá k Є $M_{8x8}[GF(2^8)]$:

$$\sigma[k](a) = b \Leftrightarrow b_{ij} = a_{ij} \oplus k_{ij},\ 0 \le i,j \le 7$$

Ánh xạ này cũng cho biết các hằng số vòng trong bảng xếp lịch khoá.

3.3.3.6. Hằng số vòng cr

Hằng số vòng cho vòng thứ r , r>0, là một ma trận c^r Є $M_{8x8}[GF(2^8)]$, được định nghĩa như sau:

$$c^r_{0j} \equiv S[8(r-1)+j],\ 0 \leqslant j \leqslant 7,$$
$$c^r_{ij} \equiv 0,\qquad 1 \leqslant i \leqslant 7, 0 \leqslant j \leqslant 7.$$

3.3.3.7. Hàm vòng p[k]

Hàm vòng thứ r là ánh xạ hợp (*composite mapping*)

p[k] : $M_{8x8}[GF(2^8)]$ " $M_{8x8}[GF(2^8)]$

được tham số hoá bởi ma trận khoá k Є $M_{8x8}[GF(2^8)]$ và được cho bởi :

p[k] ≡ σ[k] ° θ ° π ° γ

3.3.3.8. Bảng xếp lịch khoá

Bảng xếp lịch khoá được mở rộng khoá mật mã 512 bit K Є $M_{8x8}[GF(2^8)]$ thành các khoá vòng K^0,…, K^R :

$K^0 = K$,

$K^r = \rho[c^r](K^{r-1}),\ r > 0.$

3.3.3.9. Mật mã khối nội W

Mật mã toàn bộ khối 512 bit W[K] : $M_{8x8}[GF(2^8)]$ " $M_{8x8}[GF(2^8)]$, được tham số hoá bởi khoá mật mã nội 512 bit K, được định nghĩa như sau:

$$W[K] = \left(\bigcirc_{1}^{r=R} \rho[K^r] \right) \circ \sigma[K^0],$$

ở đó các khoá vòng K^0,…, K^R dẫn xuất từ K bởi bảng xếp lịch khóa. Số vòng ngẫu nhiên là R = 10.

3.3.3.10. Thêm các bit và tăng cường MD

Trước khi được băm, thông điệp M có chiều dài $L < 2^{256}$ được thêm một bit '1' và một số bit '0' được thêm vào để được chuỗi bit có chiều dài là bội số lẻ của 256 và cuối cùng là 256 bit nhị phân chứa chiều dài L của thông điệp

trước khi thêm các bit được thêm vào tạo thành một thông điệp m, thông điệp kết quả này được chia thành t khối m^1,…, m^t. Các khối này được xem như là các mảng byte bằng cách nhóm liên tục các bit thành các đoạn 8-bit.

3.3.3.11. Chức năng nén(Nguyên tắc nén)

WHIRLPOOL lặp chiến lược băm Miyaguchi Preneel trên t khối thông điệp thêm bit m_i, $1 \le i \le t$, dùng mật mã toàn bộ khối 512-bit W (dedicated 512-bit block cipher W) :

$$\eta_i = \mu(m_i),$$
$$H_0 = \mu(IV),$$
$$H_i = W[H_{i-1}](\eta_i) \oplus H_{i-1} \oplus \eta_i,\ 1 \leqslant i \leqslant t,$$

với IV(Initialisation Vector) là một chuỗi 512-bit '0'.

3.3.3.12. Tính thông điệp băm

Thông điệp băm WHIRLPOOL từ thông điệp nguồn M được định nghĩa như đầu ra H^t của hoạt động nén, ánh xạ ngược lại chuỗi bit:

WHIRLPOOL(M) $\equiv \mu^{-1}(H^t)$.

3.3.4. Đánh giá hàm băm Whirpool

- Có khả năng mở rộng phạm vi sử dụng hơn đa số các hàm băm khác.
- Hiệu quả trên đa số phần cứng (hỗ trợ cho các bộ xử lý 8-bit cũ, các bộ xử lý 32-bit hiện tại và các bộ xử lý 64-bit trong tương lai).
- Không yêu cầu không gian luu trữ quá mức cho cả mã lẫn các S-box.
- Được cài đặt hiệu quả trên môi trường có những ràng buộc như các thẻ thông minh, các thiết bị cầm tay,…; và hoạt động với hiệu suất rất cao khi hoạt động trên các bộ nhớ cache lớn hơn của các bộ xử lý hiện đại.
- Chỉ sử dụng những lệnh đơn giản được hỗ trợ sẵn trong bộ xữ lý máy tính, không dùng những lệnh không thông thường hay những câu lệnh từ các tiện ích.
- Chiều dài giá trị băm dài làm tăng khả năng chống lại các tấn công.

Chương 4. Giấu dữ liệu – Watermarking

Các chương trước đã trình bày các phương pháp bảo mật dữ liệu và các giải pháp hỗ trợ để cải tiến tốc độ. Chương này sẽ giới thiệu một phương pháp bảo mật dữ liệu khác cũng đang phát triển mạnh và thu hút nhiều người tham gia nghiên cứu: giấu dữ liệu (watermarking).

4.1. Giấu dữ liệu

Watermark có thể là một thông điệp, một file, một logo, v.v..

Giấu dữ liệu là quá trình nhúng dữ liệu watermark vào một đối tượng multimedia (text, hình ảnh, video, audio) nhằm một mục đích nào đó: bảo vệ watermark, bảo vệ bản quyền, v.v…; và nếu cần, watermark có thể trích ra sau đó.

(a) Nhúng không nhìn thấy (b) Nhúng có thể nhìn thấy

Hình 4.1. Hai mẫu watermark

4.2. Phân loại:

Kỹ thuật giấu dữ liệu có thể chia thành nhiều loại khác nhau theo các cách khác khác nhau.

- Phân loại theo tài liệu watermark thì có bốn dạng giấu dữ liệu:
  - Giấu dữ liệu trên text
  - Giấu dữ liệu trên ảnh
  - Giấu dữ liệu trên file audio
  - Giấu dữ liệu trên file video

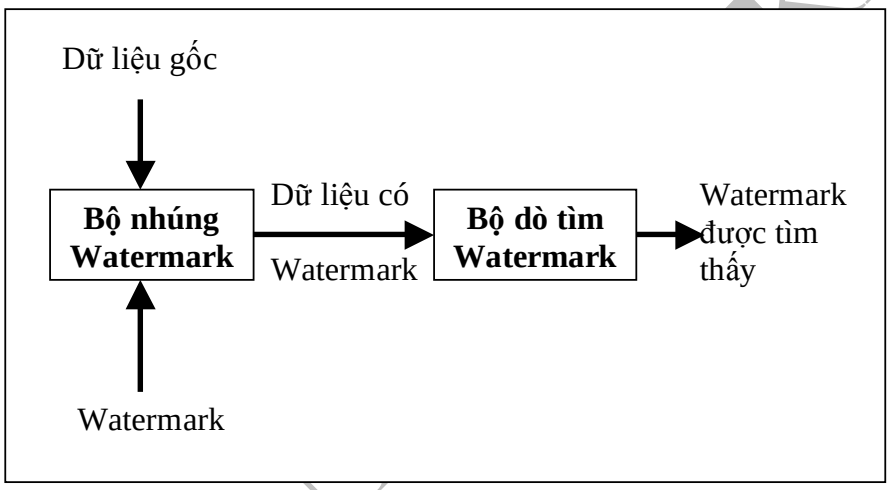
- Phân loại theo tính chất của watermark thì có hai dạng giấu dữ liệu:
  - *Giấu dữ liệu có thể nhìn thấy*: loại này xâm thực vào dữ liệu multimedia. Ví dụ: logo nhỏ ở góc màn hình có thể thấy được trong trạm truyền hình mạng

⇒ có thể cắt đi phần watermark

§ *Giấu dữ liệu không thể nhìn thấy:* loại này thuận lợi hơn loại có thể nhìn thấy, do vi trí của watermark nằm ở đâu chúng ta không biết được. Hơn nữa, để tránh trường hợp watermark bị cắt bỏ chúng ta có thể phân phối watermark trên toàn dữ liệu gốc. Và do không thể nhìn thấy được nên loại này ít ảnh hưởng đến hoạt động của dữ liệu.

4.3. Mô hình chung:

Thông thường, một hệ thống giấu dữ liệu bao gồm hai quá trình: nhúng watermark và dò tìm watermark.



Hình 4.2. Mô hình chung của hệ thống giấu dữ liệu

q **Bộ nhúng watermark**
§ Đây là qui trình nhúng một thông điệp vào dữ liệu gốc.
§ Đầu vào: thông điệp watermark và thông tin của dữ liệu gốc.
§ Đầu ra: dữ liệu được watermark.

q **Bộ dò tìm watermark**
§ Đây là qui trình trích thông điệp từ dữ liệu đã watermark.
§ Đầu vào: dữ liệu được watermark hay là dữ liệu được watermark đã bị chỉnh sửa, biến đổi ở mức độ nhất định nào đó
§ Đầu ra: thông điệp đã được nhúng (nếu có). Hầu hết các bộ dò tìm sẽ xác định xem có tồn tại watermark trong dữ liệu watermark hay không

4.4. Các yêu cầu của bài toán giấu dữ liệu

Tùy vào ứng dụng của watermarking và mục đích của ứng dụng, các yêu cầu khác nhau được đặt ra. Nhưng yêu cầu thường gặp là tính không cảm nhận được và tính độc lập với mục tiêu ứng dụng. Một ứng dụng watermarking có những yêu cầu cơ bản sau:

- Tính hiệu quả: xác suất thành công của quá trình nhúng. Có nghĩa là xác suất tìm thấy được watermark ngay sau khi nhúng.

- Tính trong suốt: là tính không cảm nhận được vết watermark sau khi được nhúng vào dữ liệu gốc. Chẳng hạn đối với một tấm hình, người sử dụng hầu như sẽ không thấy được bất kỳ sự thay đổi nào đối với tấm hình; đối với một đoạn phim video, người sử dụng cũng rất khó thấy được sự khác biệt so với phim gốc.

- Độ tin cậy: mức độ tương tự giữa dữ liệu gốc và dữ liệu có watermark. Nếu dữ liệu có watermark bị giảm chất lượng hay bị biến đổi trong quá trình truyền tải hay phát hành, thì độ tin cậy của hệ thống này chính là mức độ tương tự giữa nguồn dữ liệu gốc và dữ liệu có watermark sau quá trình truyền tải.

- Tính bền vững: khả năng dò tìm được vết watermark sau khi có những thao tác xử lý tín hiệu thông thường (lọc không gian (spatial filtering) trong ảnh, nén có mất dữ liệu, quét, biến dạng hình học (quay, dịch chuyển, tỷ lệ) v.v…

- Tải trọng dữ liệu: số bits của một watermark được mã hoá trong một đơn vị thời gian hoặc một nguồn dữ liệu. Ví dụ: đối với một tấm ảnh, tải trọng dữ liệu chỉ số bits watermark được mã hoá trong tấm ảnh; đối với âm thanh, tải trọng dữ liệu chỉ số bits watermark được nhúng vào 1 giây dữ liệu; đối với video, tải trọng dữ liệu chỉ đến số bits của một frame hoặc số bits của 1 giây.

- Chống tấn công: tùy thuộc vào mục đích của ứng dụng, tính chống tấn công ảnh hưởng đến qui trình thiết kế

- Khôi phục watermark cần hoặc không cần dữ liệu gốc

- Lấy watermark hay xác nhận sự xuất hiện của một watermark cho trước

- Độ bảo mật và khóa watermark

- Vấn đề chi phí: chi phí cho việc triển khai bộ nhúng và bộ dò tìm rất phức tạp và tuỳ thuộc vào mô hình phát triển. Về mặt kỹ thuật, hai vấn đề chính là tốc độ mà bộ nhúng và bộ dò tìm thực hiện và số các bộ nhúng và bộ dò tìm phải được triển khai.

4.5. Phương pháp giấu dữ liệu

4.5.1. Phương pháp giấu dữ liệu có thể nhìn thấy

Do đặc điểm của quá trình nhúng watermark có thể nhìn thấy, hệ thống nhúng watermark có thể nhìn thấy không yêu cầu trích watermark, nên với các phương pháp trong loại nhúng này, chúng tôi chỉ giới thiệu quá trình nhúng watermark.

4.5.1.1. Phương pháp dựa vào phép biến đổi Cosin từng phần

Trong phương pháp này, những block ở ngoài rìa sẽ được biến đổi thấp nhất để tránh sự méo mó quan trọng của ảnh. Quá trình nhúng watermark được thực hiện theo các bước sau:

1. Chia ảnh gốc I (ảnh được Watermark) và ảnh WaterMark W thành những block 8x8. Kích thước của 2 ảnh I và W có thể không bằng nhau.

2. Tìm các hệ số DCT của từng block của ảnh I

3. Với mỗi block của ảnh gốc I, tính giá trị độ xám được chuẩn hoá của block thứ n μ_n bằng công thức (1) (ép về khoảng [0.1,1]) và tính giá trị độ xám được chuẩn hoá của ảnh I μ bằng công thức (2)

$$\mu_n = \frac{\mathbf{C}_{00}(n)}{\mathbf{C}_{00max}} \qquad (1)$$

$$\mu = \frac{1}{N}\sum_{n=1}^{N} \mathbf{C}_{00}(n) \qquad (2)$$

4. Với những hệ số thích nghi DCT, tính sự khác biệt của những hệ số thích nghi DCT của block thứ n, σ_n bằng công thức (3) (ép về khoảng [0.1,1]), và tính sự khác biệt được chuẩn hoá của những hệ số DCT của block n, σ_n' bằng công thức (4). Đặt $\sigma_n^* = \ln\sigma_n$, ta có:

$$\sigma_n = \frac{1}{64}\sum_i\sum_j \left(c_{ij} - \mu_n^{AC}\right)^2 \qquad (3)$$

$$\sigma_n' = \frac{\sigma_n^*}{\sigma_{max}^*} \qquad (4)$$

5. Những block ở ngoài rìa (được xác định bằng toán tử cạnh Sobel), thì các hệ số co giãn α_n và hệ số nhúng β_n được lấy bằng với α_{max} và β_{max}.

6. Những block không ở ngoài rìa thì hệ số co giãn α_n và hệ số nhúng β_n được tính bằng công thức (5) và (6)

$$\alpha_n = \sigma'_n \, e^{-(\mu'_n - \mu')^2} \qquad (5)$$

$$\beta_n = \frac{1}{\sigma'_n}\left(1 - e^{-(\mu'_n - \mu')^2}\right) \qquad (6)$$

7. Tìm hệ số DCT của ảnh W. Khi đó, watermark được chèn vào ảnh gốc I bằng cách thay đổi hệ số DCT của ảnh gốc I bằng công thức (7) như sau:

$$c'_{ij}(n) = \alpha_n c_{ij}(n) + \beta_n w_{ij}(n) \quad n = 1,2\ldots \quad (7)$$

Trong đó:

- $c_{ij}(n)$, $w_{ij}(n)$ là hệ số DCT của block thứ n của ảnh I, ảnh W

4 **Ghi chú:** Trong thực tiễn, α_{min}, α_{max}, β_{min} và β_{max} thường được chọn là 0.95, 0.98, 0.07 và 0.17

4.5.1.2. Phương pháp chèn giá trị độ xám

Phương pháp này nhúng watermark (logo) bằng cách chèn các giá trị độc xám của Logo vào ảnh gốc. Công thức chèn watermark đơn giản như sau:

$$I_w(m,n) = \begin{cases} I(m,n) + \dfrac{\alpha_I}{903.3} * W(m,n) * I(m,n) & I(m,n) \le 2 \\ I(m,n) + \dfrac{\alpha_I C_1}{6.0976} * W(m,n) * I(m,n) & 2 < I(m,n) \le 64 \\ I(m,n) + \dfrac{\alpha_I C_2}{6.0976} * W(m,n) * I(m,n) & 64 < I(m,n) \le 128 \\ I(m,n) + \dfrac{\alpha_I C_3}{6.0976} * W(m,n) * I(m,n) & 128 < I(m,n) \le 192 \\ I(m,n) + \dfrac{\alpha_I C_4}{6.0976} * W(m,n) * I(m,n) & 192 < I(m,n) \le 256 \end{cases}$$

Trong đó:

- § $I_w(m,n)$: pixel tại toạ độ (m,n) của ảnh đã watermark
- § I(m,n): pixel tại toạ độ (m,n) của ảnh gốc
- § α_I: quyết định độ mạnh của watermark
- § C_1, C_2, C_3, C_4: là các hệ số

4 Ghi chú: Thông thường, α_I, C_1, C_2, C_3, C_4 được chọn là: 0.0001, 4, 9, 16 và 1

4.5.2. Phương pháp giấu dữ liệu không thể thấy

Các phương pháp được trình bày trong phần này dựa vào phép biến đổi Wavelet để nhúng watermark vào ảnh và các watermark được xem như một dãy nhị phân $\{w_i, w_i \in \{0,1\}\}$

4.5.2.1. Phương pháp lượng hoá hệ số biến đổi wavelet

Bằng phương pháp này, ảnh sau khi nhúng không thay đổi kích thước và không bị méo mó. Chiều dài watermark phụ thuộc vào kích thước của ảnh gốc.

Không cần ảnh gốc khi trích watermark.

- **Quá trình nhúng watermark**

Quá trình nhúng watermark được thực hiện theo các bước sau:

1. Thực hiện phép biến đổi wavelet trên ảnh gốc và tính các hệ số biến đổi wavelet (hệ số WTC): C_i , i=0..N (N tuỳ thuộc vào mức biến đổi Wavelet)
2. Thực hiện phép lượng hoá hệ số wavelet: $C'_i = C_i/M$ (M: hệ số lượng hoá)
3. Lặp:
   Nếu (w_i=0)
   Biến đổi hệ số WTC C'_i thành số chẵn
   Ngược lại:
   Biến đổi hệ số WTC C'_i thành số lẻ
   Dừng khi hết watermark hoặc duyệt hết ảnh
4. Thực hiện phép lượng hoá ngược hệ số wavelet: $C_i = C'_i*M$
5. Thực hiện phép biến đổi wavelet ngược với hệ số wavelet C_i, ta dược ảnh đã watermark

- **Quá trình trích watermark**

Quá trình trích watermark đơn giản chỉ là quá trình ngược với quá trình nhúng watermark và watermark được trích ra theo công thức:

Nếu (C'_i là số lẻ) thì $w_i = 1$;
Ngược lại: $w_i = 0$;

với C'_i là hệ số biến đổi wavelet của ảnh đả watermark sau khi đã qua bước lượng hoá.

4.5.2.2. Phương pháp dựa vào sự khác biệt giữa các hệ số wavelet kề nhau

Phương pháp này sử dụng phép biến đổi Wavelet để nhúng watermark vào ảnh. Chất lượng ảnh sau khi nhúng không thay đổi kích thước và không bị méo mó hình ảnh. Chiều dài watermark phụ thuộc vào kích thước của ảnh gốc và phụ thuộc vào tính chất của ảnh: độ dài của watermark tỉ lệ thuận với số điểm ảnh phẳng kề nhau.

Không cần ảnh gốc khi trích watermark.

- **Quá trình nhúng watermark**

Quá trình nhúng watermark được thực hiện theo các bước sau:

1. Thực hiện phép biến đổi wavelet trên ảnh gốc và tính các hệ số biến đổi wavelet WTC: C_{ij}
2. Chia ảnh gốc thành nhiều block 2x1 theo chiều dọc và mỗi block nhúng một bit watermark
3. Lặp
   a. Tính sự khác biệt tuyệt đối d(m,n) và giá trị trung bình M(m,n) của hai hệ số WTC trong mỗi block {(m,2n),(m,2n+1)}:
      - $d = |C_{m,2n} - C_{m,2n+1}|$
      - $M = (C_{m,2n} - C_{m,2n+1})/2$
   b. Nếu d<=T thì w_j được nhúng vào block này như sau:
      - $w_j = 1$

        Nếu ($C_{m,2n} \le C_{m,2n+1}$ || d<2a)

        {

        $C_{m,2n} = M+a$

        $C_{m,2n+1} = M-a$

        }

        Ngược lại thì không thay đổi $C_{m,2n}$ và $C_{m,2n+1}$
      - $w_j = 0$

        Nếu ($C_{m,2n} \ge C_{m,2n+1}$ || d<2a)

        {

        $C_{m,2n} = M-a$

        $C_{m,2n+1} = M+a$

        }

        Ngược lại thì không thay đổi $C_{m,2n}$ và $C_{m,2n+1}$

   Ngược lại thì block này không dùng để nhúng watermark

   Dừng khi hết watermark hoặc hết kích thước ảnh

4. Thực hiện phép biến đổi wavelet ngược với hệ số wavelet C_i, ta dược ảnh đã watermark

Trong đó:

- Ø T: ngưỡng xác định block nào là phẳng. Nếu một block có d>T thì block đó thuộc vào phần phẳng và ngược lại
- Ø a là tham số tương quan chất lượng (a<T/2)

q **Quá trình trích watermark**

Quá trình trích watermark được thực hiện theo các bước sau:

1. Thực hiện phép biến đổi wavelet trên ảnh gốc và tính các hệ số biến đổi wavelet WTC: C_{ij}
2. Chia ảnh gốc thành nhiều block 2x1 theo chiều dọc
3. Lặp
   a. Tính sự khác biệt tuyệt đối d(m,n) và giá trị trung bình M(m,n) của hai hệ số WTC trong mỗi block {(m,2n),(m,2n+1)}:
      - § $d = |C_{m,2n} - C_{m,2n+1}|$
      - § $M = (C_{m,2n} - C_{m,2n+1})/2$
   b. Nếu d<=T thì block này có chứa watermark và:
      - § Nếu $C_{m,2n}<C_{m,2n+1}$ thì w = 1
      - § Ngược lại: w = 0;

   Ngược lại thì block này không chứa watermark

   Dừng khi hết watermark (theo qui định của từng người) hoặc duyệt hết ảnh

Khi đó, $W=\{w_i\}$ là watermark cần trích.

4 **Ghi chú:** Trong thực tế, T và a thường được chọn là 6.2 và 2.1

4.5.2.3. Phương pháp dựa vào phép biến đổi Wavelet dư thừa

Phương pháp này sử dụng Redundant Wavelet Trasform (RDWT: phép biến đổi Wavelet dư thừa) và độ dài của watermark phụ thuộc vào những hệ số đáng giá của ảnh.

Không cần ảnh gốc khi trích watermark.

q **Quá trình nhúng watermark**

RDWT hỗ trợ tiến tình nhúng watermarking thông qua một mặt nạ hướng dẫn nơi watermark được thêm vào.

Trong kỹ thuật này, nhiễu Gaussian trắng (white Gaussian noise) được dùng như chuỗi mã (watermark), nhiễu này được thêm vào hệ số RDWT của ảnh:

$$f'_{\{V,H,D\}}(x,y) = f_{\{V,H,D\}}(x,y) + \alpha v_{\{V,H,D\}}(x,y) w_{\{V,H,D\}}(x,y)$$

Trong đó:

- $f_{\{V,H,D\}}(x,y)$: hệ số RDWT của ảnh
- $v_{\{V,H,D\}}(x,y)$: watermark được nhúng
- V, H, D đại diện cho thành phần dọc, ngang và chéo.
- $w_{\{V,H,D\}}(x,y)$: là mối tương quan của mặt nạ phụ của thành phần dọc, ngang và chéo.

Trong kỹ thuật này, mặt nạ quan hệ được tạo bằng:

$$w_{i\{V,H,D\}}(x,y) = \prod_{j=1}^{i} f^{*}_{j\{V,H,D\}}(x,y)$$

Trong đó:

- $f^{*}_{j\{V,H,D\}}(x,y)$ là chuẩn hoá của hệ số RDWT tại scale thứ j của phép biến đổi VD:

$$f^{*}_{j\{V,H,D\}} = \frac{f_{j\{V,H,D\}}(x,y) * f_{j\{V,H,D\}}(x,y)}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f_{j\{V,H,D\}}(x,y)^2}$$

- $w_{i\{V,H,D\}}(x,y)$: là mặt nạ quan hệ chứa từ scale thứ 1 đến scale thứ i của phép biến đổi.

q **Quá trình trích watermark**

Việc tìm watermark có tồn tại hay không đơn giản chỉ là một tiến trình ngược với tiến trình nhúng watermark. Sau khi thực hiện RDWT, phương pháp đơn giản để tìm ra watermark tương tự và đồng nhất là dựa vào tính tương tự:

$$\rho = \max_{\{V,H,D\}}[\rho_S]$$

với phép đo tính tương tự cho subband s là:

$$\rho_S = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f'_S(x,y) v_S(x,y)$$

Trong đó:

- $f'_s(x,y)$: hệ số RDWT của ảnh được kiểm tra
- $v_s(x,y)$: watermark được nhúng vào chiều dọc, ngang, và chéo
- ρ: hệ số tương tự lớn nhất từ mỗi hướng khác nhau

4 Ghi chú: Nếu trong quá trình dò tìm có thêm ảnh gốc thì công thức tính độ tương tự của subband s sẽ là:

$$\rho_S = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1}(f'_S(x,y) - f_S(x,y))\, v_S(x,y)$$

4.5.2.4. Phương pháp dựa trên việc chia block thích nghi

Phương pháp này sử dụng mô hình chia ảnh thành từng block thích nghi sử dụng những tần số thấp của phép biến đổi cosin rời rạc và phép biến đổi wavelet. Ảnh được chia thành các block theo tính chất cục bộ và HVS và chọn ra các block quan trọng. Phương pháp này đủ mạnh để chống lại các tấn công như: lọc, cắt xén, phép quay, nén mất thông tin như JPEG hay JPEG2000,… mà không bị giảm chất lượng thông tin nhúng.

- **Thuật toán chia ảnh thành các block thích nghi**

1. Chia ảnh thành các block 64x64
2. Với mỗi block tính hệ số khác biệt V_k và hệ số trung bình μ_k.

$$\mu_k = \frac{\sum_{i=1}^{n} x_k(i)}{n}$$

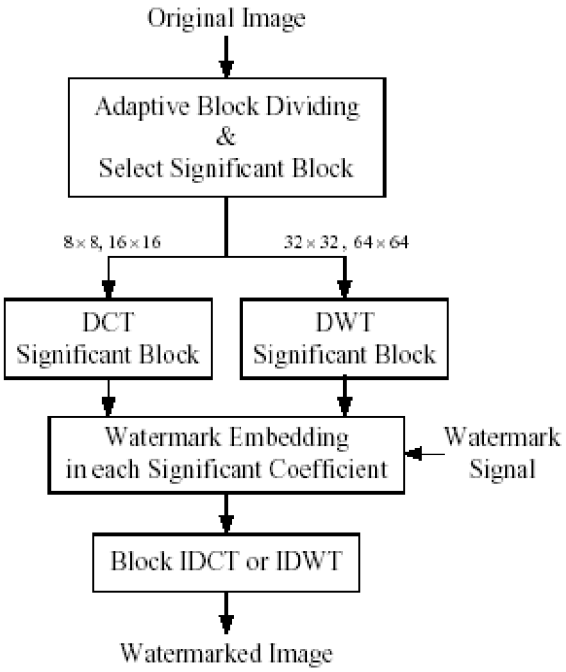
$$V_k = \frac{\sum_{i=1}^{n} (\mu_k - x_k(i))^2}{n-1}$$

Trong đó:

- k: số block
- n số pixel trong 1 block
- $x_k(i)$: là pixel thứ i trong block thứ k

3. Nếu V_k và μ_k cao hơn ngưỡng V_{th} và μ_{th} thì hoặc là chia lại block với kích thước nhỏ nhất là 8x8 và dừng tiến trình chia hoặc là những block được chia đôi tiếp tục và quay lại B2

q **Quá trình nhúng watermark**



Hình 4.3. Sơ đồ nhúng watermark bằng phương pháp dựa trên block thích nghi

- Chia ảnh cần watermark thành các block theo thuật toán trên
- Chọn những block quan trọng (tần số thấp và được chọn theo HVS của con người, vì ở các tần số thấp thì giá trị ít bị thay đổi khi bị tấn công; tuy nhiên sẽ làm thay đổi ảnh)
- Thực hiện phép biến đổi ảnh trên các block đã chọn (với block có kích thước 64x64 và 32x32 thì dùng phép biến đổi wavelet còn với các block có kích thước 8x8 và 16x16 thì dùng phép biến đổi Cosin rời rạc DCT
- Nhúng watermark vào các block được chọn

  Nếu (w=0)

  $$C'(i,j) = C(i,j) + \alpha C(i,j)\log V_k$$

  Ngược lại:

  $$C'(i,j) = C(i,j) - \alpha C(i,j)\log V_k$$
- Thực hiện phép biến đổi ngược

Trong đó: α là chiều dài của watermark cần nhúng

q **Quá trình trích watermark**

Quá trình trích watermark phải cần đến ảnh gốc. Tương tự như tiến trình nhúng watermark, ảnh watermark cũng được chia thành các block thích nghi

như trên và các block cũng phải phép biến đổi. Khi đó, việc trích watermark như sau:

Nếu $(C'(i,j) > C(i,j) + \beta C(i,j)\log V_k)$ thì w=1
Ngược lại:
Nếu $(C'(i,j) < C(i,j) - \beta C(i,j)\log V_k)$ thì w=0

Trong đó: $\beta < \alpha$

4.6. Các dạng tấn công

Một số dạng tấn công trên watermark như:

- Nâng cao chất lượng tín hiệu (gọt, tăng độ tương phản, sửa màu)
- Nhiễu bổ sung và nhiễu bội (Gauss, đồng bộ, v.v..)
- Lọc tuyến tính (lọc lowpass, highpass, bandpass)
- Lọc phi tuyến (lọc morphology, median)
- Nén mất thông tin (âm thanh MP3)
- Các biến đổi affine toàn cục hay cục bộ (dịch, quay, tỉ lệ, shearing)
- Giảm dữ liệu (cắt, dán, thay đổi lược đồ xám)
- Kết hợp dữ liệu (chèn logo, kết hợp cảnh)
- Thay đổi mã hóa (H.263→MPEG-2, WAV→MP3, MP3→WAV…)
    - Chuyển đổi D/A và A/D
    - Đa watermarking
    - Tấn công dạng thông đồng
    - Trung bình thống kê
    - Tấn công Mosaic

4.7. Ứng dụng của phương pháp giấu dữ liệu

Ngày nay, phương pháp giấu dữ liệu được ứng dụng rộng rãi trong nhiều lĩnh vực như:

- Theo dõi phát sóng
- Xác minh chủ sở hữu
- Chứng minh sở hữu
- Theo dõi giao dịch
- Chứng thực: nhúng thông tin về nguồn gốc của dữ liệu
- Điều khiển sao chép: ví dụ như trong 1 đĩa CD hoặc VCD, ta nhúng vào một thông tin cho phép chép hay không? (cho sao chép, chỉ được sao chép 1 lần, không được sao chép)

§ Điều khiển thiết bị

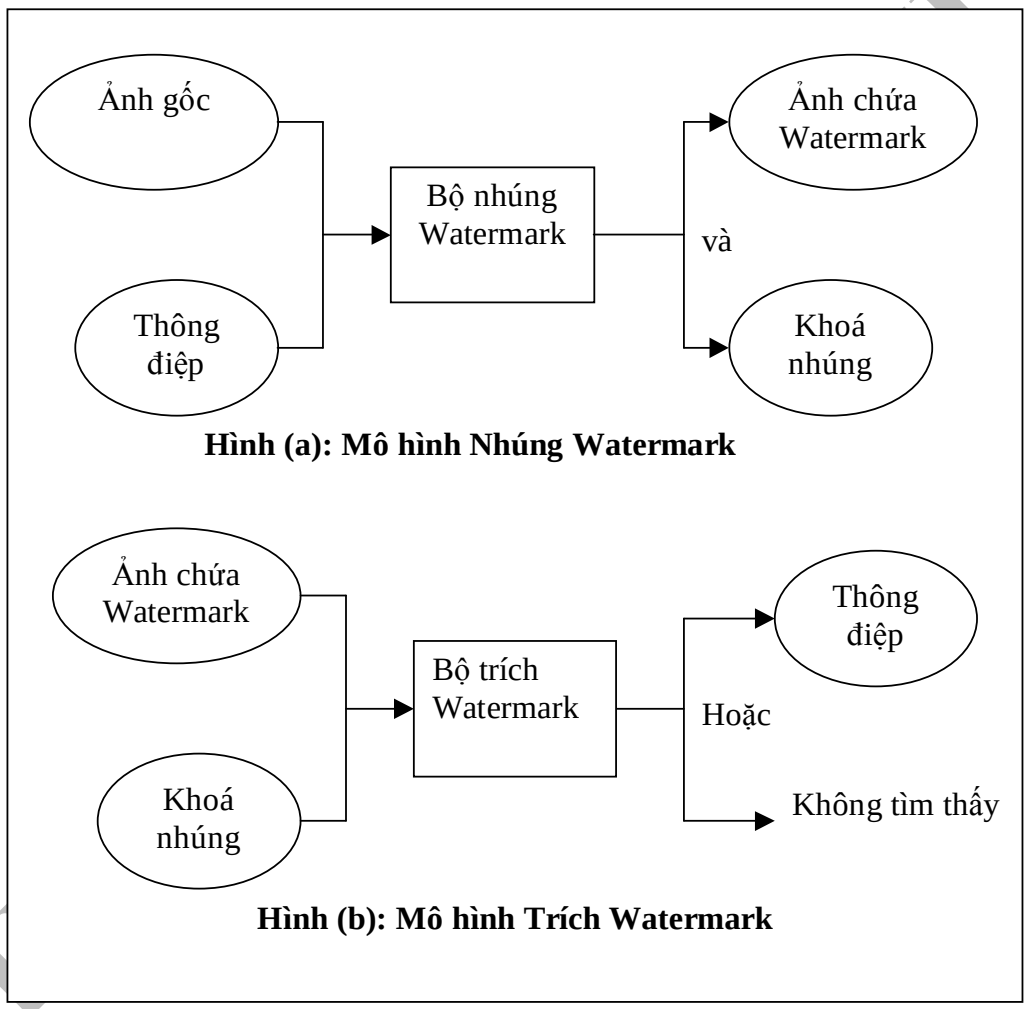
§ Xác nhận thông tin dữ liệu có thay đổi hay không: nhúng vào trong ảnh một đặc tính gì đó, khi kiểm tra thì kiểm tra xem ảnh có những đặc tính giống như ảnh gốc hay không.

Chương 5. Một số ứng dụng

Chương này sẽ trình bày một số ứng dụng được xây dựng dựa trên các cơ sở lý thuyết đã trình bày trong các chương trước.

5.1. Giấu tin trên ảnh

5.1.1. Nghi thức giấu tin đa tầng trên ảnh



Hình 5.1. Mô hình hệ thống giấu watermark trên ảnh

- **Format thông điệp nhúng**

Thông điệp được xem như một dãy các ký tự: $m_0, m_1, \ldots, m_N$
Watermark xem như dãy các bit: $w_0, w_1, \ldots, w_K$ và chứa các thông tin:

- Khoá nhúng

§ Vị trí của thông điệp tiếp theo
§ Thông điệp

q **Thuật toán**

Sử dụng thuật toán giấu tin dựa vào sự khác biệt giữa các hệ số wavelet kề nhau (4.5.2.2) với biến đổi wavelet Harr

Ø ***Thuật toán nhúng Watermark***

§ Đầu vào: - Bảng màu ảnh gốc
- Thông điệp

§ Đầu ra: - Bảng màu chứa thông tin watermark
- Khoá nhúng
- Nhúng thành công hay không

§ Thuật toán

1. Biến đổi wavelet với thành phần màu R của hệ màu RGB của bảng màu ảnh gốc
2. Dò tất cả các thông điệp có trong ảnh để lấy các khoá nhúng trước đó
3. Tính toán kích thước watermark
4. Kiểm tra xem ảnh có chứa đủ watermark không
   Nếu ảnh chứa hết watermark
   4.1. Phát sinh khoá nhúng
   4.2. Chuyển thông tin watermark thành dãy bit
   4.3. Nhúng dãy bit vào các hệ số wavelet
   4.4. Thực hiện phép biến đổi wavelet ngược
   4.5. Trả về TRUE
   Ngược lại: trả về FALSE

Ø ***Thuật toán trích Watermark***

§ Đầu vào: - Bảng màu ảnh
- Khoá: K

§ Đầu ra: - Thông điệp
- Trích thành công hay không

§ Thuật toán

1. Biến đổi wavelet với thành phần màu R của hệ màu RGB của bảng màu ảnh gốc
2. Dò các thông điệp trong ảnh.
3. Nếu tìm thấy thông điệp có khoá K
   3.1. Đọc thông điệp
   3.2. Trả về TRUE
   Ngược lại: trả về FALSE

Trong đó, phép biến đổi Harr Wavelet trên ảnh theo dòng hay theo cột như sau:

- $I_{WT}[i] = h_0 * I[2i] + h_1 * I[2i + 1]$
- $I_{WT}[i + N/2] = g_0 * I[2i] + g_1 * I[2i + 1]$

và phép biến đổi ngược:

- $I[2i] = h_0 * I_{WT}[i+N/2] + h_1 * I_{WT}[i+N/2+1]$
- $I[2i + 1] = g_0 * I_{WT}[i+N/2] + g_1 * I_{WT}[i+N/2+1]$

với:

§ $h0 = h1 = g0 = \frac{1}{\sqrt{2}}$, $g1 = -g0$

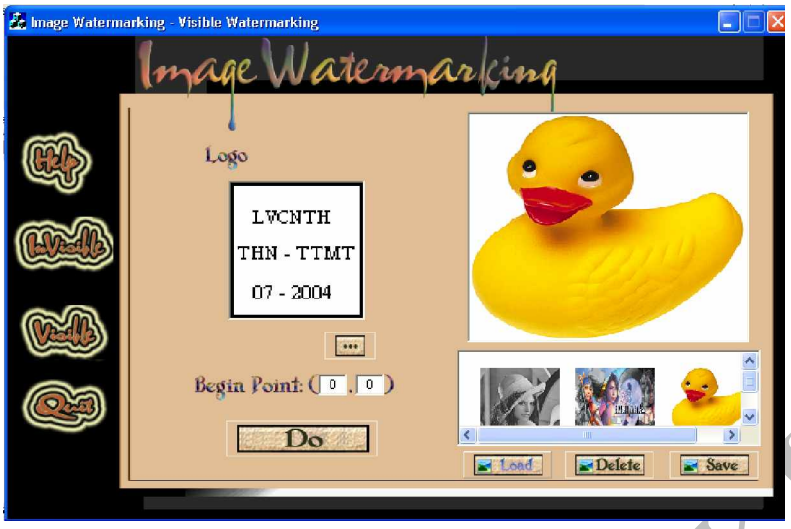
§ $I_{WT}[k]$: là giá trị điểm ảnh thứ k trên một dòng hay một cột của ảnh sau khi thực hiện phép biến đổi Wavelet.

§ I [k]: là giá trị điểm ảnh thứ k trên một dòng hay một cột của ảnh gốc

§ N: số điểm ảnh trên một dòng hay một cột

5.1.2. Giao diện ứng dụng

Hình 5.2. Màn hình giao diện nhúng không nhìn thấy được

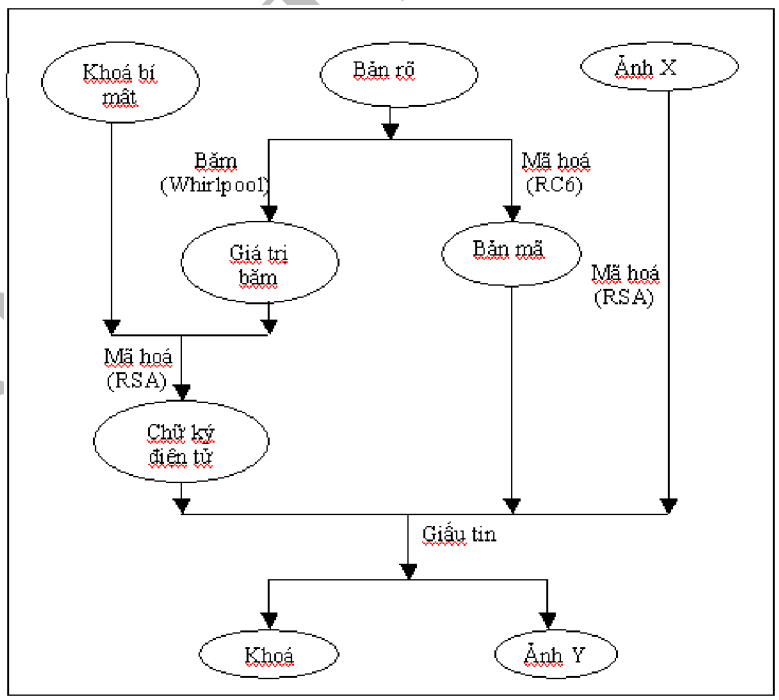


Hình 5.3. Màn hình giao diện nhúng nhìn thấy được

5.2. Mô hình chữ ký điện tử

5.2.1. Mô hình tạo chữ ký

A muốn gửi cho B một thông điệp M một cách đáng tin cậy trên kênh truyền không an toàn và B muốn phục hồi lại thông điệp M và xác thực M là từ A gửi chứ không phải từ một người nào khác, A sẽ dùng mô hình tạo chữ ký, và B sẽ sử dụng mô hình xác thực chữ ký như sau:



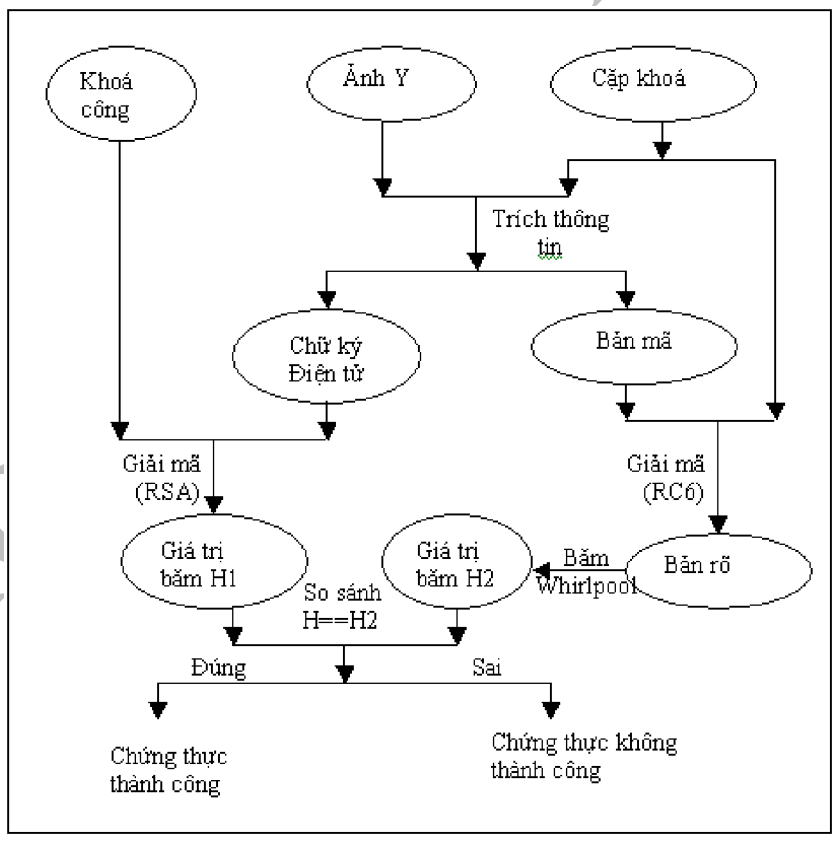
Hình 5.4. Mô hình tạo chữ ký điện tử

- **Thuật toán tạo chữ ký**
  - **Đầu vào:**
    - Bản rõ: thông tin cần gởi đến người nhận
    - Khoá bí mật của người gởi
    - Ảnh X để nhúng thông tin
  - **Đầu ra:** ảnh Y chứa chữ ký và thông tin đã mã hoá; cặp khoá
  - **Thuật toán:**

1. Phát sinh khoá mã hoá
2. Mã hóa bản rõ tạo thành bản mã.
3. Băm bản rõ tạo giá trị băm.
4. Mã hóa giá trị băm để tạo chữ ký điện tử.
5. Phát sinh khoá nhúng
6. Nhúng bản mã và chữ ký điện tử vào ảnh

5.2.2. Mô hình chứng thực chữ ký điện tử

B nhận được ảnh X, để trích được thông điệp M và xác thực thông điệp M có phải được gửi từ A hay không, B sẽ dùng mô hình chứng thực chữ ký điện tử sau:



Hình 5.5. Mô hình chứng thực chữ ký điện tử

- **Thuật toán chứng thực chữ ký điện tử**
  - **Đầu vào:**
    - Ảnh Y chứa bản mã của thông điệp và chữ ký
    - Các khoá: khoá công khai của người gởi, cặp khoá
  - **Đầu ra:**
    - Nếu chứng thực thành công, thuật toán trả về thông điệp của người gởi
    - Ngược lại: chứng thực không thành công.
  - **Thuật toán**:

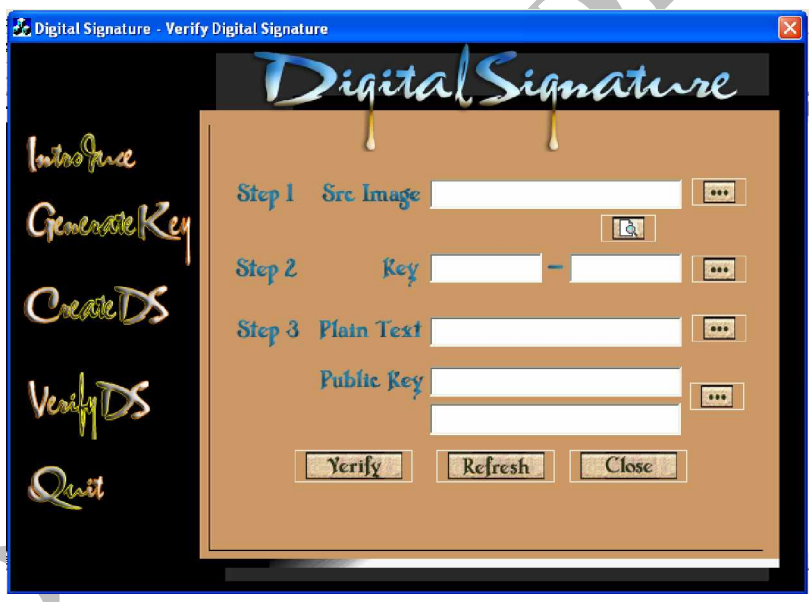
1. Trích thông tin từ ảnh Y: bản mã và chữ ký điện tử.
2. Giải mã bản mã thành bản rõ.
3. Giải mã chữ ký điện tử cho ra giá trị băm H1.
4. Băm bản rõ thu được từ Bước 2 thu được giá trị băm H2.
5. So sánh hai giá trị băm H1 và H2.

Nếu (H1=H2) thì chứng thực thành công và trả về thông điệp
Ngược lại: chứng thực không thành công.

5.2.3. Giao diện ứng dụng

Hình 5.6. Màn hình giao diện phát sinh cặp khoá

Hình 5.7. Màn hình giao diện tạo chữ ký điện tử



Hình 5.8. Màn hình giao diện chứng thực chữ ký điện tử

5.3. Nhúng tin vào phim và ứng dụng

5.3.1. Mô hình nhúng cơ sở dữ liệu trên phim

5.3.1.1. Tổ chức Cơ sở dữ liệu

File video được xem là một dãy các frame ảnh và được tổ chức như sau (mỗi frame ảnh là một block):

HEADER	FAT	B_0	B_1	B_2	…..	B_N-1	B_N

- **Header:** gồm có các trường sau, nằm ở các Frame đầu tiên:
  - Key: dùng để truy cập ổ cứng ảo
  - Label: nhãn của ổ cứng ảo
  - Size: chứa dung lượng đĩa(Gb-Mb-Kb-b)
  - Empty Size: dung lượng đĩa còn trống (Gb-Mb-Kb-b)
  - NumBlock: số Block

- **Bảng FAT:** ở các Frame tiếp theo Header và có các trường:
  - Flag: cho biết Block này đã được sử dụng chưa (trống, đang dùng, đã xoá)
  - NumByteUsed: số byte được ghi trong Block (nếu là tập tin) hoặc chứa số entry của thư mục (nếu là RDET)
  - NumByteEmpty: số byte còn trống trong block
  - NextBlock: chỉ số của Block kế tiếp

- **Bảng thư mục gốc RDET:**
  - Name (8 byte): tên file
  - Ext (3 byte): phần mở rộng
  - Type: cho biết file này là loại gì (tập tin, thư mục)
  - CreateDate: ngày tạo file hay thư mục
  - iBlock: chỉ số của Block đầu
  - Flag: cờ cho biết trạng thái của file này (đã xoá, đang dùng)

5.3.1.2. Tập lệnh trên ổ cứng ảo

Trên ổ cứng ảo hỗ trợ các chức năng sau:

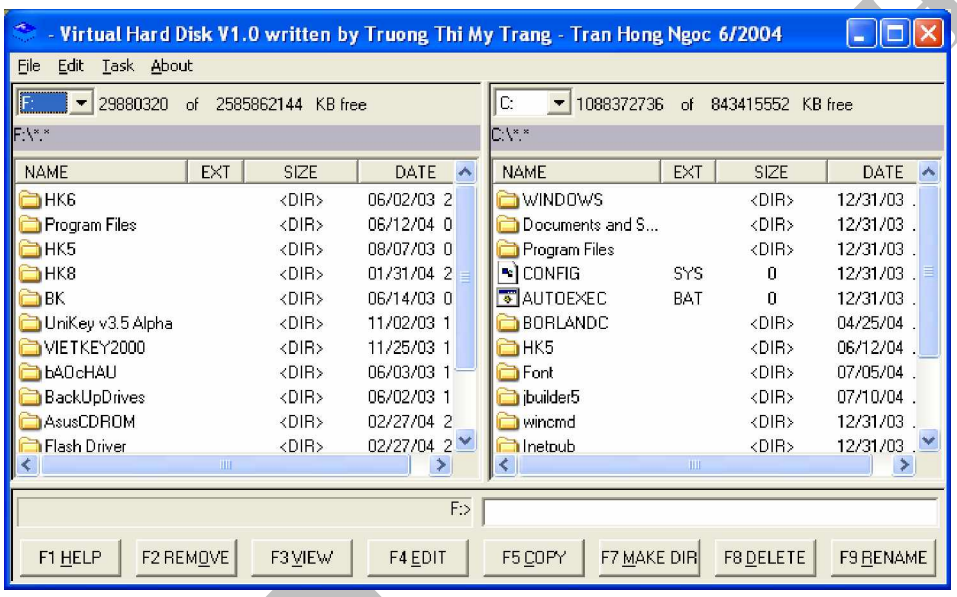
1. Format ổ cứng ảo
2. Mở ổ cứng ảo
3. Đóng ổ cứng ảo
4. Tạo thư mục
5. Sao chép tập tin, thư mục
6. Xoá tập tin, thư mục
7. Đổi tên tập tin, thư mục
8. Phục hồi các file bị xoá trong ổ cứng ảo
9. Tìm kiếm trên ổ cứng ảo

5.3.1.3. Thuật toán

File video được xem như một tập các bitmap, trong đó mỗi bitmap là một frame ảnh và áp dụng thuật toán nhúng tin trên ảnh để nhúng dữ liệu vào file video.

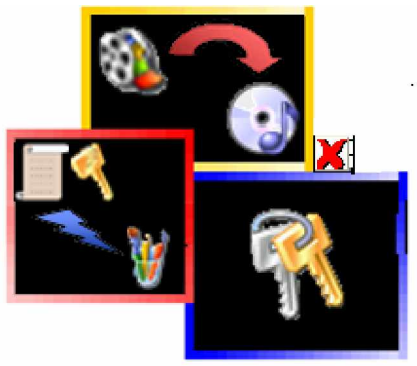
- **Thuật toán nhúng tin trên file video:**
  - Đầu vào: file video, thông tin nhúng, chỉ số của frame cần nhúng
  - Đầu ra: file video có chứa thông tin nhúng
  - Thuật toán:
    1. Đọc frame ảnh cần nhúng
    2. Nhúng thông tin lên frame ảnh
    3. Lưu lại frame ảnh vào file video

5.3.2. Giao diện ứng dụng



Hình 5.9. Màn hình giao diện ứng dụng ổ cứng ảo

5.4. Giao diện của chương trình chính



Hình 5.10. Giao diện của chương trình chính

Chương 6. Kết luận – Hướng phát triển

6.1. Kết luận

Luận văn đã trình bày khá đầy đủ lý thuyết về các phương pháp bảo mật thông tin đang phát triển mạnh hiện nay như: mã hoá khoá công khai, mã đối xứng, giấu tin và giải pháp nâng cao tốc độ xử lý như hàm băm. Đồng thời, luận văn cũng xây dựng một số ứng dụng minh hoạ cho các phương pháp này như: nhúng tin trên ảnh, chữ ký điện tử, nhúng tin vào file video.

Hiện tại, luận văn đạt được một số kết quả sau:

- **Nhúng tin trên ảnh**
  - **Ưu điểm**
    - Cho phép nhúng nhiều watermark trên một file ảnh mà chất lượng thông tin của mỗi lần nhúng không bị thay đổi
    - Watermark không bị biến dạng qua phép quay
    - Chất lượng và kích thước ảnh sau khi nhúng không bị thay đổi
    - Hỗ trợ các tiện ích khác: nhúng visible, load ảnh, lưu ảnh
  - **Khuyết điểm**
    - Khả năng chống lại các dạng tấn công còn kém
- **Chữ ký điện tử**
  - **Ưu điểm**
    - Hỗ trợ Wizard hướng dẫn sử dụng cho người dùng
    - Bản rõ có thể là tất cả các loại file (*.txt, *.bmp, *.pdf,…)
    - Tính bảo mật cao
    - Giao diện tiện sử dụng
  - **Khuyết điểm**
    - Chương trình còn chậm khi nhúng tin trên ảnh có kích thước lớn
- **Nhúng tin vào file video**
  - **Ưu điểm**
    - Hỗ trợ đầy đủ các chức năng cơ bản trên ổ cứng ảo
    - Có thể nhúng tất cả các loại file (*.txt, *.bmp, *.pdf, *.avi,…)
    - Tổ chức cơ sở dữ liệu trên ổ cứng ảo linh động: kích thước từng entry trong bảng thư mục gốc và bảng FAT thay đổi tuỳ thuộc vào từng file avi
    - Tính bảo mật tương đối tốt
    - Giao diện tiện sử dụng

Ø **Khuyết điểm**

§ Khi format ổ cứng ảo còn chậm do thực hiện phép biến đổi wavelet trên toàn file

§ Hiện chỉ hỗ trợ cho file avi chỉ chứa hình ảnh với bảng màu 24bit

Tuy nhiên, do phạm vi của đề tài rộng và thời gian thực hiện hạn hẹp, luận văn chưa nghiên cứu kỹ một số vấn đề

6.2. Hướng phát triển

Dựa trên các kết quả đã thực hiện trong luận văn, chúng tôi đề xuất các hướng cải tiến sau:

§ Mở rộng ứng dụng ổ cứng ảo trên các chuẩn file khác

§ Nghiên cứu phương pháp nhúng tin khác để tăng độ an toàn cho thông tin nhúng khi bị tấn công (resize ảnh, nén mất thông tin, …) và tăng độ dài của watermark (sử dụng cả ba thành phần màu để nhúng watermark).

§ Cải tiến tốc độ load ảnh

§ Cài đặt các lược đồ của phương pháp mã hoá ECC

Tài liệu tham khảo

[1] Saraju Prasad Mohanty, "Watermarking of Digital Images", A master of Engineering in System Science and Automatic, Electrical Engineering Department, Indian Institue Of Science Bangalore, India, January, 1999

[2] Kumiko, Kenggo, Atsushi, Makoto Fujimura, Hiroki Imamura and Hideo Kuroda, "An adaptive data hiding method using best combination neighboring pairs of WTC for division into flat/non-flat parts", Faculty of Engineering, Nagasaki University, Japan

[3] Jian-Guo Cao, James E. Fowler, and Nicholas H. Younan, "An image-adaptive watermark based on a redundant wavelet transform", Department of Electrical & Computer Engineering Mississippi State University

[4] Deepa Kundur and Dimitrios Hatzinakos, "Digital watermarking using multiresolution wavelet decomposition", Department of Electrical and Computer Engineering University of Toronto, Toronto, Ontario, Canada

[5] S. Asif Mahmood Gilani and A. N. Skodras, "DLT-based digital image watermarking", Electronics Laboratory, University of Patras, Greece

[6] Paulo S.L.M. Barreyo and Vincent Rijment, "The Whirlpool Hashing Function", Sao Paulo, Brazil

[7] Simon Blake Wilson, "Elliptic Curve Crytography", Certicom 1999

[8] S. Blake-Wilson, D. Brown, "RFC3278", Certicom Corp

[9] Joel Allardyce, Nitesh Goyal, "Elliptic Curve Cryptography", April 15, 2004

[10] David Pointcheval, "Provable Security in Cryptography, DL-based Systems", Ecole normale supérieure, France, ECC - Sept 24th 2002 – Essen

[11] Victor *S.* Miller, "Use of Elliptic Curves in Cryptography", Exploratory Computer Science, IBM Research, P.O. Box 21 8, Yorktown Heights, Y 10598

[12] Ross Anderson and Eli Biham, “Tiger_ A Fast New Hash Function”, Cambridge University, England

[13] Blaine Hoffman and Evan Lewis, “Tiger Hash Summary Paper”, CS 402 – Network Security

[14] Bart Preneel, K.U.Leuven, “Hash functions”, Version 2.b — January 18, 2004

[15] Taizo Shirai, Kyoji Shibutani, “On the di_usion matrix employed in the Whirlpool hashing function”

[16] Trần Trọng Tuyên, “Nghiên cứu nghi thức thực hiện chữ ký điện tử và ứng dụng. trong mô hình thương mại điện tử”, Luận văn thạc sĩ khoa học năm 2003

[17] Nguyễn Văn Hùng, “Nghiên cứu chữ ký điện tử sử dụng bảo vệ thông tin trên mạng”, Luận văn thạc sĩ tin học 2003

[18] Trần Minh Triết , Lương Hán Cơ, “Mã hoá và ứng dụng”, Luận văn cử nhân tin học năm 2001

[19] Nguyễn Quang Hưng, “Phép biến đổi Wavelet và Ứng dụng”, Luận văn Thạc sĩ toán học năm 2001

[20] Nguyễn Văn Diêu, “Biến đổi Wavelet và ứng dụng trong tìm kiếm trên cơ sở dữ liệu ảnh”, Luận văn thạc sĩ tin học, năm 2003

Phụ lục A: Biến đổi Wavelet

Trong phần này, chúng tôi chỉ giới thiệu các khái niệm cơ bản về wavelet và các tính chất của nó. Một số các tính chất được cho dưới dạng định lý không chứng minh (chi tiết chứng minh này xem trong [19][20]).

1 Biến đổi wavelet liên tục

1.1 Định nghĩa

- Xét hàm số ψ: **R** $\rightarrow$ **C** thoả:

$$\psi \in L^2(R), \|\Psi\| = 1,$$

$$C_\psi = 2\pi \int_{-\infty}^{+\infty} d\xi \frac{|\hat{\psi}(\xi)|^2}{|\xi|}, \quad 0 < C_\psi < +\infty$$

thì ψ được gọi là hàm wavelet (hay wavelet).

- Biến đổi wavelet liên tục (còn gọi là biến đổi tích phân wavelet) của một hàm số f $\in$ **L^2(R)** là:

$$W_f(a,b) = \frac{1}{|a|^{1/2}} \int_{-\infty}^{+\infty} f(t)\overline{\psi\left(\frac{t-b}{a}\right)}dt, \qquad a, b \in R, a \neq 0$$

trong đó

- ψ là wavelet thoả $\int_{-\infty}^{+\infty} \psi(t)dt = 0$
- **L^2(R):** không gian các hàm có mức năng lượng hữu hạn thoả

$$\int |f(x)|^2 dx < \infty$$

khi đó **W_f** được gọi là biến đổi wavelet của f kết hợp với wavelet ψ

4 **Ghi chú**: Nếu đặt $\psi_{a,b}(t) = \frac{1}{|a|^{1/2}}\psi\left(\frac{t-b}{a}\right)$

Thì ta có thể viết:

$$W_f(a,b) = \int_{-\infty}^{+\infty} f(t)\overline{\psi_{a,b}(t)}\,dt$$
$$= < f, \psi_{a,b} >$$

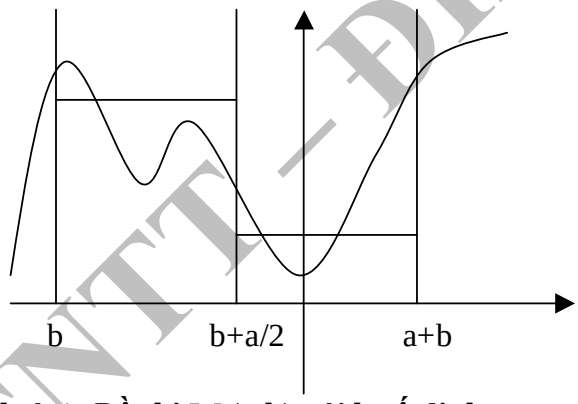
Ví dụ: Cho hàm wavelet Haar với a>0

$$\psi\left(\frac{t-b}{a}\right)=\begin{cases}1 & (b\le x<b+\frac{a}{2})\\ -1 & (b+\frac{a}{2}\le x<b+a)\\ 0 & \text{khác}\end{cases}$$

thì biến đổi wavelet của hàm số f là:

$$W_f(a,b)=\frac{1}{\sqrt{a}}\left(\int_b^{b+a/2} f(x)dx-\int_{b+a/2}^{b+a} f(x)dx\right)$$
$$=\frac{\sqrt{a}}{2}\left(\frac{2}{a}\int_b^{b+a/2} f(x)dx-\frac{2}{a}\int_{b+a/2}^{b+a} f(x)dx\right)$$

Đồ thị của $W_f(a,b)$ với b cố định trên Hình A.1 , giá trị $W_f(a,b)$ là giá trị trung bình của f ở trong khoảng a/2



Hình A.1: Đồ thị $W_f(a,b)$ với b cố định

1.2 Tính chất

Nếu f, g ∈ $L^2(R)$ thì với wavelet ψ ta có

$$\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} W_f(a,b).\overline{W_g(a,b)}.|a|^2\,dadb=C<f,g>$$

trong đó $C=\int_{-\infty}^{+\infty}\frac{|\hat{\psi}(\xi)|^2}{|\xi|}d\xi$

Cho f ∈ $L^2(R)$ và wavelet ψ thoả $C=\int_{-\infty}^{+\infty}\frac{|\hat{\psi}(\xi)|^2}{|\xi|}d\xi<\infty$

Khi đó biến đổi wavelet của f có biến đổi ngược và

$$f = C \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} W_f(a,b).\psi_{a,b} \cdot \frac{1}{a^2}\, da\, db$$

Giả sử Wavelet ψ có tψ ∈ L1(R) và f ∈ L^2(R) bị chặn trên R và liên tục Hölder tại b, nghĩa là ∃α ∈ (0,1] sao cho với mọi t thuộc lân cận của b thì

$$|f(t) - f(b)| \le C|t - b|^{\alpha}$$

khi đó ta có: $|W_f(a,b)| \le C' |a|^{\alpha+\frac{1}{2}}$

Hệ quả: giả sử Wavelet ψ có tψ ∈ $\mathbf{L^1}$. Nếu hàm f ∈ $\mathbf{L^2}$ là Lipschitz trên R thì tồn tại hằng số C sao cho:

$$|W_f(a,b)| \le C' |a|^{\frac{3}{2}}$$ (áp dụng địng lý 3 với α = 1)

2 Biến đổi rời rạc – Phân tích đa phân giải

2.1 Rời rạc hóa biến đổi wavelet

Về phương diện vật lý, phép biến đổi wavelet có tính chất cục bộ về thời gian và tần số của một tín hiệu là rất tốt. Trong mục này, chúng ta sẽ khảo sát một tập con rời rạc của tập {**ψ(a,b) / (a,b) ∈ (R*,R)**} là một cơ sở của L2(R) mà vẫn giữ nguyên được tính cục bộ về thời gian và tần số của họ liên tục các wavelet ban đầu.

Trong biến đổi wavelet chúng ta sử dụng họ các hàm wavelet

$$\psi_{a,b}(t) = \frac{1}{|a|^{\frac{1}{2}}}\psi\left(\frac{t-b}{a}\right)$$

Để thuận tiện cho việc rời rạc hoá ta giới hạn a>0, lúc này ta có:

$$C_\psi = \int_0^{+\infty} \frac{|\hat{\psi}(\xi)|^2}{|\xi|}\, d\xi = \int_{-\infty}^{0} \frac{|\hat{\psi}(\xi)|^2}{|\xi|}\, d\xi < +\infty$$

Từ đây, ta có thể rời rạc hoá hàm wavelet bằng cách chọn $a = a_0^j$, với j ∈ Z và độ nở $a_0 \neq 1$. Để đảm bảo sau khi rời rạc hoá vẫn giữ nguyên tính chất của hàm wavelet liên tục ta chọn $b = nb_0a_0^j$ với b0>0 và n ∈ Z. Tóm lại việc rời rạc hoá được thực hiện như sau:

$a = a_0^j$

$b = nb_0a_0^j$, $\quad a_0 > 1,\ b_0 > 0,\ j,n \in Z$

Lúc này ta được hàm wavelet rời rạc:

$$\psi_{j,n}(x) = a_0^{-j/2}\psi\left(\frac{x - nb_0a_0^j}{a_0^j}\right) = a_0^{-j/2}\psi(a_0^{-j}x - nb_0)$$

Như vậy, với wavelet ψ, biến đổi wavelet rời rạc của một hàm $f \in L^2(R)$ là:

$$W_f\left(\frac{1}{2^j}, \frac{k}{2^j}\right) = 2^{j/2} \int_{-\infty}^{+\infty} f(x).\overline{\psi\left(2^j x - k\right)}dx, \quad \forall j, k \in Z$$

$$\text{trong đó: } a = \frac{1}{2^j}, \text{ và } b = \frac{k}{2^j}$$

$$a_0 = \frac{1}{2}, n = k, b_0 = 1$$

2.2 Phân tích đa phân giải (Multiresolution analysis - MRA) của L2(R)

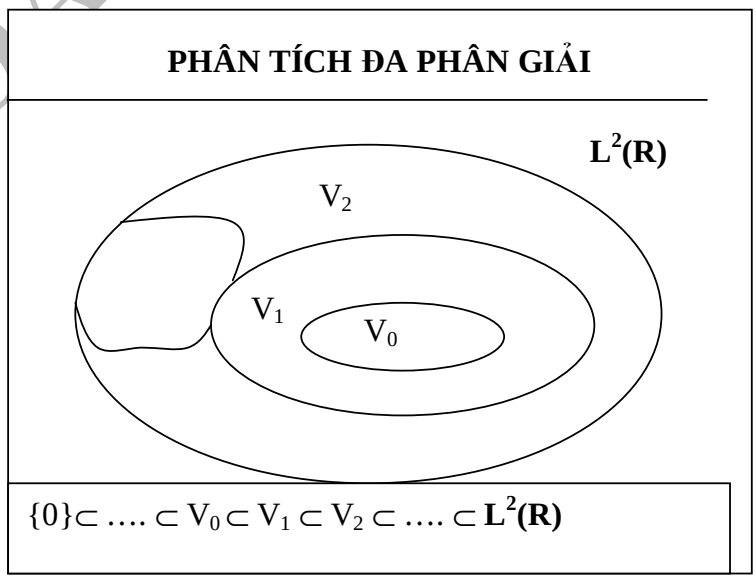
2.2.1 Định nghĩa

Một phân tích đa phân giải MRA của $\mathbf{L^2(R)}$ là dãy các không gian con đóng $\{V_j\}_{j\in Z}$ của $\mathbf{L^2(R)}$ thoả các điều kiện sau:

1. $V_j \subset V_{j+1}\ \forall j \in Z$
2. $\overline{\bigcup_{j\in Z} V_j} = L^2$
3. $\overline{\bigcap_{j\in Z} V_j} = \{0\}$
4. $f(x) \in V_j \Leftrightarrow f(2x) \in V_{j+1},\ \forall j \in Z$
5. $f(x) \in V_0 \Leftrightarrow f(x - n) \in V_0,\ \forall n \in Z$
6. $\exists \varphi \in V_0$ sao cho $\{\phi_{0,n}\}_n$ là một cơ sở trực chuẩn của V_0, với:

$$\phi_{j,k} = 2^{\frac{j}{2}}.(2^j x - k), \forall j, k \in Z$$

Với mọi $j \in Z$, gọi W_j là phần bù trực giao của V_j trong V_{j+1}.
Ta có: $V_{j+1} = V_j \oplus W_j$ và $\forall u \in V_{j+1}$, $u = v + w$ trong đó $v \in V_j$, $w \in W_j$, $\langle v,w\rangle = 0$ và sự phân tích của u là duy nhất.



Hình A.2: Phân tích đa phân giải trong $L^2(R)$

2.2.2 Tính chất

Nếu một dãy các không gian con đóng $\{V_j\}_{j\in Z}$ thoả các điều kiện 1, 2, 3 trong định nghĩa II.2.1 và các không gian con tương ứng W_j là trực giao đôi một thì

$$\bigoplus_{j\in Z} W_j = L^2(R)$$

Nếu một dãy các không gian con đóng $\{V_j\}_{j\in Z}$ là các đa phân giải của **$L^2(R)$** thì tồn tại wavelet $\psi \in$ **$L^2(R)$** sao cho họ:

$$\left\{\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k),\ j,k \in Z\right\}$$

là một cơ sở trực chuẩn của **$L^2(R)$** thoả:

$$P_{j+1} = P_j + \sum_{k\in Z} <\cdot, \psi_{j,k}>\psi_{j,k},\ \forall\ j\in Z$$

2.2.3 Lược đồ phân tích và tái tạo của phân tích đa phân giải

Tổng quát ta có biến đổi wavelet nhanh trong phân tích đa phân giải gồm hai giai đoạn:

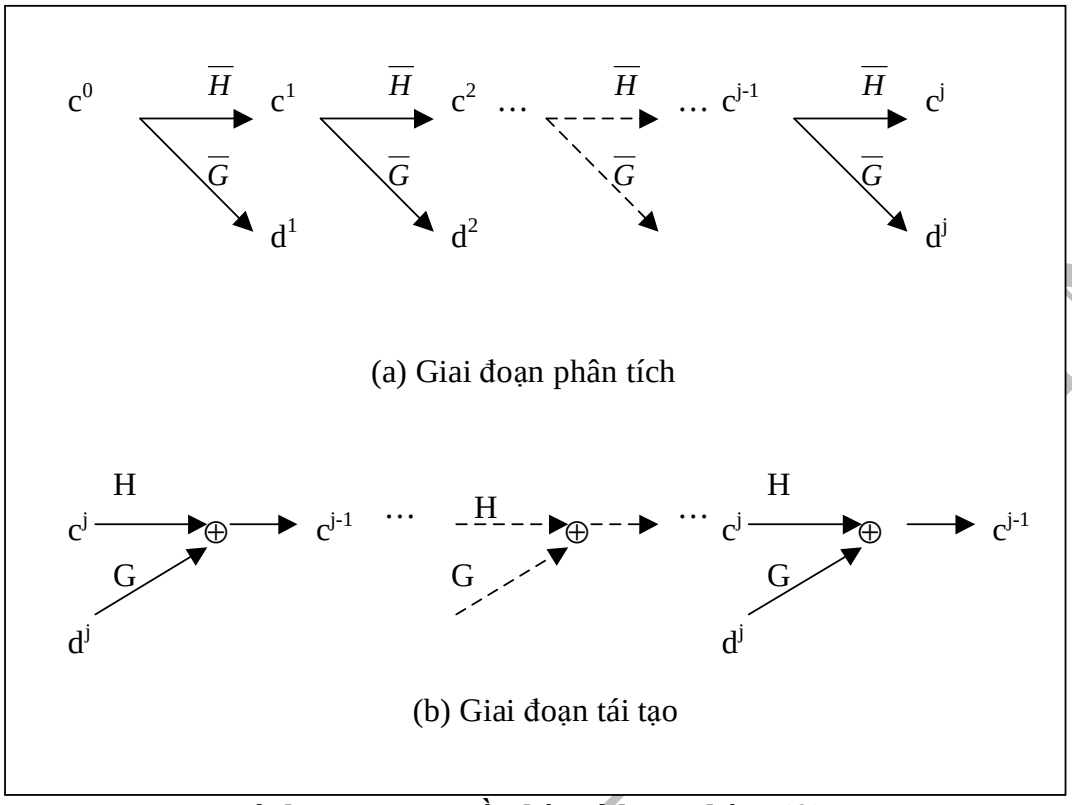
1. Giai đoạn phân tích

$$c^{j+1} = \overline{H}\, c^j$$

$$d^{j+1} = \overline{G}\, c^j$$

2. Giai đoạn tái tạo

$$c^j = H\, c^{j+1} + G\, c^{j+1}$$

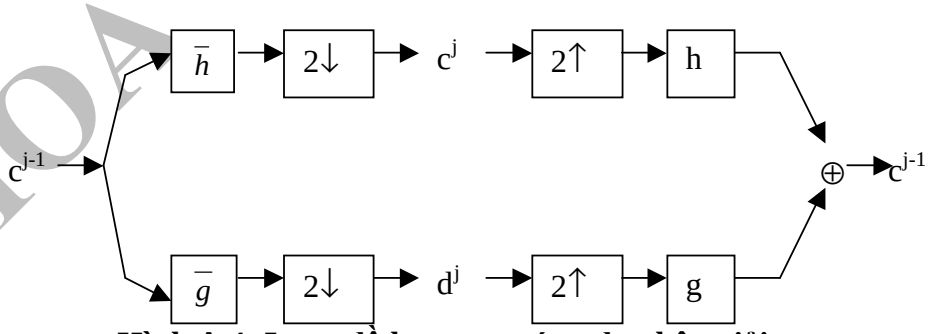


Hình A.3: Lược đồ phân tích đa phân giải

2.2.4 Phân tích đa phân giải với dàn lọc chiếu kính liên hợp

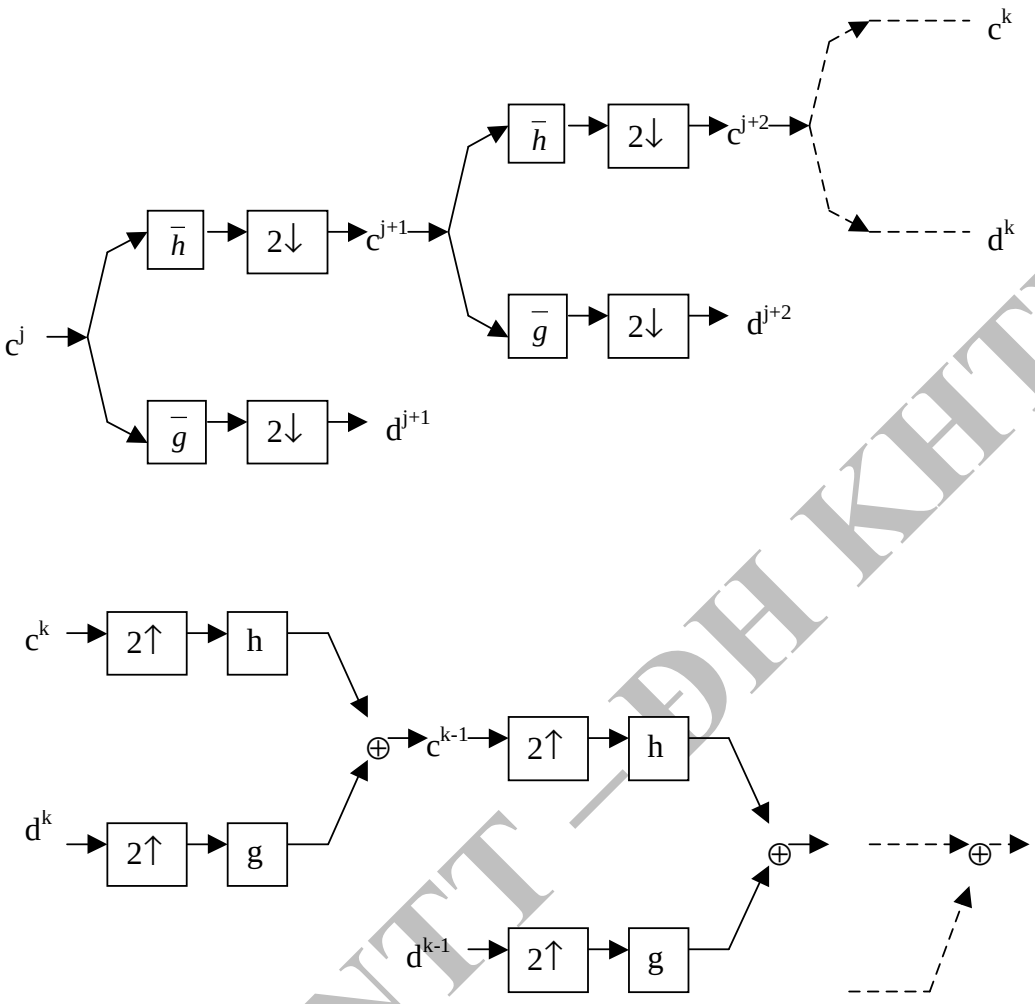
Việc phân tích đa phân giải từ một mức n đến mức kế thô hơn tương ứng với các mức biến đổi wavelet và sau đó được tái tạo lại; qui trình này được mô hình hoá bằng lược đồ lọc trong Hình 4. Đặt :

$(\bar{h})_n = \overline{h_{-n}}$, $(\bar{g})_n = \overline{g_{-n}}$ và có $g_n = (-1)^n h_{-n+1}$



Hình A.4: Lược đồ lọc tương ứng đa phân giải

Tương tự, nếu ứng với mỗi mức lọc ta lại lọc tiếp như trong phân tích đa phân giải ta sẽ được một dàn lọc như lược đồ trong Hình 5.



Hình A.5: Dàn lọc tương ứng phân tích đa phân giải

2.3 Wavelet có giá compact

2.3.1 Wavelet có giá compact của Harr

Cho hàm:

$$\varphi(x) = \begin{cases} 1 & 0 \le x < 1 \\ 0 & x \in R \setminus [0,1) \end{cases}$$

Khi đó ta xây dựng được đa phân giải $\{V_j\}_{j\in Z}$ trong đó:
$V_j = \{f \in L2(R): f = const \text{ trên } [2^j k, 2^j(k+1)]\ \} \ \forall\ k \in Z$

Ta có:

$$\varphi(x) = \varphi(2x) + \varphi(2x-1)$$
$$= \sqrt{2}\left[\frac{1}{\sqrt{2}}(\varphi(2x) + \varphi(2x-1))\right]$$

suy ra: $h_0 = h_1 = \frac{1}{\sqrt{2}}$, $h_k = 0\ \forall\ k \in Z\setminus\{0,1\}$

Từ đó, ta có được wavelet ψ định bởi

$$\psi(x) = \varphi(2x) - \varphi(2x-1) = \begin{cases} 1 & 0 \le x < \frac{1}{2} \\ -1 & \frac{1}{2} \le x < 1 \\ 0 & x \notin [0,1) \end{cases}$$

Tiếp theo ta xây dựng được hệ cơ sở trực chuẩn có giá compact $\{\psi_{j,k}\}j,k \in Z$ trong đó: $\psi_{j,k}(x) = 2^{\frac{j}{2}}\psi\left(2^j x - k\right)$

2.3.2 Wavelet có giá compact của Daubechies

Daubechies đã xây dựng các wavelet bắt đầu bằng hàm m_0 cho số tự nhiên $N \ge 2$. Khi đó, m_0 có dạng:

$$m_0(\xi) = \left(\frac{1+e^{-i\xi}}{2}\right)^N \bullet \lambda(\xi)$$

Trong đó, $\lambda(\xi)$ là một đa thức lượng giác thoả $|\lambda(\xi)|^2 = P\left(\sin^2\frac{\xi}{2}\right)$
với P là một đa thức sao cho:

$$P(y) = \mathrm{P_N}(y) + \mathrm{y}^{\mathrm{N}} R\left(y - \frac{1}{2} \right)$$

$$\text{trong đó}: \mathrm{P_N}(y) = \sum_{k=0}^{N-1} \left(\frac{\mathrm{N} + \mathrm{k} - 1}{\mathrm{k}} \right) y^k$$

và R là một hàm đa thức lẻ sao cho P(y)≥0 ∀y∈ [0,1]

Khi đó m_0 thoả: $|m_0(\xi)|^2 + |m_0(\xi + \pi)|^2 = 1$ và $\mathrm{m}_0(0) = 1$

Từ đó ta định nghĩa:

$$\hat{\varphi}(\xi) = \frac{1}{\sqrt{2\pi}} m_0\left(\frac{\xi}{2}\right) m_0\left(\frac{\xi}{4}\right) \ldots\ldots m_0\left(\frac{\xi}{2^k}\right) \ldots$$

$$\textit{và} \quad \hat{\psi}(\xi) = -\mathrm{e}^{-\frac{i\xi}{2}} \overline{m_0\left(\frac{\xi}{2} + \pi\right)} \hat{\varphi}\left(\frac{\xi}{2}\right)$$

Suy ra: φ và ψ có giá là các đoạn có độ dài bằng 2N -1 :

$$\varphi(x) = \sqrt{2} \sum_{n=0}^{N} h_n \varphi(2\mathrm{x} - \mathrm{n})$$

$$\psi(x) = \sqrt{2} \sum_{n=0}^{N} (-1)^{n-1} h_{-n+1} \varphi(2\mathrm{x} - \mathrm{n})$$

Hàm chuẩn và wavelet Daubechies tương ứng với N được ký hiệu bởi N^{ϕ} và N^{Ψ}.

Phụ lục B: Kết quả thử nghiệm hàm băm Tiger và Whirlpool

Kích thước giá trị băm là như nhau đối với từng phương pháp băm cho dù kích thước file trước khi băm có khác nhau:

- § Đối với Tiger, kích thước giá trị băm là 192 bit
- § Đối với Whirlpool, kích thước giá trị băm là 512 bit

Stt	Tên file	Kích thước trước khi băm (b)	Thời gian băm (ms)	
			Tiger	Whirlpool
1.	3.txt	7.266	435	0
2.	1.txt	231	10	0
3.	2.txt	11.942	831	0
4.	3.pdf	175.487	11576	80
5.	4.pdf	361.518	27940	170
6.	Tk.ppt	786.944	60047	530
7.	Reversed Engineering1.pps	1.075.712	83520	1.222
8.	QLCHPM1.doc	920.576	72745	1.151
9.	Figure1.gif	11.975	811	0
10.	Figure2.gif	2.369	180	0
11.	Figure3.gif	21.836	1.722	10
12.	Html1.htm	38.029	2.794	20
13.	Html2.htm	47.843	3.394	20
14.	Html3.htm	82.519	5.678	50
15.	FileSys.doc	40.448	2.724	20
16.	KetQuaBam.doc	68.608	4.877	30
17.	FZ.zip	179.861	13.780	100
18.	FT.txt	9.615	631	10
19.	CNET.pdf	168.419	13.089	90
20.	CTDL2.pdf	58.923	3.985	30
21.	ADO.ppt	184.832	12.968	100

Bảng B.1. Máy CPU Celeron 533 MHz, SDRAM 256 MB, HDD 24GB, Processor 32bit

Stt	Tên file	Kích thước trước khi băm (b)	Thời gian băm (ms)	
			Tiger	Whirlpool
1	MoHinhBaLop.doc	3021824	79.500	1.200
2	FileSys.doc	40448	1.046	15
3	KetQuaBam.doc	68608	1.812	16
4	Total.zip	22990659	605.047	17.360

5	FZ.zip	179861	4.781	62
6	Sumo.txt	492740	12.609	188
7	FT.txt	9615	281	0
8	OPENGL.pdf	15140771		11.906
9	CNET.pdf	168419	4.421	62
10	CTDL2.pdf	58923	1.547	15
11	OPENGL.ppt	12875776	346.062	5.000
12	ADO.ppt	184832	5.078	78
13	Figure1.gif	11975	312	0
14	Figure2.gif	2369	62	0
15	Figure3.gif	21836	563	0
16	Html1.htm	38029	1.015	15
17	Html2.htm	47843	1.266	15
18	Html3.htm	82519	2.188	31

Bảng B.2. Máy CPU Pentium IV 1.7GHz, DDRAM 128 MB, HDD 80GB, Processor 32bit

Stt	Tên file	Kích thước trước khi băm (b)	Thời gian băm (ms)	
			Tiger	Whirlpool
1	MoHinhBaLop.doc	3021824	213.817	3.515
2	FileSys.doc	40448	2.583	20
3	KetQuaBam.doc	68608	4.606	40
4	FZ.zip	179861	12.628	90
5	FT.txt	9615	731	0
6	CNET.pdf	168419	12.268	90
7	CTDL2.pdf	58923	3.825	40
8	OPENGL.ppt	12875776		
9	ADO.ppt	184832	13.129	121
10	Figure1.gif	11975	781	0
11	Figure2.gif	2369	150	0
12	Figure3.gif	21836	1.712	10
13	Html1.htm	38029	4.947	20
14	Html2.htm	47843	3.254	30
15	Html3.htm	82519	5.568	40

Bảng B.3. Máy CPU Pentium III, SDRAM 128 MB, HDD 80GB, Processor 32bit