HÀM BĂM
HASH FUNCTIONS

Giáo viên: Phạm Nguyên Khang
pnkhang@cit.ctu.edu.vn

Tổng quan

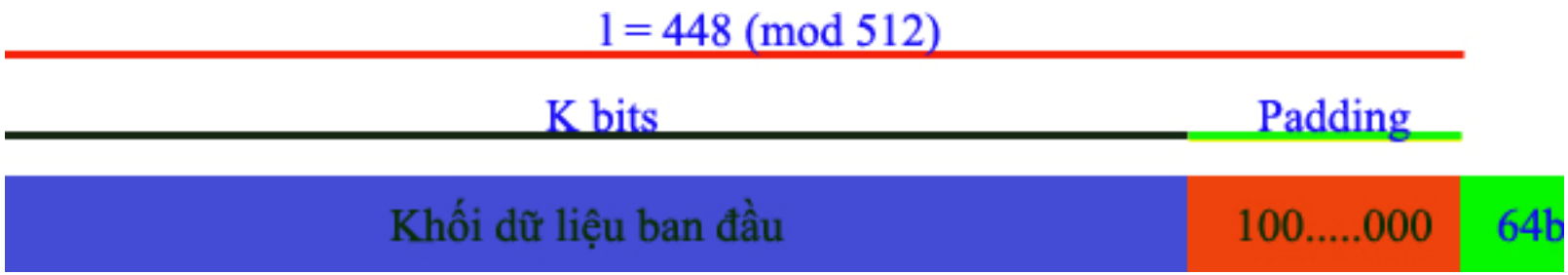
- Mục tiêu: các hàm băm (H) tạo ra bản nhận dạng (fingerprint) cho một tập tin, thông điệp hay một khối dữ liệu truyền đi nhằm kiểm tra tính toàn vẹn.
- Các đặc điểm
  - H có thể được áp dụng trên khối dữ liệu có độ dài bất kỳ
  - H tạo đầu ra có độ dài cố định
  - H(x) tính toán mọi x tương đối dễ dàng, tạo điều kiện cho việc cài đặt trên phần cứng lẫn phần mềm được thiết thực
  - Với bất kỳ giá trị băm h, không thể tính được x sao cho H(x)=h. Hay H được gọi là **hàm một chiều**
  - **Tính bền xung đột yếu (weak collision resistance):** với bất kỳ giá trị x, không thể tính được $y \neq x$ sao cho H(y) = H(x).
  - **Tính bền xung đột mạnh (strong collision resistance):** Không thể tính được một cặp (x, y) sao cho H(x) = H(y)

Giải thuật MD5

- Phát triển bởi Ron Rivest tại đại học MIT
- Input: thông điệp với độ dài bất kỳ
- Output: giá trị băm (message digest) 128 bits
- Giải thuật gồm 5 bước thao tác trên khối 512 bits

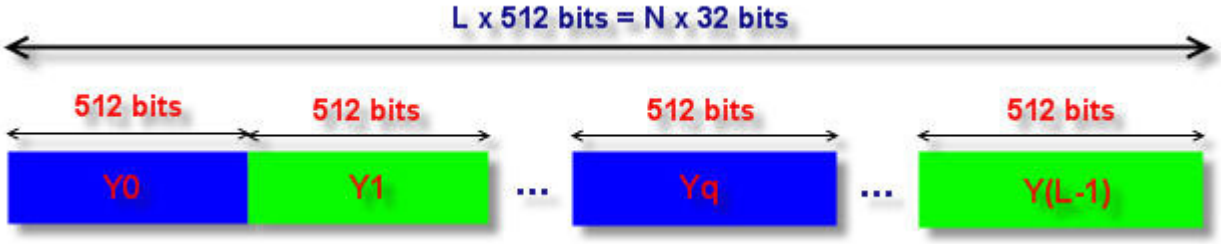
Giải thuật MD5 – Nguyên lý

- Bước 1: **nhồi dữ liệu**
  - Nhồi thêm các bits sao cho dữ liệu có độ dài ***l ≡ 448 mod 512*** hay ***l = n * 512 + 448 (n,l nguyên)***
  - Luôn thực hiện nhồi dữ liệu ngay cả khi dữ liệu ban đầu có độ dài mong muốn. Ví dụ, dữ liệu có độ dài 448 được nhồi thêm 512 bits để được độ dài 960 bits.
  - Số lượng bit nhồi thêm nằm trong khoảng 1 đến 512
  - Các bit được nhồi gồm 1 bit "1" và các bit 0 theo sau.



Giải thuật MD5 – Nguyên lý

- Bước 2: **thêm vào độ dài**
  - Độ dài của khối dữ liệu ban đầu được biểu diễn dưới dạng nhị phân 64-bit và được thêm vào cuối chuỗi nhị phân kết quả của bước 1
  - Nếu độ dài của khối dữ liệu ban đầu > 2^{64}, chỉ 64 bits thấp được sử dụng, nghĩa là giá trị được thêm vào bằng **K mod 2^{64}**
  - Kết quả có được từ 2 bước đầu là một khối dữ liệu có độ dài là bội số của 512. Khối dữ liệu được biểu diễn:
    - Bằng một dãy *L khối* 512-bit $Y_0, Y_1, \ldots, Y_{L-1}$
    - Bằng một dãy *N từ (word)* 32-bit M_0, M_1, M_{N-1}. Vậy **N = L x 16** (32 x 16 = 5



Giải thuật MD5 – Nguyên lý

- Bước 3: **khởi tạo bộ đệm MD** (MD buffer)
  - Một bộ đệm 128-bit được dùng lưu trữ các giá trị băm trung gian và kết quả. Bộ đệm được biểu diễn bằng 4 thanh ghi 32-bit với các giá trị khởi tạo ở dạng little-endian (byte có trọng số nhỏ nhất trong từ nằm ở địa chỉ thấp nhất) như sau:
    - A = 67 45 23 01
    - B = EF CD AB 89
    - C = 98 BA DC FE
    - D = 10 32 54 76
  - Các giá trị này tương đương với các từ 32-bit sau:
    - A = 01 23 45 67
    - B = 89 AB CD EF
    - C = FE DC BA 98
    - D = 76 54 32 10

Giải thuật MD5 – Nguyên lý

- Bước 4: **xử lý các khối dữ liệu 51? bit**
  - Trọng tâm của giải thuật là **hàm nén** (compression function) gồm 4 "vòng" xử lý. Các vòng này có cấu trúc giống nhau nhưng sử dụng cá hàm luận lý khác nhau gồm F, G, H và I
    - $F(X,Y,Z) = X \wedge Y \vee \neg X \wedge Z$
    - $G(X,Y,Z) = X \wedge Z \vee Y \wedge \neg Z$
    - $H(X,Y,Z) = X \text{ xor } Y \text{ xor } Z$
    - $I(X,Y,Z) = Y \text{ xor } (X \vee \neg Z)$
  - Mảng 64 phần tử được tính theo công thức: $T[i] = 2^{32} \times \text{abs}(\sin(i))$, được tính theo radian.
  - Kết quả của 4 vòng được cộng (th modulo 2^{32} với đầu vào CV_q để tạc CV_{q+1}

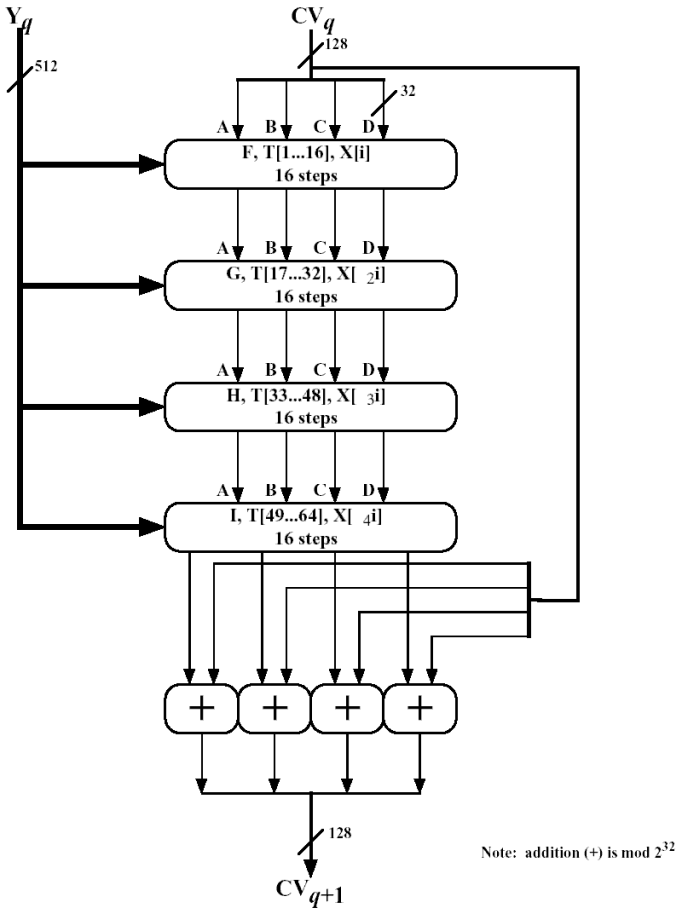


Figure 9.2 MD5 Processing of a Single 512-bit Block (MD5 Compression Function)

Giải thuật MD5 – Nguyên lý

- Các giá trị trong bảng T

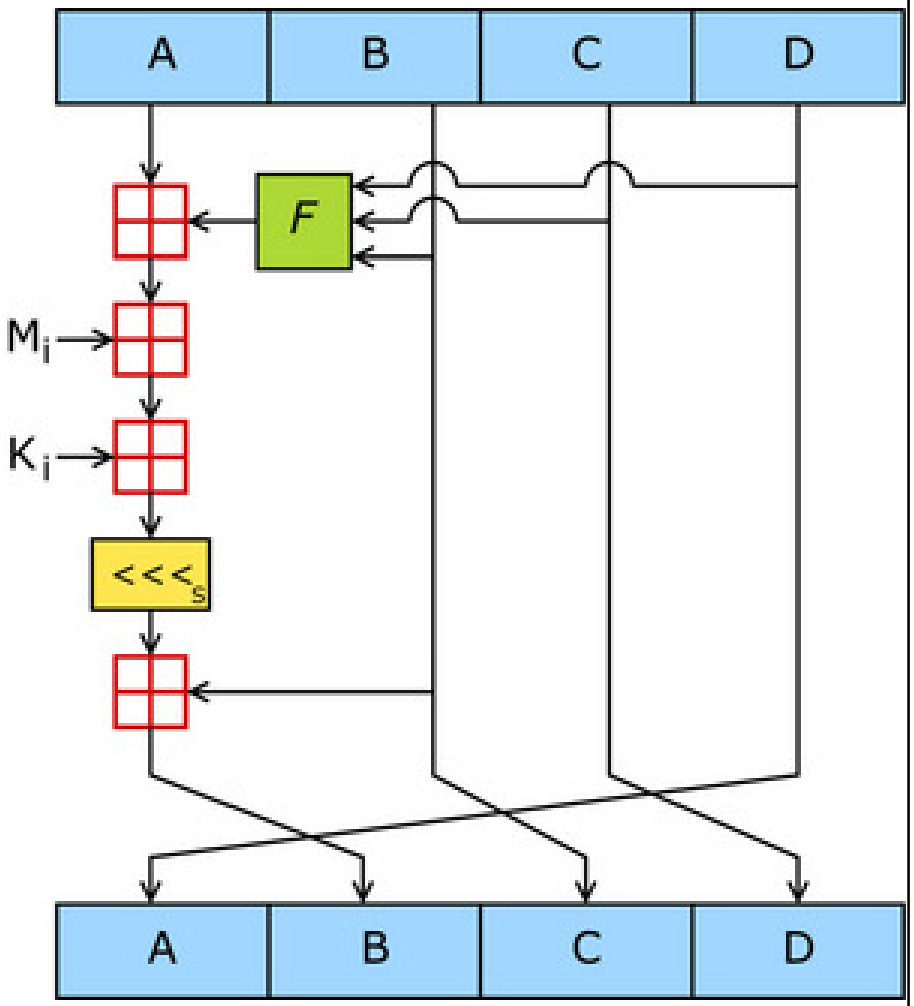
T[1] = d76aa478	T[17] = f61e2562	T[33] = fffa3942	T[49] = f4292244
T[2] = e8c7b756	T[18] = c040b340	T[34] = 8771f681	T[50] = 432aff97
T[3] = 242070db	T[19] = 265e5a51	T[35] = 6d9d6122	T[51] = ab9423a7
T[4] = c1bdceee	T[20] = e9b6c7aa	T[36] = fde5380c	T[52] = fc93a039
T[5] = f57c0faf	T[21] = d62f105d	T[37] = a4beea44	T[53] = 655b59c3
T[6] = 4787c62a	T[22] = 2441453	T[38] = 4bdecfa9	T[54] = 8f0ccc92
T[7] = a8304613	T[23] = d8a1e681	T[39] = f6bb4b60	T[55] = ffeff47d
T[8] = fd469501	T[24] = e7d3fbc8	T[40] = bebfbc70	T[56] = 85845dd1
T[9] = 698098d8	T[25] = 21e1cde6	T[41] = 289b7ec6	T[57] = 6fa87e4f
T[10] = 8b44f7af	T[26] = c33707d6	T[42] = eaa127fa	T[58] = fe2ce6e0
T[11] = ffff5bb1	T[27] = f4d50d87	T[43] = d4ef3085	T[59] = a3014314
T[12] = 895cd7be	T[28] = 455a14ed	T[44] = 4881d05	T[60] = 4e0811a1
T[13] = 6b901122	T[29] = a9e3e905	T[45] = d9d4d039	T[61] = f7537e82
T[14] = fd987193	T[30] = fcefa3f8	T[46] = e6db99e5	T[62] = bd3af235
T[15] = a679438e	T[31] = 676f02d9	T[47] = 1fa27cf8	T[63] = 2ad7d2bb
T[16] = 49b40821	T[32] = 8d2a4c8a	T[48] = c4ac5665	T[64] = eb86d391

Giải thuật MD5 – Nguyên lý

- Bước 5: **Xuất kết quả**
  - Sau khi xử lý hết L khối 512-bit, đầu ra của lần xử lý thứ L là giá trị băm 128 bits.
- Giải thuật MD5 được tóm tắt như sau:
  - $CV_0 = IV$
  - $CV_{q+1} = SUM_{32}[CV_q, RF_I(Y_q, RF_H(Y_q, RF_G(Y_q, RF_F(Y_q, CV_q))))]$
  - $MD = CV_{L-1}$
- Với các tham số
  - IV: bộ đệm gồm 4 thanh ghi ABCD
  - Y_q: khối dữ liệu thứ q gồm 512 bits
  - L: số khối 512-bit sau khi nhồi dữ liệu
  - CV_q: đầu ra của khối thứ q sau khi áp dụng hàm nén
  - RF_x: hàm luận lý sử dụng trong các "vòng" (F,G,H,I)
  - MD: message digest – giá trị băm
  - SUM_{32}: cộng modulo 2^{32}

Giải thuật MD5 – Hàm nén

- Mỗi vòng thực hiện 16 bước, mỗi bước thực hiện các phép toán để cập nhật giá trị buffer ABCD, mỗi bước được mô tả như sau
  - A ← B + ((A + F(B,C,D) + X[k] + T[i]) <<< s)
  - A,B,C,D: các từ của thanh ghi
  - F: một trong các hàm F,G,H,I
  - <<< s : dịch vòng trái s bits
  - M_i ~ X[k]: từ 32-bit thứ k của khối dữ liệu 512 bits.k=1..15
  - K_i ~ T[i]: giá trị thứ i trong bảng T.
  - +: phép toán cộng modulo 2^{32}



Giải thuật SHA-1

- Secure Hash Algorithm (SHA) phát triển bởi National Institute of Standard and Technology (NIST)
- Đầu vào: thông điệp với độ dài tối đa 2^{64} bits
- Đầu ra: giá trị băm (message digest) có độ dài 160 bits
- Giải thuật gồm 5 bước thao tác trên các khối 512 bits

Giải thuật SHA-1 – Nguyên lý

- Bước 1: **nhồi thêm dữ liệu**
  - Thông điệp được nhồi thêm các bits sao cho độ dài ***l ≡ 448 mod 512*** hay ***l = n * 512 + 448 (n,l nguyên)***
  - Thông điệp luôn luôn được nhồi thêm dữ liệu
  - Số bits nhồi thêm nằm trong khoảng 1 đến 512
  - Phần dữ liệu nhồi thêm bao gồm một bit 1 và theo sau là các bit 0
- Bước 2: **thêm vào độ dài**
  - Độ dài của khối dữ liệu ban đầu được biểu diễn dưới dạng nhị phân 64-bit và được thêm vào cuối chuỗi nhị phân kết quả của bước 1
  - Độ dài được biểu diễn dưới dạng nhị phân 64-bit không dấu
  - Kết quả có được từ 2 bước đầu là một khối dữ liệu có độ dài là bội số của 512. Khối dữ liệu được biểu diễn:
    - Bằng một dãy *L khối* 512-bit $Y_0, Y_1, \ldots, Y_{L-1}$
    - Bằng một dãy *N từ (word)* 32-bit M_0, M_1, M_{N-1}. Vậy **N = L x 16**

Giải thuật SHA-1 – Nguyên lý

- Bước 3: **khởi tạo bộ đệm MD** (MD buffer)
  - Một bộ đệm 160-bit được dùng lưu trữ các giá trị băm trung gian và kết quả. Bộ đệm được biểu diễn bằng 5 thanh ghi 32-bit với các giá trị khởi tạo ở dạng big-endian (byte có trọng số lớn nhất trong từ nằm ở địa chỉ thấp nhất) như sau:
    - A = 01 23 45 67
    - B = 89 AB CD EF
    - C = FE DC BA 98
    - D = 76 54 32 10
    - E = C3 D2 E1 F0
  - Các giá trị này tương đương với các từ 32-bit sau:
    - A = 01 23 45 67
    - B = 89 AB CD EF
    - C = FE DC BA 98
    - D = 76 54 32 10
    - E = C3 D2 E1 F0

Giải thuật SHA-1 – Nguyên lý

- Bước 4: **xử lý các khối dữ liệu 512-b**
  - Trọng tâm của giải thuật bao gồm 4 lặp thực hiện tất cả 80 bước.
  - 4 vòng lặp có cấu trúc như nhau, chỉ nhau ở các hàm logic f_1, f_2, f_3, f_4
  - Mỗi vòng có đầu vào gồm khối 512- thời và một bộ đệm 160-bit ABCDE. thao tác sẽ cập nhật giá trị bộ đệm
  - Mỗi bước sử dụng một hằng số K_t (0 79)
    - 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 ($60 \le t \le 79$)
  - Đầu ra của 4 vòng (bước 80) được c đầu ra của bước CV_q để tạo ra CV_{q+1}

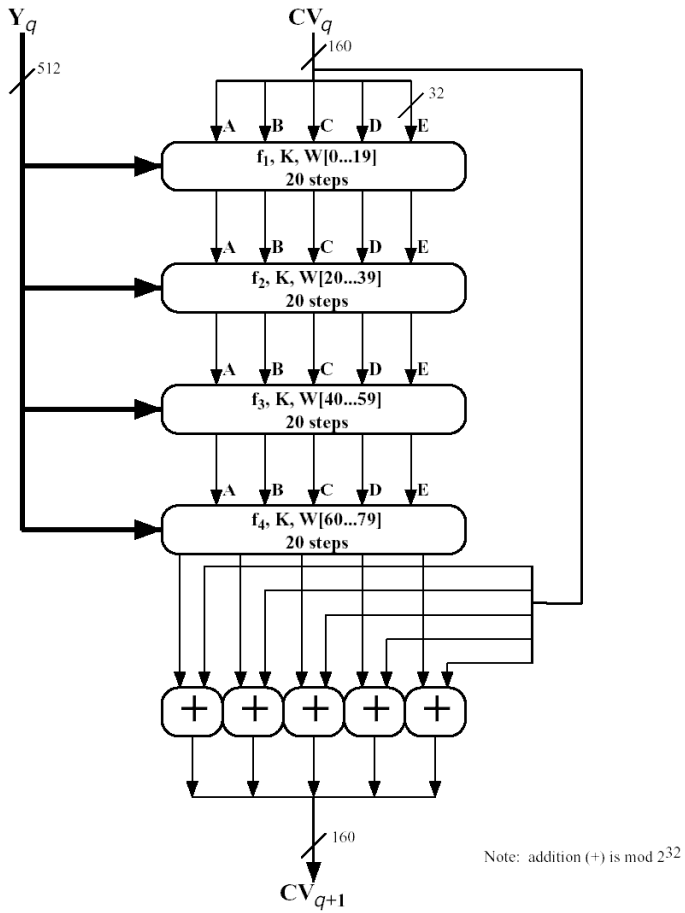


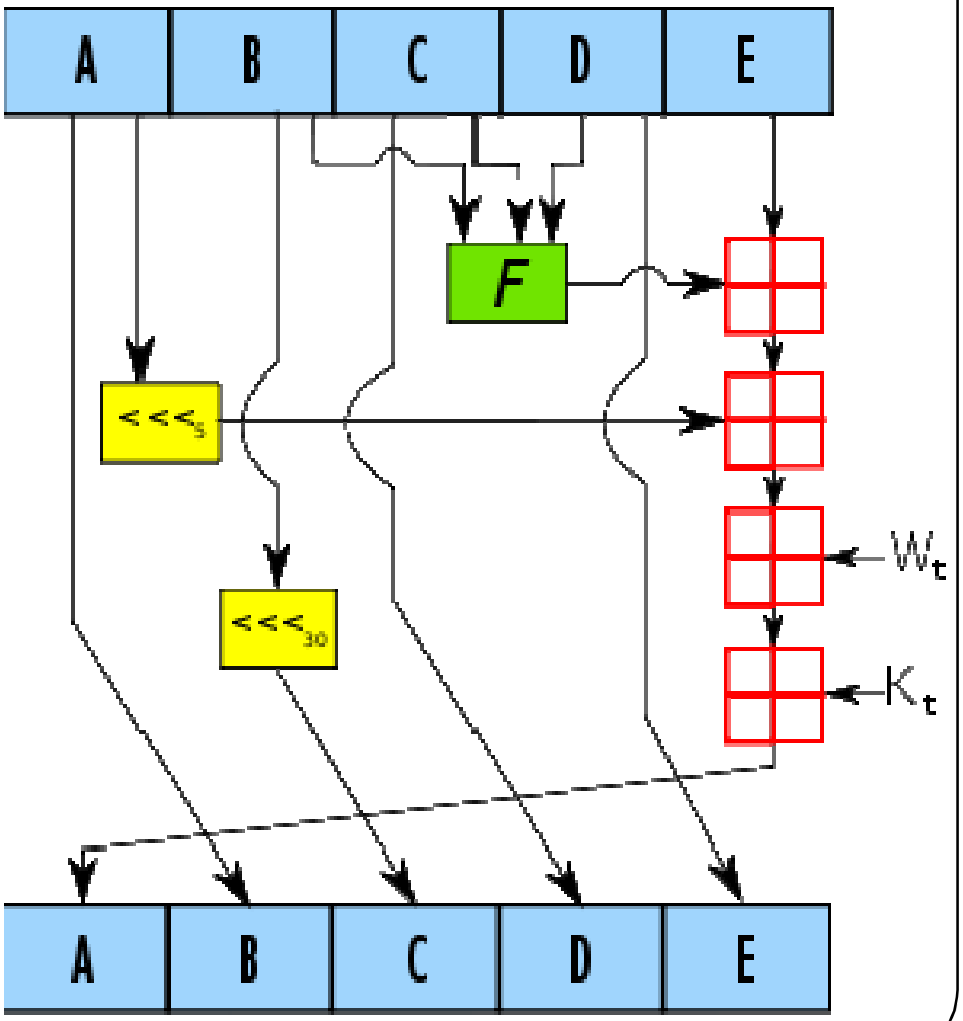
Figure 9.5 SHA-1 Processing of a Single 512-bit Block (SHA-1 Compression Function)

Giải thuật SHA-1 – Nguyên lý

- Bước 5: **xuất kết quả**
  - Sau khi thao tác trên toàn bộ L blocks. Kết quả của khối thứ L là bảng băm 160-bit
  - Giải thuật được tóm tắt như sau:
    - $CV_0 = IV$
    - $CV_{q+1} = SUM_{32}(CV_q, ABCDE_q)$
    - $MD = CV_L$
  - Với
    - IV = giá trị khởi tạo của bộ đệm ABCDE
    - $ABCDE_q$ = đầu ra của hàm nén trên khối thứ q
    - L = số khối 512-bit của thông điệp
    - SUM_{32} = phép cộng modulo 2^{32} trên từng từ (32 bits) của đầu vào
    - MD = giá trị băm

Giải thuật SHA-1 – Hàm nén

- Giải thuật thực hiện tất cả 80 bước, mỗi bước được mô tả như sau:
  - $A \leftarrow E + f(t, B, C, D) + S^5(A) + W_t + K_t$
  - $B \leftarrow A$
  - $C \leftarrow S^{30}(B)$
  - $D \leftarrow C$
  - $E \leftarrow D$
- Trong đó
  - A,B,C,D,E = các từ trong bộ đệm
  - t = số thứ tự của bước
  - F(t,B,C,D) = làm logic tại bước t
  - S^k = dịch vòng trái k bits
  - W_t = từ thứ t của khối dữ liệu
  - K_t = hằng số
  - + = phép cộng modulo 2^{32}



Giải thuật SHA-1 – Hàm nén

- Các hàm f

Bước	Hàm	Giá trị
$(0 \le t \le 19)$	$f_1 = f(t,B,C,D)$	$(B \wedge C) \vee (\neg B \wedge D)$
$(20 \le t \le 39)$	$f_2 = f(t,B,C,D)$	B xor C xor D
$(40 \le t \le 59)$	$f_3 = f(t,B,C,D)$	$(B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
$(60 \le t \le 79)$	$f_4 = f(t,B,C,D)$	B xor C xor D

- Từ 16 từ 32-bit từ khối dữ liệu đầu vào, mở rộng thành 80 từ W_t
  - Với $0 \le t \le 15$, giá trị W_t lấy trực tiếp từ khối dữ liệu
  - Với $t > 15$: $W_t = S^1(W_{t-16}$ xor W_{t-14} xor W_{t-8} xor $W_{t-3})$

So sánh MD5 và SHA-1

- Khả năng chống lại tấn công brute-force:
  - Để tạo ra thông điệp có giá trị băm cho trước, cần 2^{128} thao tác với MD5 và 2^{160} với SHA-1
  - Để tìm 2 thông điệp có cùng giá trị băm, cần 2^{64} thao tác với MD5 và 2^{80} với SHA-1
- Khả năng chống lại thám mã (cryptanalysis): cả 2 đều có cấu trúc tốt
- Tốc độ:
  - Cả hai dựa trên phép toán 32 bit, thực hiện tốt trên các kiến trúc 32 bit
  - SHA-1 thực hiện nhiều hơn 16 bước và thao tác trên thanh ghi 160 bit nên tốt độ thực hiện chậm hơn
- Tính đơn giản: cả hai đều được mô tả đơn giản và dễ dàng cài đặt trên phần cứng và phần mềm

Hàm băm - Ứng dụng

- Key Stretching (tạo khóa bí mật từ mật khẩu)
- Integrity checking (kiểm tra tính toàn vẹn dữ liệu)
- HMAC - Hashed Message Authentication Code (mã chứng thực thông điệp sử dụng hàm băm)
- Chữ ký điện tử

Hàm băm trên JAVA

- Tạo hàm băm:
  - `MessageDigest md;`
  - `md = MessageDigest.getInstance(“MD5”);`
- Băm (“Digestion”) dữ liệu:
  - `byte[] data1, data2, result;`
  - `md.update(data1);`
  - `result = md.digest(data2);`

Mật mã hóa dựa trên mật khẩu (PBE)

- Khóa của DES:
  - Chiều dài 56 bits (trong thực tế cài đặt cần 64 bits)
  - Phức tạp, Khó nhớ

➔ Sử dụng mật khẩu (password)

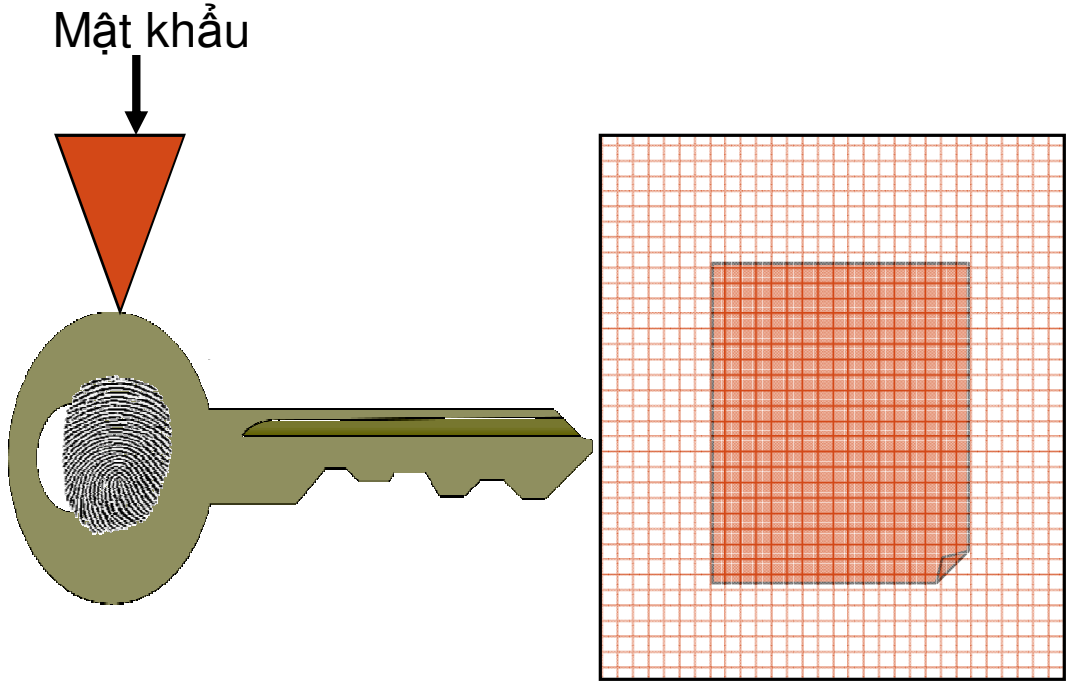
- Chiều dài thay đổi, không phải lúc nào cũng có 64 bits (hay 8 ký tự)

➔ Mật mã hóa dựa trên mật khẩu

- Băm mật khẩu có kích thước bất kỳ thành khóa có đúng 64 bits.

Mật mã hóa dựa trên mật khẩu (PBE)

- Password Based Encryption
  - Kết hợp một **hàm băm** và một giải thuật **mã hóa đối xứng** để mật hóa dữ liệu.



Mật mã hóa dựa trên mật khẩu (PBE)

- PBE hoạt động dựa trên cơ chế các hàm băm.
- Đầu vào:
  - Một *password*
  - Một giá trị *salt* (ngẫu nhiên)
  - Số lần lặp *iteration*
- Giải thuật :

```
key = hash(password + salt)
for 1 to iteration – 1 do
   key = hash(key + salt)
```

- *Chú ý:* phép + ở đây là phép kết nối.

Mã hóa dựa trên mật khẩu (PBE)

- Tạo PBE
  - new Cipher("PBEWith<hàm băm>And<mã hóa>")
- Ví dụ:

```
//Dữ liệu dùng để tạo khóa
byte[] salt = {12, 13, 14, 15, 16, 17, 18, 19};
int nb_iter = 10; //số vòng lặp
specParam = new PBEParameterSpec(salt, nb_iter);
//Tạo khóa bí mật từ mật khẩu
specKeyPBE = new PBEKeySpec(read_password());
factor = SecretKeyFactory.getInstance("PBEWithMD5AndDES");
SecretKey key = factor.generateSecret(specKeyPBE);
//Tạo và khởi động bộ mã hóa
Cipher cipher;
cipher = Cipher.getInstance("PBEWithMD5AndDES");
Cipher.init(Cipher.ENCRYPT_MODE, key, specParam);
//Mã hóa
byte[] plain = "bảng rõ".getBytes();
byte[] encrypted = cipher.doFinal(plain);
```

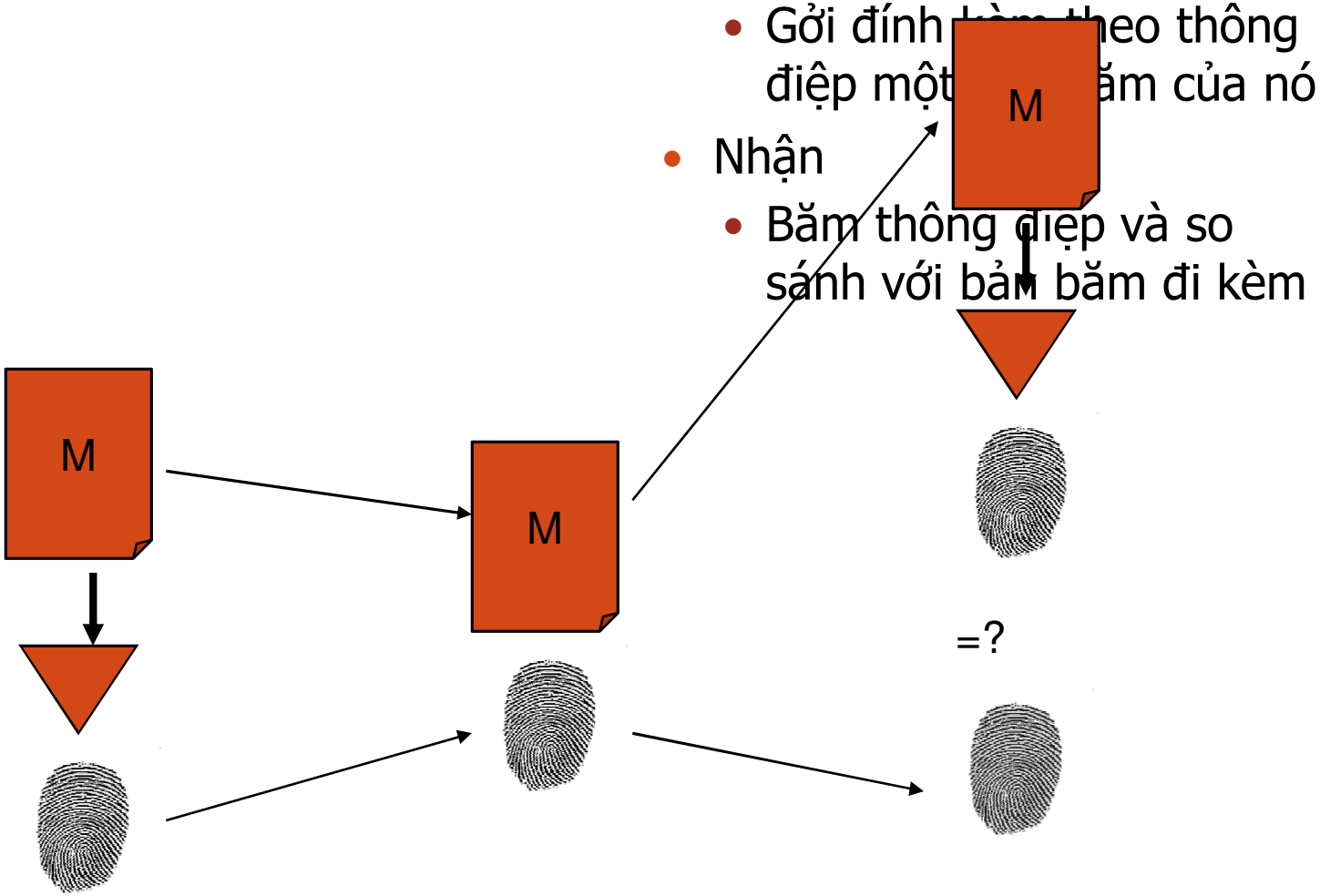
Toàn vẹn dữ liệu (Integrity)

Chuyển $100 cho TK 123-456-789

Chuyển $5000 cho TK 123-456-789

Tòan vẹn dữ liệu (Integrity)

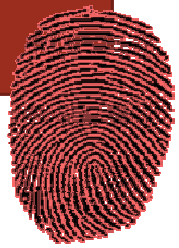
- Gởi
  - Gởi đính kèm theo thông điệp một băm của nó
- Nhận
  - Băm thông điệp và so sánh với bảm băm đi kèm



Toàn vẹn dữ liệu (Integrity)

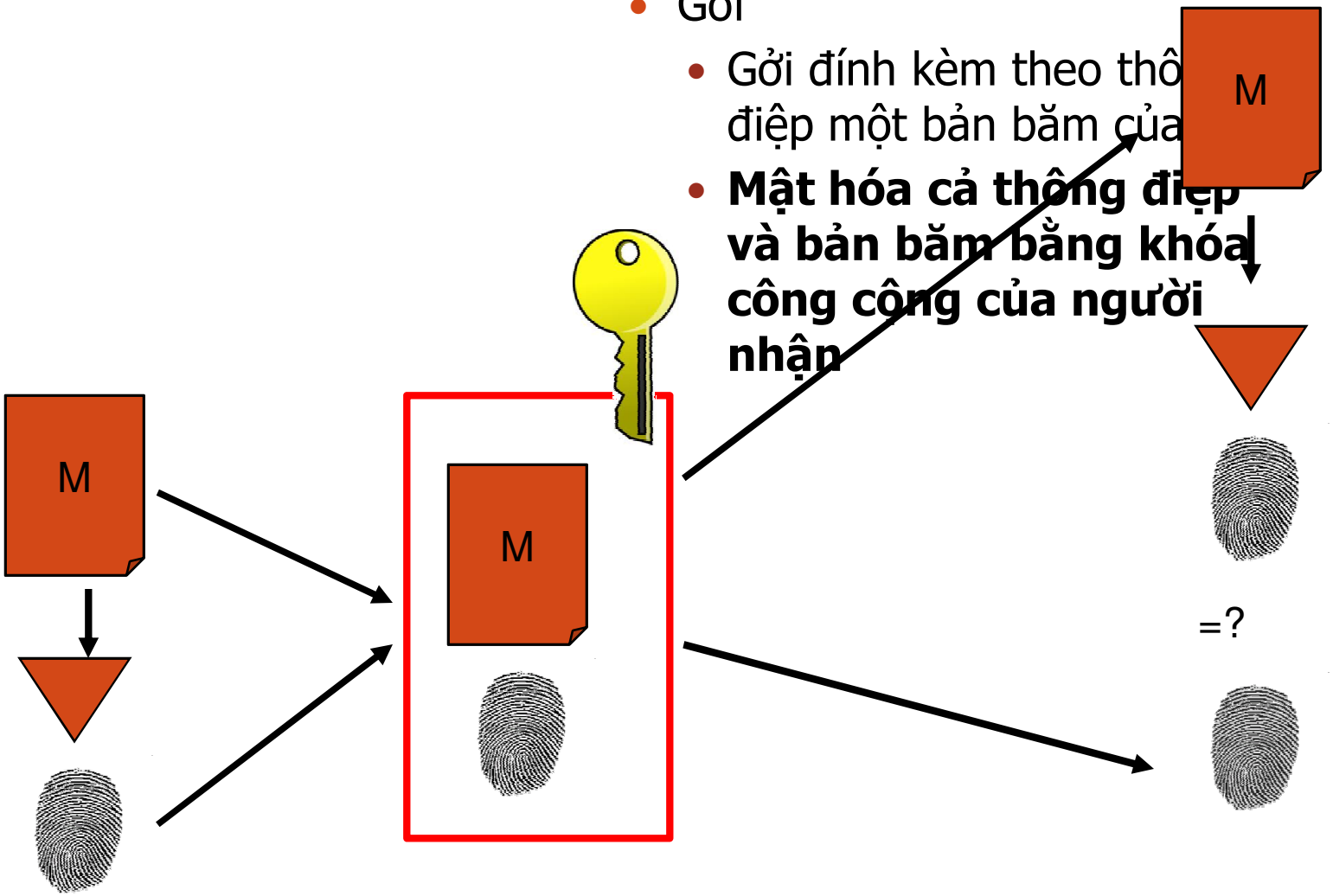
Chuyển $100 cho TK 123-456-789

Chuyển $5000 cho TK 123-456-789



Toàn vẹn dữ liệu (Integrity)

- Gởi
  - Gởi đính kèm theo thô
    điệp một bản băm của
  - **Mật hóa cả thông điệp và bản băm bằng khóa công cộng của người nhận**

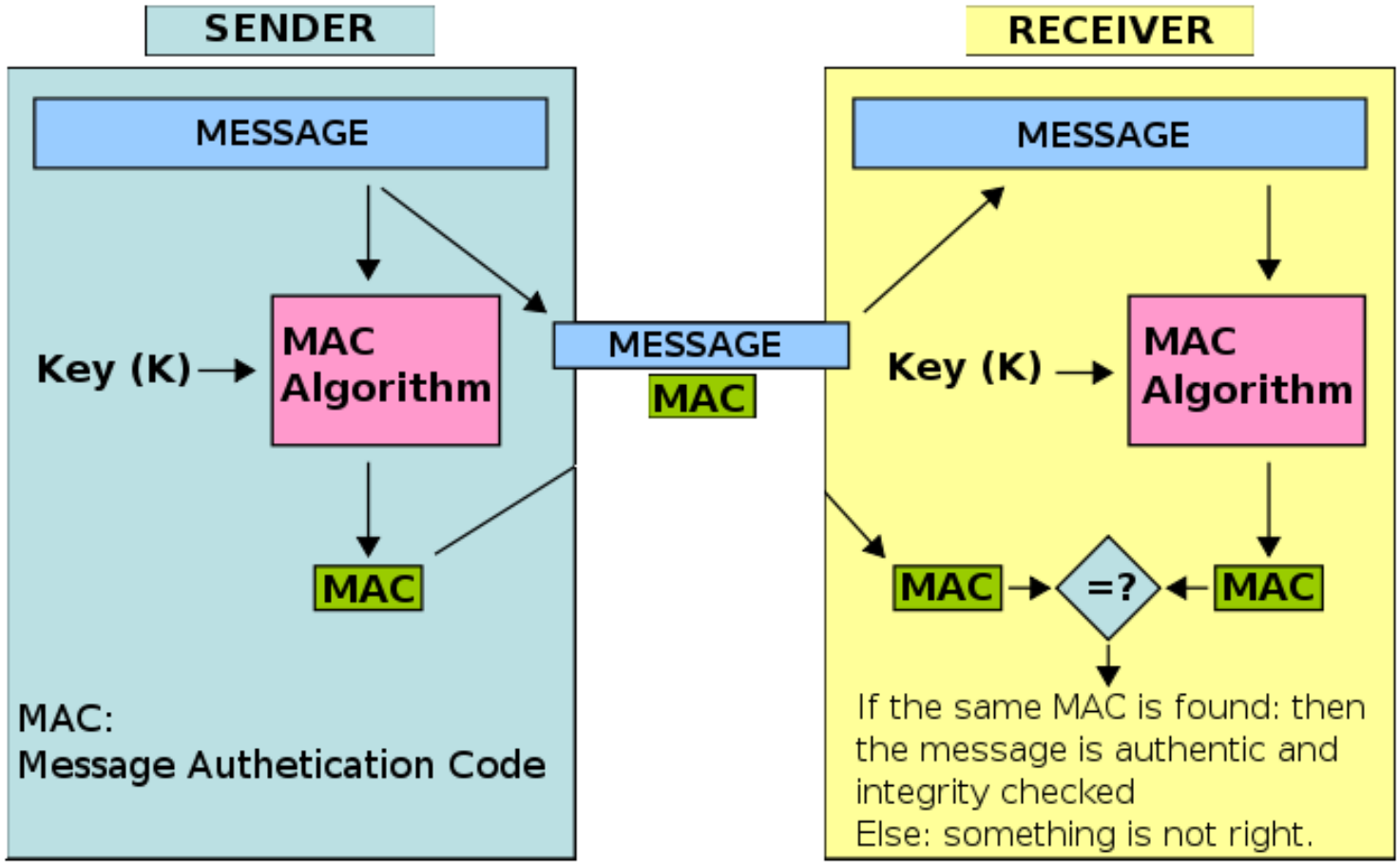


Toàn vẹn dữ liệu (Integrity)

Chuyển $100 cho TK 123-456-789

Chuyển $5000 cho TK 123-456-789

Mã chứng thực thông điệp

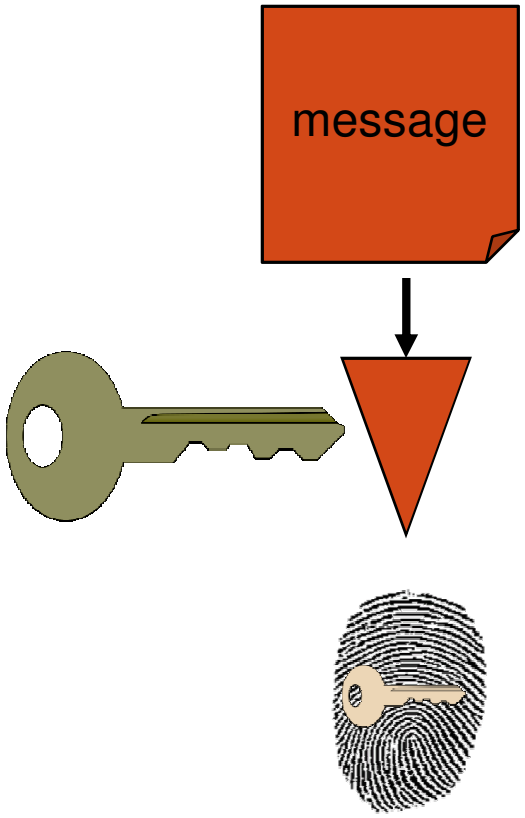


Cả hai sử dụng chung khóa bí mật K

Message Authentication Code - MAC

- MAC = hàm băm hoạt động với một khóa bí mật

- Giải thuật
  - RIPE-MAC
  - IBC-Hash
  - Hmac-MD5
  - Hmac-SHA1



HMAC – giải thuật

- $HMAC_k(M) = H[(K^+ \text{ xor opad}) \,||\, H[(K^+ \text{ xor ipad}) \,||\, M]]$
- Với
  - H = hàm băm bất kỳ
  - M = thông điệp đầu vào
  - K = khóa bí mật
  - K^+ = khóa được thêm các bit 0 vào bên trái để được khối 512-bit
  - ipad = 00110110
  - opad = 01011100

Lớp MAC

- Tạo một đối tượng MAC
  - MAC mac = MAC.getInstance("Hmac-SHA1");
- //Tạo khóa
  - SecretKey key;
  - byte[] data_key = …dữ liệu để tạo khóa
  - Key = new SecretKeySpec(data_key, "Hmac-SHA1");
  - byte[] data = …dữ liệu muốn mã hóa
  - mac.init(key);
- //Băm
  - mac.update(data);
  - byte[] result = mac.doFinal();

Chữ ký điện tử

- Chữ ký điện tử (Digital signature) đảm bảo
  - Authentication
  - Integrity
  - Non-repudiation
- Nguyên lý, Alice muốn "ký" vào thông điệp m của mình, cô ta sẽ:
  - mã hóa m bằng khóa bí mật của mình để nhận được bản mã $\{m\}_{K\_private}$
  - Thông điệp <m, $\{m\}_{K\_private}$ > là thông điệp m được ký bởi Alice.
- Chữ ký điện tử $\{m\}_{K\_private}$ chỉ có thể được tính toán bởi Alice vì cô ta sử dụng khóa bí mật của mình, vì vậy sẽ đảm bảo được:
  - Chứng thực (authentication): thông điệp này được Alice ký
  - Không từ chối trách nhiệm (Non-repudiation): Alice không thể chối bỏ việc ký này vì ngoài cô ta không ai có thể "ký" được "chữ ký" này.

Chữ ký điện tử

- Kiểm tra chữ ký điện tử
  - Khi nhận được thông điệp <m, $\{m\}_{K\_private}$> được Alice ký
  - Ta giải mã chữ ký $\{m\}_{K\_private}$ bằng khóa công khai của Alice để nhận được bản rõ m′.
  - So sánh m với m′. Nếu m ≡ m′ thì thông điệp này chính là của Alice và nó không bị sửa đổi. Tính chất này đảm bảo tính toàn vẹn dữ liệu (integrity) của việc gởi-nhận dữ liệu.
- Tối ưu hóa chữ ký điện tử:
  - Để tránh mã hóa toàn bộ thông điệp m bằng khóa bí mật (thường mất thời gian), ta chỉ mã hóa hàm băm của nó H(m) để nhận được $\{H(m)\}_{K\_private}$.