*Chủ đề 8:*

# Hàm băm mật mã
# Hash & MAC

PGS.TS. Trần Minh Triết


KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

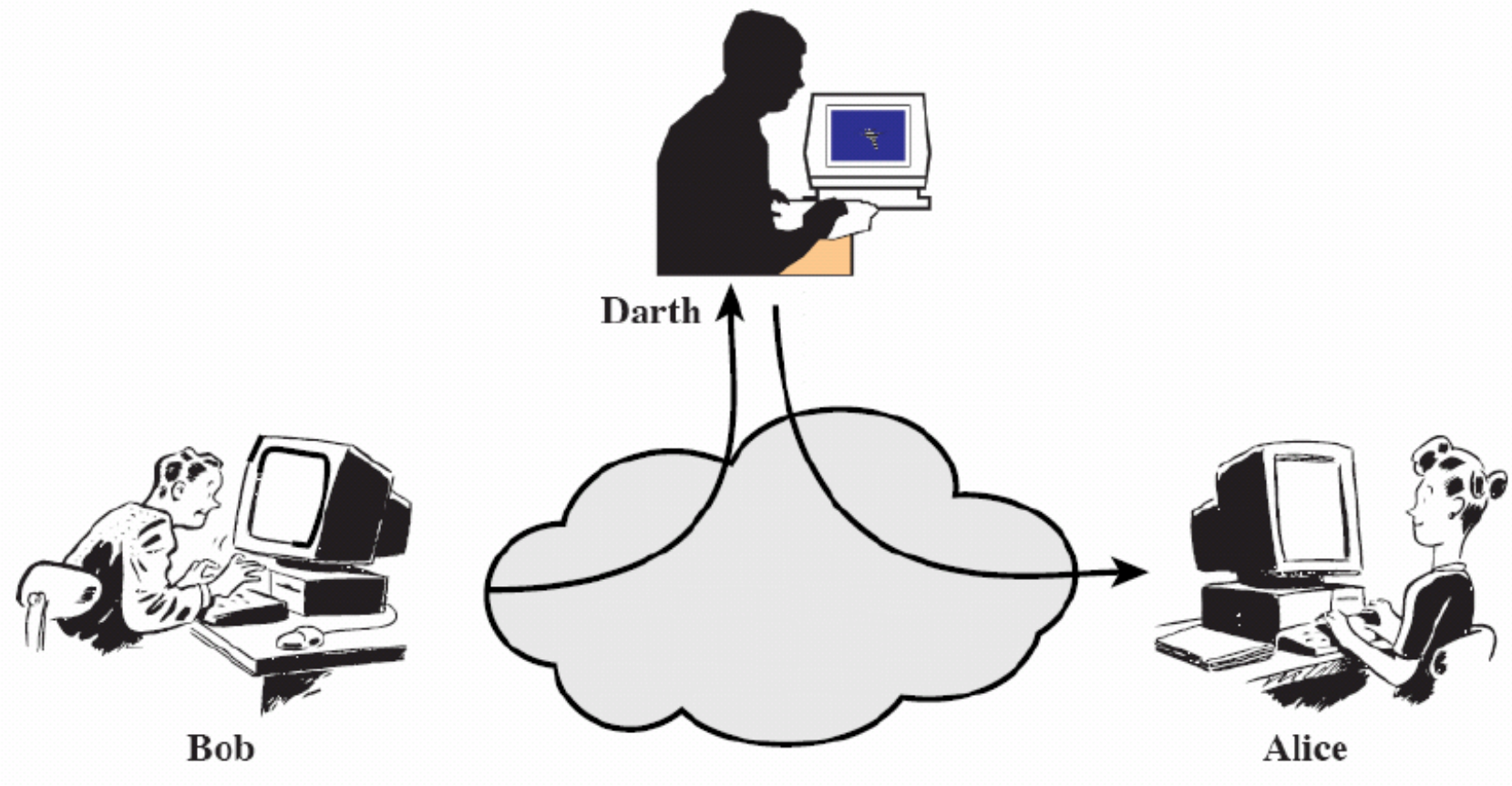
# Nội dung

- Mở đầu
  - Đặt vấn đề
  - Một số tính chất và yêu cầu an toàn của hàm băm
  - Phân loại
- Kiến trúc hàm băm
  - Kiến trúc Merkle-Damgård
  - Kiến trúc Matyas-Meyer-Oseas
  - Kiến trúc Davies-Meyer
  - Kiến trúc Miyaguchi-Preneel
- Một số hàm băm cụ thể: MD4, MD5, SHA1
- MAC

# Nội dung

- Sử dụng hàm băm mật mã trong chữ ký điện tử
  - Sử dụng tạo chữ ký điện tử (đính kèm)
  - Sử dụng tạo chữ ký điện tử (khôi phục được thông tin)
- Sử dụng hàm băm mật mã trong mã hóa bất đối xứng
  - OAEP
- Một số ứng dụng thực tế của hàm băm
  - Chứng nhận thông tin
  - Chứng thực người dùng
  - Liên lạc an toàn
  - Email
  - Các ứng dụng khác…

# Mở đầu

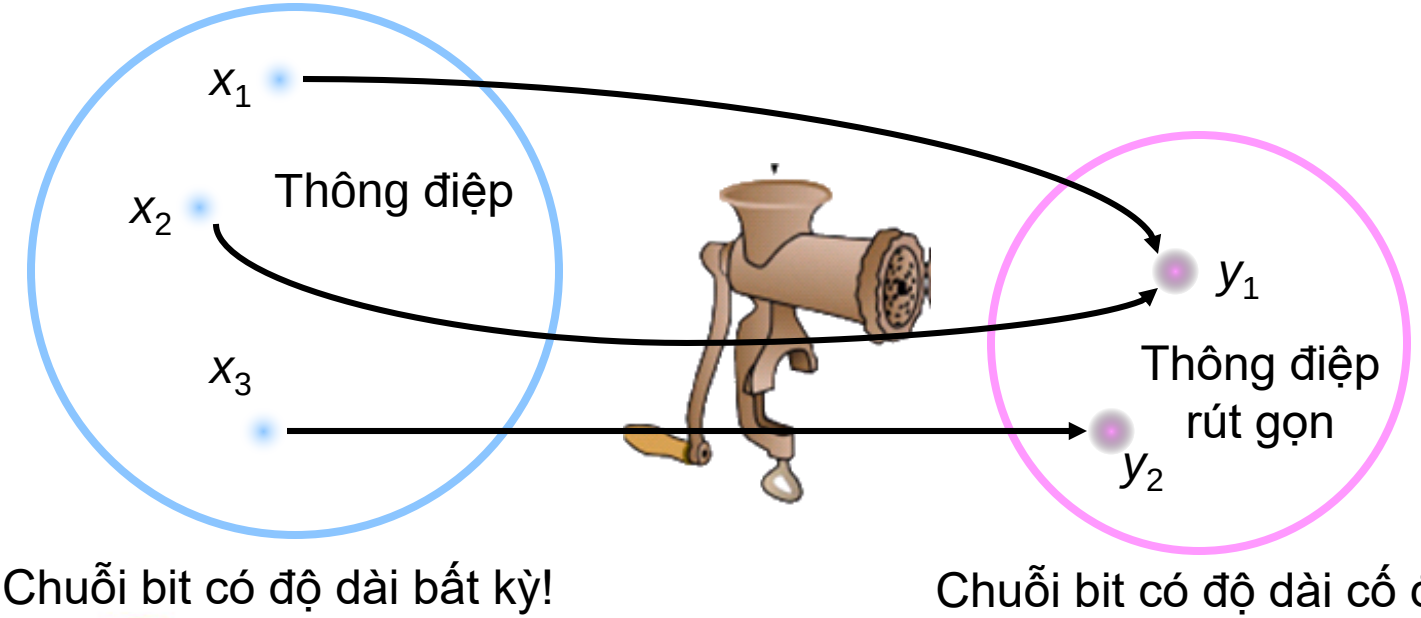


# Tính toàn vẹn và tính bí mật

- Tính toàn vẹn (Integrity): người tấn công không thể can thiệp để sửa nội dung thông điệp
- Mã hóa chỉ nhằm đảm bảo tính bí mật, không giúp đảm bảo tính toàn vẹn thông tin
- ➔ Người tấn công có thể sửa đổi nội dung thông điệp đã được mã hóa mà không cần biết nội dung thật sự của thông điệp
- Ví dụ:
  - Trong đấu giá trực tuyến, có thể thay đổi giá đặt của đối thủ mà không cần biết nội dung thật sự của giá đặt

# Ý tưởng chính của hàm băm mật mã

- H là hàm nén mất thông tin (lossy compression function)
- Hiện tượng đụng độ (Collision): $H(x)=H(x')$ với $x \neq x'$
- Kết quả của việc băm "nhìn có vẻ ngẫu nhiên"



# Hàm băm mật mã *H*

- *H* có thể áp dụng trên dữ liệu có kích thước bất kỳ
- Kết quả của *H* là một chuỗi *n*-bit (*n* có định)
- Dễ dàng tính giá trị *H*(*x*) với *x* bất kỳ
- *H* là hàm một chiều
- *H* an toàn đối với hiện tượng “đụng độ”

# Tính "một chiều"

- Hàm $H$ rất khó bị biến đổi ngược
  - Cho trước chuỗi bit ngẫu nhiên $y \in \{0,1\}^n$, rất khó tìm ra được chuỗi bit $x$ sao cho $H(x)=y$
- Ví dụ:
  - Brute-force: Với mỗi giá trị $x$, kiểm tra $H(x)=y$
  - SHA-1 cho kết quả là chuỗi gồm 160-bit
    - Giả sử phần cứng cho phép thực hiện $2^{34}$ phép thử trong một giây
    - Có thể thực hiện $2^{59}$ phép thử trong một năm
    - Cần $2^{101}$ (~ $10^{30}$) năm để biến đổi ngược SHA-1 với giá trị ngẫu nhiên $y$ cho trước

# Tính an toàn đối với hiện tượng đụng độ

- Rất khó có thể tìm được $x$, $x'$ sao cho $H(x)=H(x')$
- Tìm kiếm đụng độ bằng Brute-force chỉ cần $O(2^{n/2})$, không phải $O(2^n)$
- Birthday paradox
  - Cho t giá trị $x_i$ và giá trị tương ứng $y_i=h(x_i)$
  - Với mỗi cặp $x_i$,$x_j$, xác suất đụng độ là $1/2^n$
  - Tổng số cặp $C^2_t=t(t-1)/2 \sim O(t^2)$
  - Nếu $t$ xấp xỉ $2^{n/2}$, số lượng cặp xấp xỉ $2^n$
  - Với mỗi cặp, xác suất xảy ra đụng độ là $1/2^n$, do đó, xác suất tìm được một cặp giá trị đụng độ rất gần 1

# Birthday Paradox

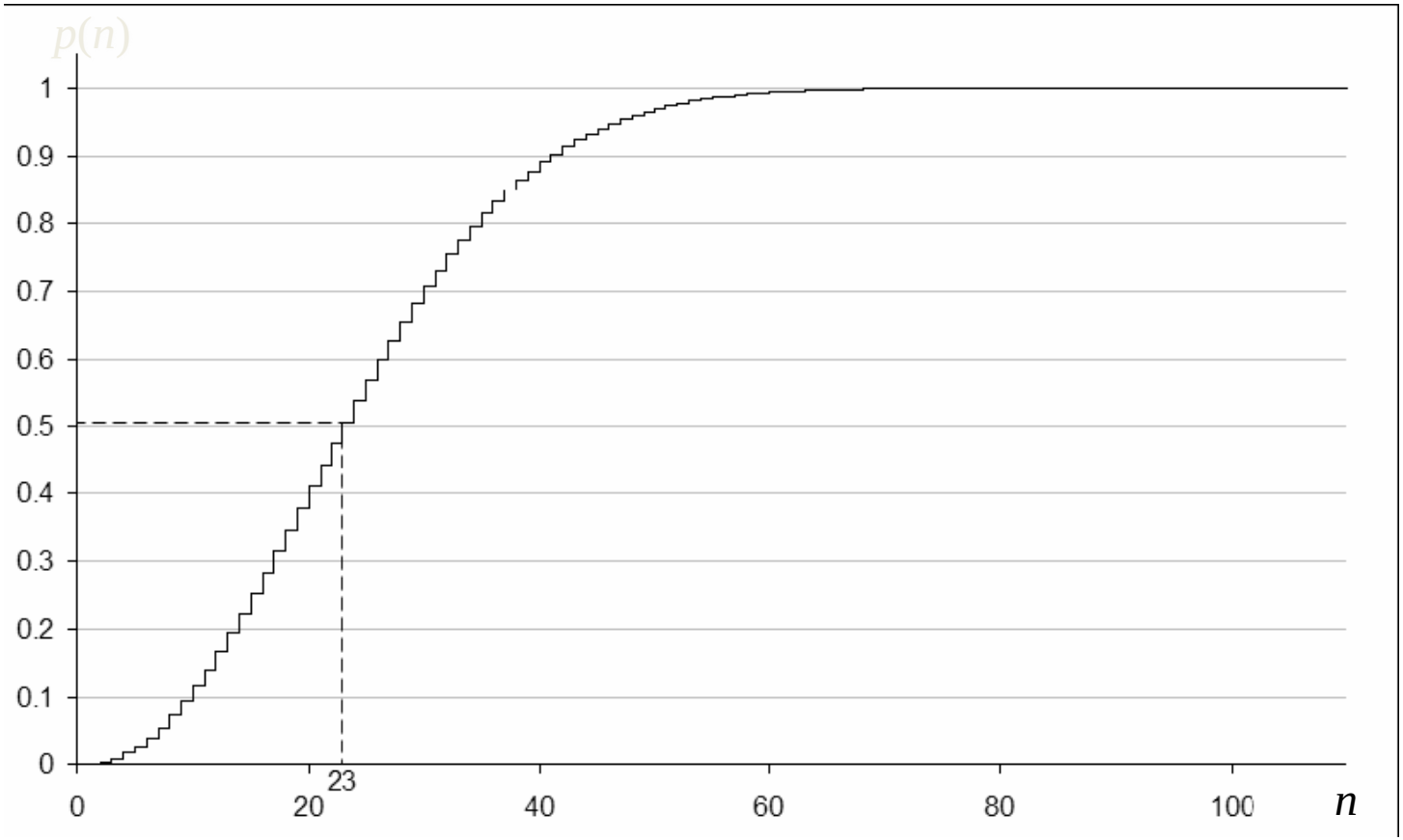
- Ví dụ:
  - Gọi $p(n)$ là xác suất tìm được 2 người có cùng ngày sinh trong nhóm $n$ người
  - Gọi $\overline{p}(n)$ là xác suất 2 người bất kỳ trong nhóm $n$ người đều có ngày sinh khác nhau.

$$p(n) + \overline{p}(n) = 1$$

  - Với $n \leq 365$, ta có

$$\overline{p}(n) = 1\left(1-\frac{1}{365}\right)\left(1-\frac{2}{365}\right)\ldots\left(1-\frac{n-1}{365}\right) = \frac{365!}{365^n(365-n)!}$$

# Birthday Paradox



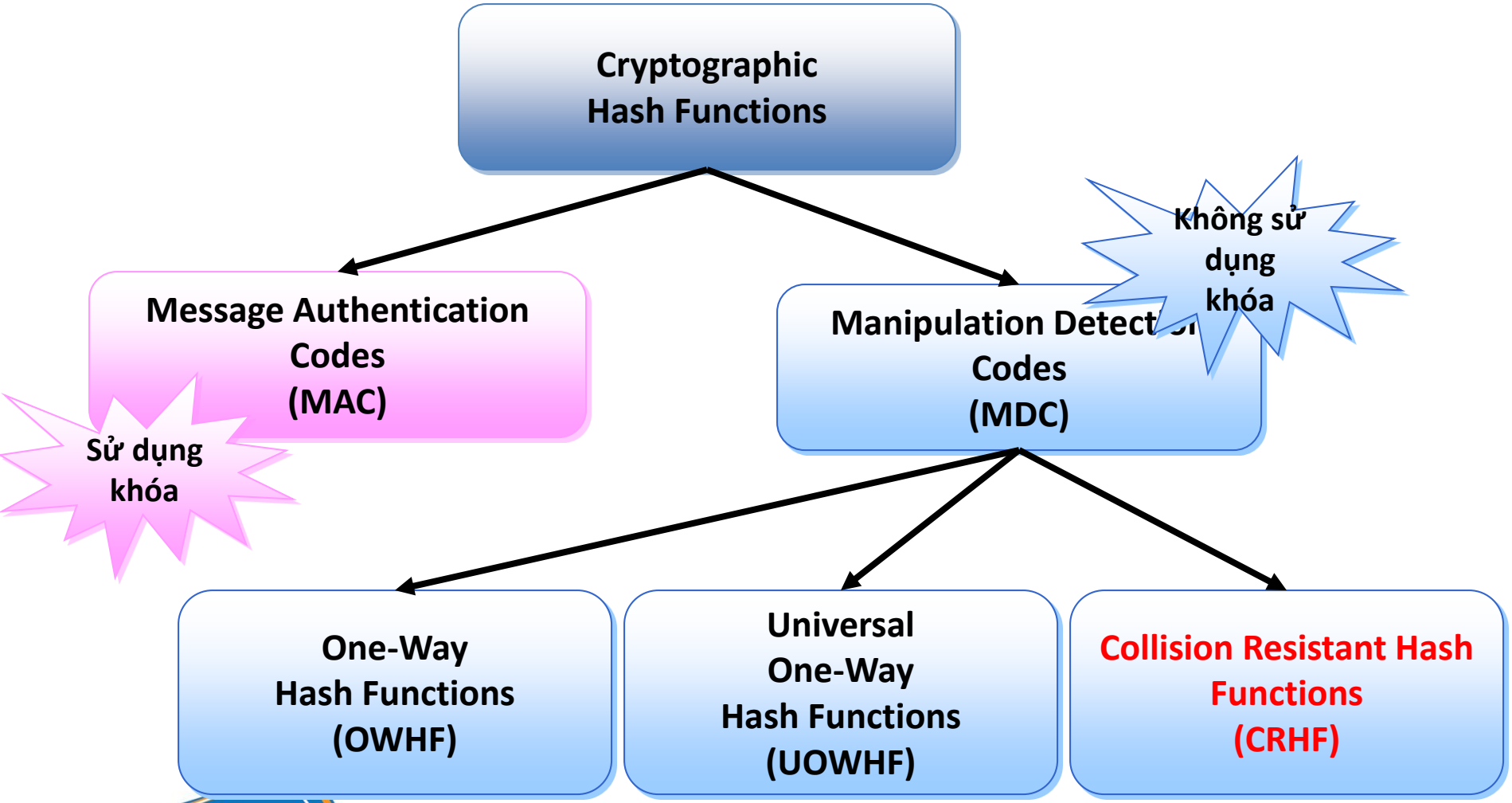
# An toàn với hiện tượng đụng độ "yếu"

- **Weak Collision Resistance**
- Cho dãy bit *x* chọn trước ngẫu nhiên, rất khó tìm được *x*'sao cho *H*(*x*)=*H*(*x*')
- Người tấn công phải tìm được giá trị đụng độ với giá trị *x* cụ thể cho trước. Điều này khó hơn  việc tìm và chỉ ra một cặp giá trị *x* và *x*' đụng độ với nhau.
- Tấn công Brute-force: $O(2^n)$
- **Nhận xét:** An toàn với hiện tượng đụng độ "yếu" không đảm bảo an toàn với hiện tượng đụng độ

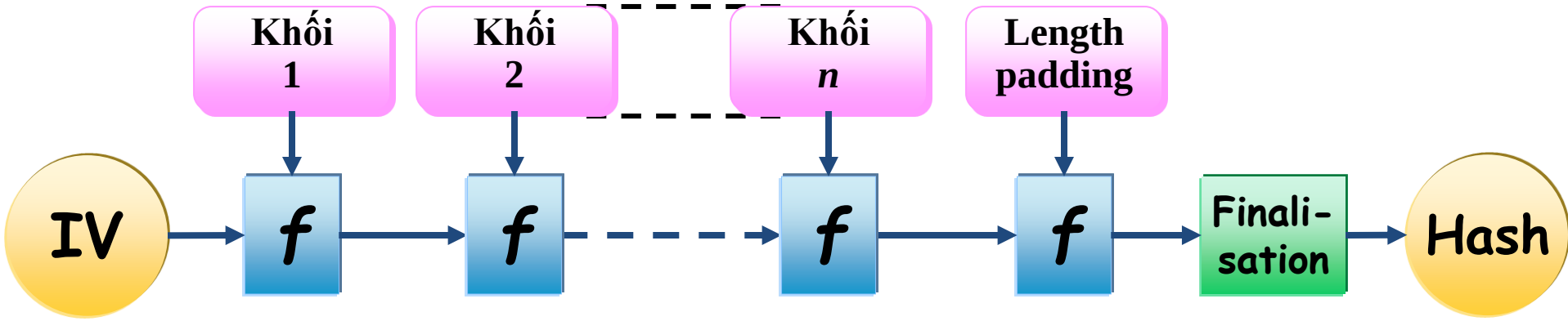
# Tính chất của hàm băm

- An toàn đối với tấn công "tiền ảnh"
  - Preimage resistance
  - cho trước *y*, rất khó tìm được giá trị x sao cho $H(x)=y$
- An toàn đối với tấn công "tiền ảnh thứ 2"
  - $2^{nd}$ preimage resistance
  - cho trước *x* và $y=H(x)$, rất khó tìm được giá trị $x' \neq x$ sao cho $H(x')=H(x)$
- An toàn đối với hiện tượng đụng độ:
  - rất khó tìm được hai giá trị phân biệt *x* và *x'* sao cho $H(x')=H(x)$

# Phân loại hàm băm mật mã

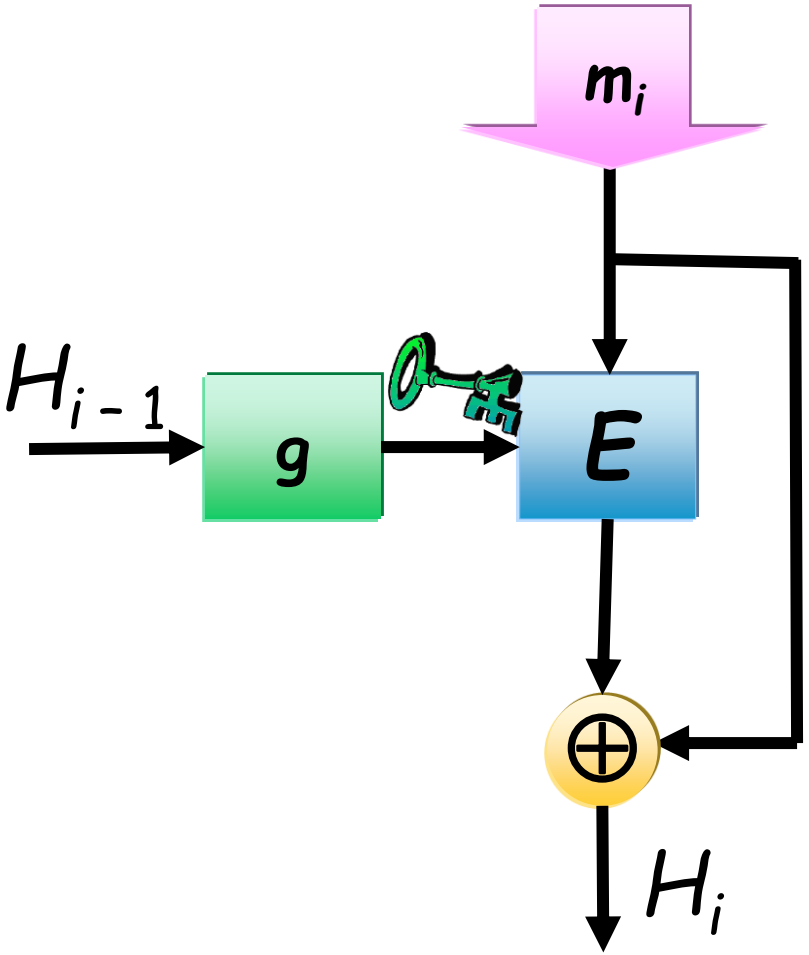


# Kiến trúc Merkle-Damgård




- Tác giả: Ralph Merkle, Ivan Damgård
- Hầu hết các hàm băm đều sử dụng cấu trúc này
- Ví dụ: SHA-1, MD5

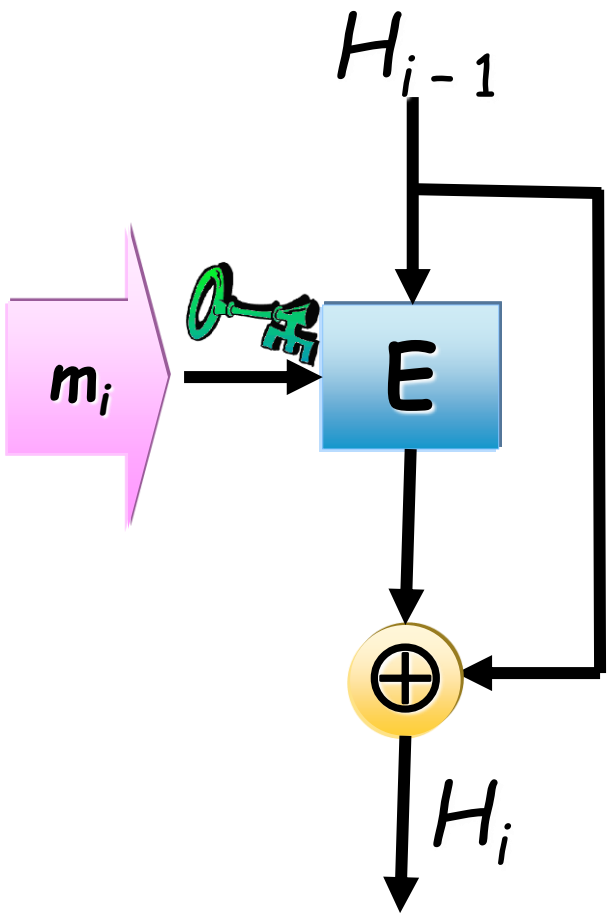
# Kiến trúc Matyas-Meyer-Oseas




- Kiến trúc "đối ngẫu" với kiến trúc **Davies-Mayer**
- Ở khối đầu tiên, cần sử dụng giá trị khởi đầu $H_0$
- Nếu hàm $E$ sử dụng khóa và khối kích thước khác nhau, hàm $g$ cần biến đổi $H_{i-1}$ thành khóa phù hợp cho hàm $E$

$$H_i = E_{g(H_{i-1})}(m_i) \oplus m_i$$

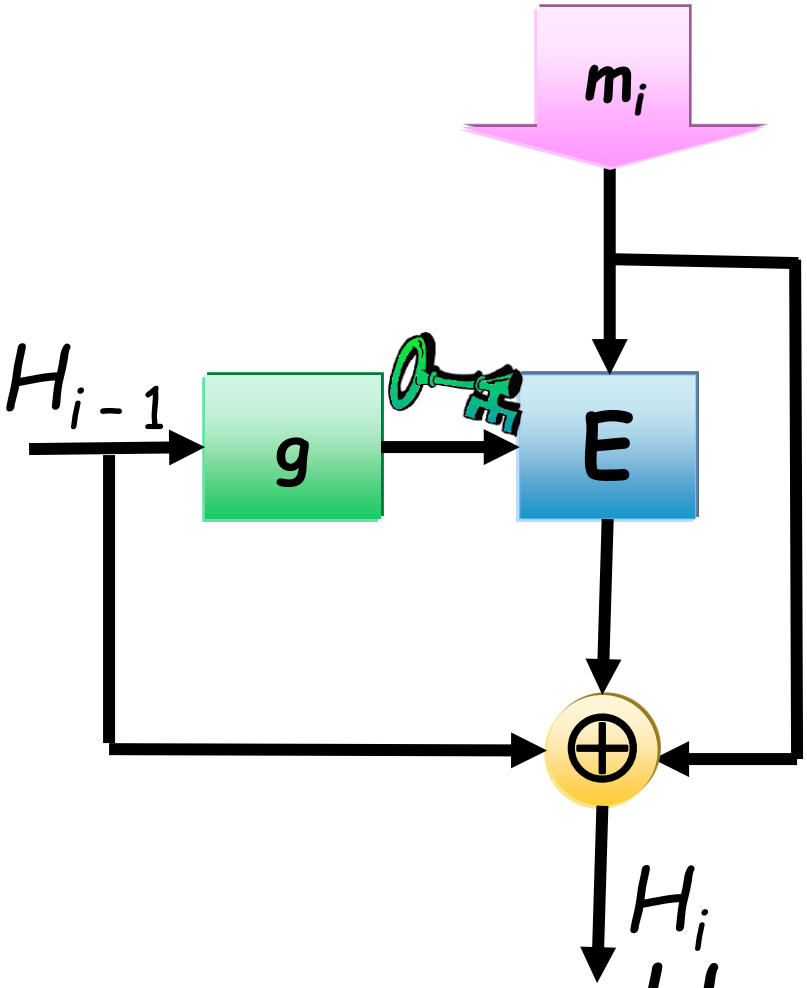
# Kiến trúc Davies-Meyer




- Kiến trúc "đối ngẫu" với kiến trúc **Matyas-Meyer-Oseas**
- Ở khối đầu tiên, cần sử dụng giá trị khởi đầu $H_0$
- Nếu hàm $E$ không an toàn thì có thể áp dụng phương pháp fixed point attack để tấn công hàm băm tương ứng

$$H_i = E_{m_i}(H_{i-1}) \oplus H_{i-1}$$

# Kiến trúc Miyaguchi-Preneel




- Mở rộng của kiến trúc **Matyas-Meyer-Oseas**
- Ở khối đầu tiên, cần sử dụng giá trị khởi đầu *H*0
- Nếu hàm *E* sử dụng khóa và khối kích thước khác nhau, hàm *g* cần biến đổi $H_{i-1}$ thành khóa phù hợp cho hàm *E*

$$H_i = E_{g(H_{i-1})}(m_i) \oplus H_{i-1} \oplus m_i$$

# MD5

- Hàm băm MD4 (Message Digest 4) được Giáo sư Rivest đề nghị vào năm 1990.
- Vào năm sau, phiên bản cải tiến MD5 của thuật toán này ra đời.

# MD5

- *Khởi gán các biến:*
  - h0 := 0x67452301
  - h1 := 0xEFCDAB89
  - h2 := 0x98BADCFE
  - h3 := 0x10325476

# MD5

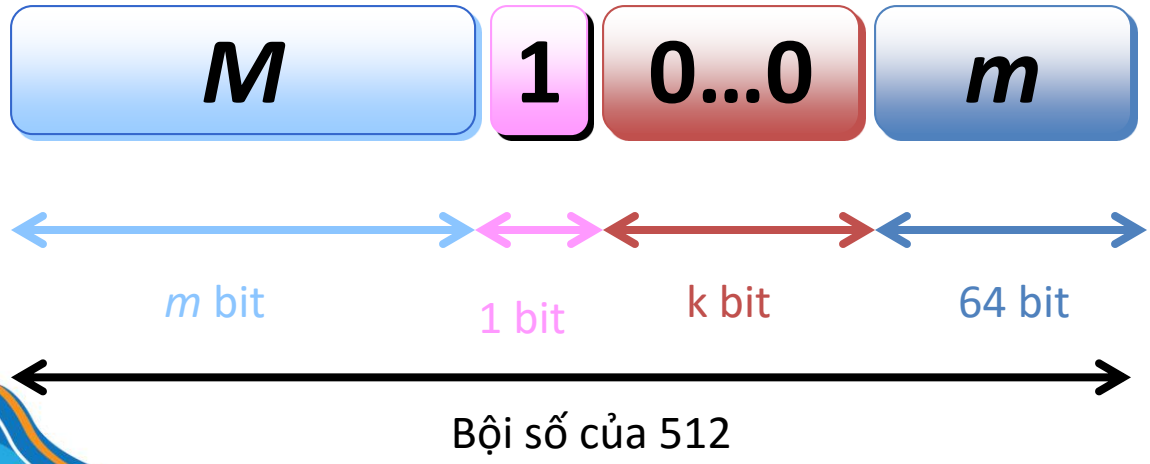
- Hệ số quay trái *R*[*i*]của mỗi chu kỳ:
  - *R*[ 0..15] := {7, 12, 17, 22, 7, 12, 17, 22,
    7, 12, 17, 22, 7, 12, 17, 22}
  - *R*[16..31] := {5, 9, 14, 20, 5, 9, 14, 20,
    5, 9, 14, 20, 5, 9, 14, 20}
  - *R*[32..47] := {4, 11, 16, 23, 4, 11, 16, 23,
    4, 11, 16, 23, 4, 11, 16, 23}
  - *R*[48..63] := {6, 10, 15, 21, 6, 10, 15, 21,
    6, 10, 15, 21, 6, 10, 15, 21}
- Hằng số K[*i*]

**for** *i* **from** 0 **to** 63

*K*[*i*] := floor(abs(sin(*i* + 1)) × (2 **pow** 32))

# MD5

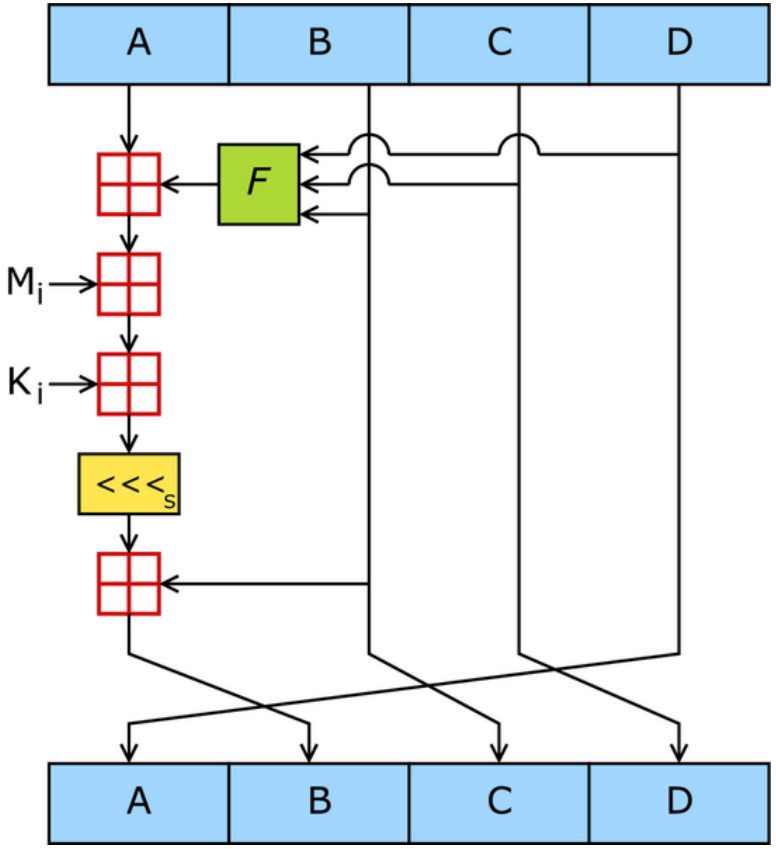
- *Tiền xử lý:*
  - Thêm bit 1 vào cuối thông điệp
  - Thêm vào k bit 0 sao cho độ dài thông điệp nhận được đồng du 448 (mod 512)
  - Thêm 64 bit biểu diễn độ dài dài của thông điệp gốc (giá trị lưu dạng little-endian)



# MD5

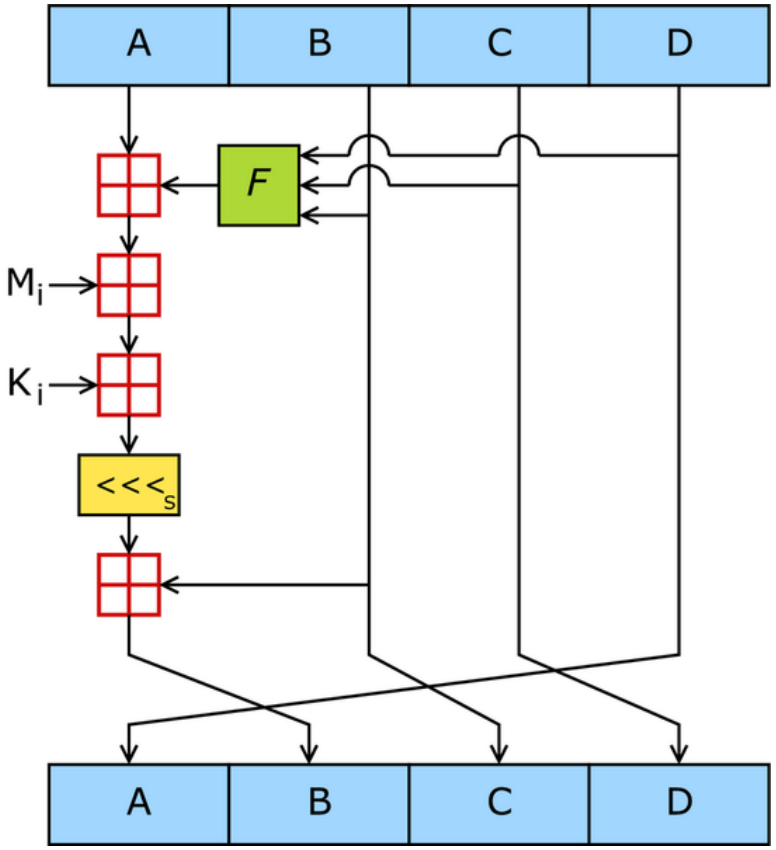
- Chia thông điệp (đã padding) thành các khối 512 bit
- Với mỗi khối 512-bit:
  - Chia thành 16 word (32 bit, little-endian) w[0..15]
  - A= h0, B= h1, C= h2, D= h3
  - 64 chu kỳ xử lý
  - h0+=A, h1+=B, h2+=C, h3+=D, h4+=E
- Kết quả:= h0 | h1 | h2 | h3

# Chu kỳ xử lý trong MD5




- A, B, C, D là 4 word (32 bit) của trạng thái
- *F* là hàm phi tuyến (thay đổi tùy theo chu kỳ)
- <<< *n* là phép quay trái *n* vị trí
- $\boxplus$ phép cộng modulo $2^{32}$.
- $K_t$ là hằng số

# Chu kỳ xử lý trong MD5




**for** *i* **from** 0 to 63

*f* = *F*[*i*] (B, C, D)

*g* = G[*i*] (*i*)

temp = D

D = C

C = B

B = ((A + *f* + *K*[*i*] + w[*g*])

**<<<** R[*i*]) + B

A = temp

# Chu kỳ xử lý trong MD5

- $0 \le i \le 15$
  - $f := (B \wedge C) \vee ((\neg B) \wedge D)$
  - $g := i$
- $16 \le i \le 31$
  - $f := (D \wedge B) \vee ((\neg D) \wedge C)$
  - $g := (5 \times i + 1)$ **mod** 16
- $32 \le i \le 47$
  - $f := B \oplus C \oplus D$
  - $g := (3 \times i + 5)$ **mod** 16
- $48 \le i \le 63$
  - $f := C \oplus (B \vee (\neg D))$
  - $g := (7 \times i)$ **mod** 16

# SHA1

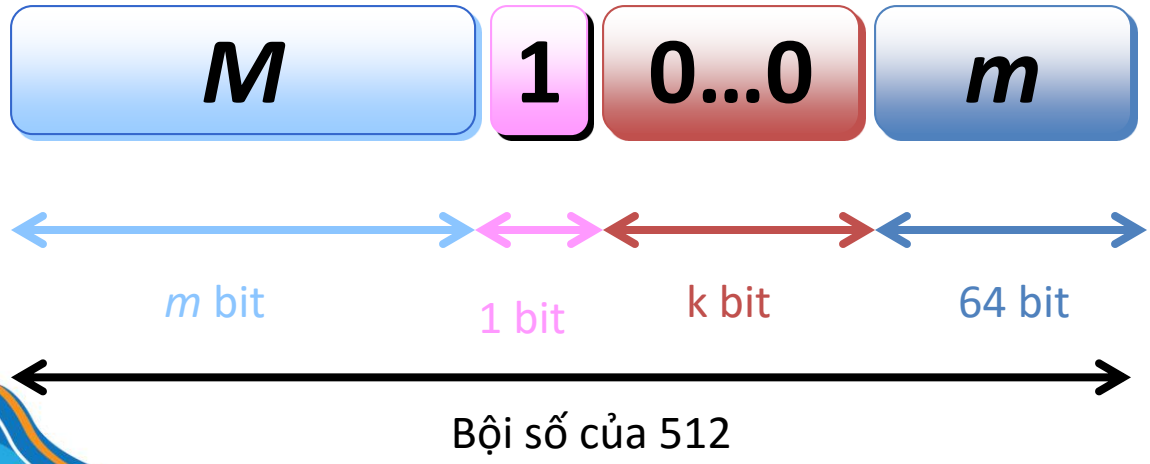
- Phương pháp Secure Hash Standard (SHS hay SHA1) do NIST và NSA xây dựng được công bố trên Federal Register vào ngày 31 tháng 1 năm 1992 và sau đó chính thức trở thành phương pháp chuẩn từ ngày 13 tháng 5 năm 1993.
- Thông điệp được xử lý theo từng khối 512-bit
- Thông điệp rút gọn độ dài 160-bit

# SHA1

- *Khởi gán các biến:*
  - h0 := 0x67452301
  - h1 := 0xEFCDAB89
  - h2 := 0x98BADCFE
  - h3 := 0x10325476
  - h4 := 0xC3D2E1F0

# SHA1

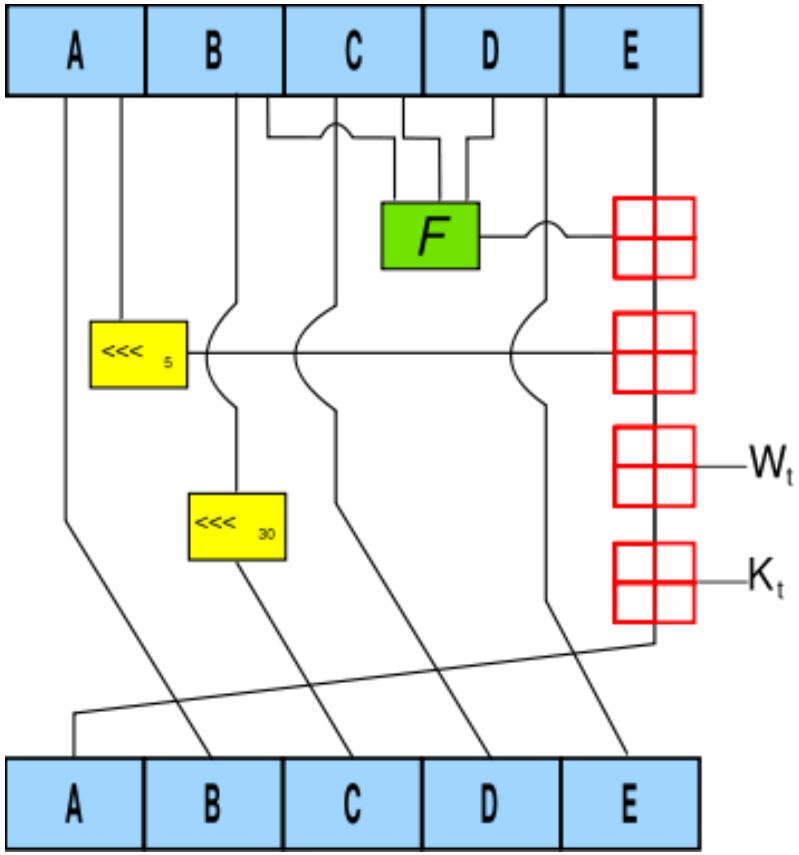
- *Tiền xử lý:*
  - Thêm bit 1 vào cuối thông điệp
  - Thêm vào k bit 0 sao cho độ dài thông điệp nhận được đồng du 448 (mod 512)
  - Thêm 64 bit biểu diễn độ dài dài của thông điệp gốc (giá trị lưu dạng big-endian)



# SHA1

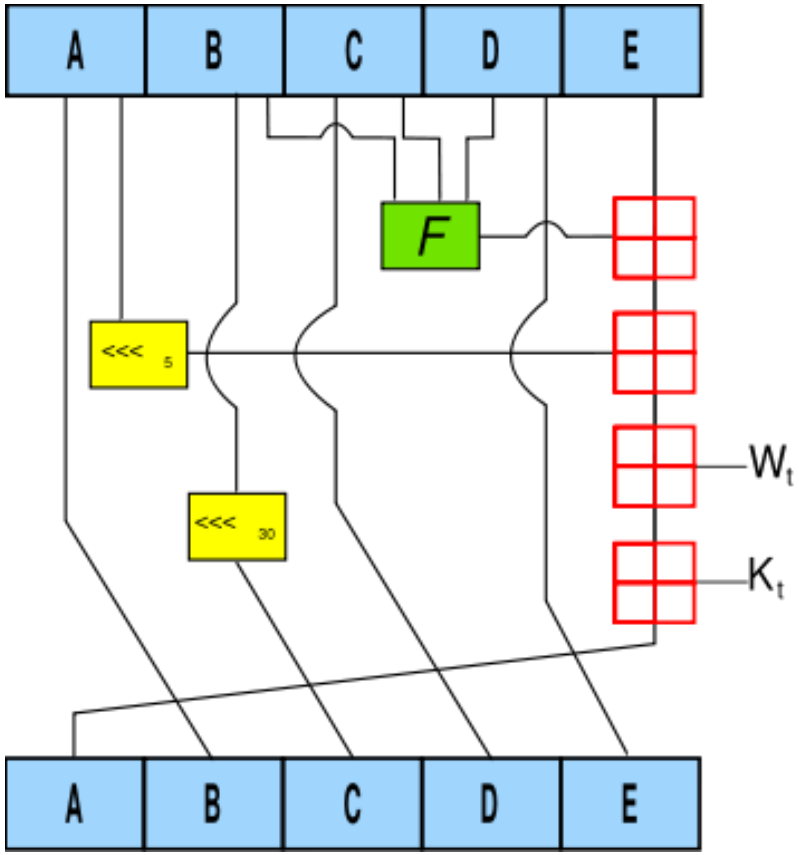
- Chia thông điệp (đã padding) thành các khối 512 bit
- Với mỗi khối 512-bit:
  - Chia thành 16 word (32 bit, big-endian) w[0..15]
  - Mở rộng 16 word (32 bit) thành 80 word (32 bit)
  - w[i]=(w[i-3]⊕ w[i-8] ⊕ w[i-14] ⊕ w[i-16]) **<<<** 1 với $16 \leq i < 80$
  - A= h0, B= h1, C= h2, D= h3, E= h4
  - 80 chu kỳ xử lý
  - h0+=A, h1+=B, h2+=C, h3+=D, h4+=E
- Kết quả:= h0 | h1 | h2 | h3 | h4

# Chu kỳ xử lý trong SHA1




- *t* là số thứ tự của chu kỳ
- A, B, C, D, E là 5 word (32 bit) của trạng thái
- *F* là hàm phi tuyến (thay đổi tùy theo chu kỳ)
- <<< *n* là phép quay trái *n* vị trí
- $\boxplus$ phép cộng modulo $2^{32}$.
- $K_t$ là hằng số

# Chu kỳ xử lý trong SHA1




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

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

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

E = D

D = C

C = B **<<<** 30

B = A

A = temp

# Chu kỳ xử lý trong SHA1

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

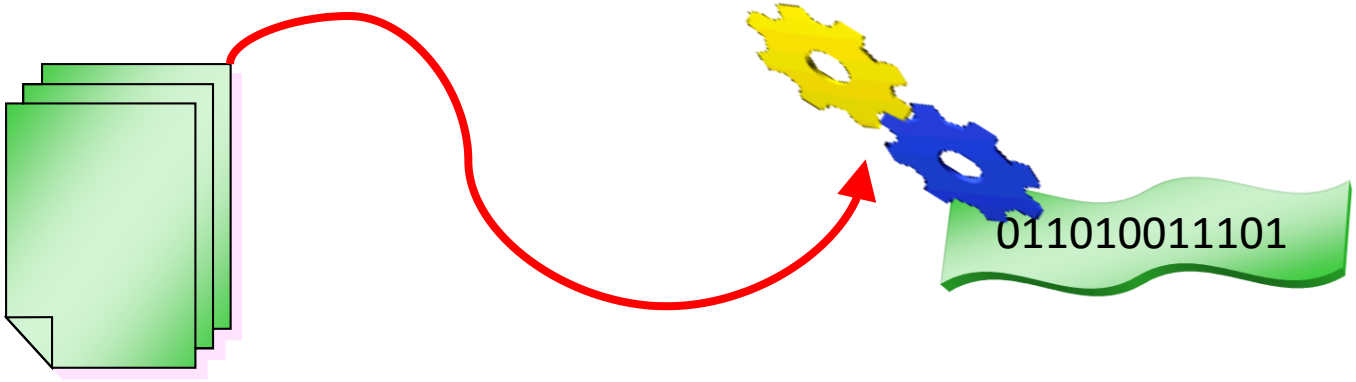
$$K_t = \begin{cases} \texttt{0x5a827999}, & 0 \le t \le 19 \\ \texttt{0x6ed9eba1}, & 20 \le t \le 39 \\ \texttt{0x8f1bbcdc}, & 40 \le t \le 59 \\ \texttt{0xca62c1d6}, & 60 \le t \le 79 \end{cases}$$

# Chu kỳ xử lý trong SHA1

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

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

# Nhóm hàm băm SHA




SHA-256 2002

SHA-384 2002

SHA-1 1994

SHA-224 2004

SHA-512 2002

Secure Hash Standard

# Các thuật toán SHA

| Thuật toán | Kết quả (bit) | Trạng thái (bit) | Khối (bit) | Thông điệp tối đa (bit) | Word (bit) | # chu kỳ | Thao tác | Đụng độ |
|---|---|---|---|---|---|---|---|---|
| SHA-0 | 160 | 160 | 512 | $2^{64} - 1$ | 32 | 80 | +,and,or, xor,rotl | Có |
| SHA-1 | 160 | 160 | 512 | $2^{64} - 1$ | 32 | 80 | +,and,or, xor,rotl | $2^{63}$ thao tác |
| SHA-256/224 | 256/ 224 | 256 | 512 | $2^{64} - 1$ | 32 | 64 | +,and, or,xor, shr,rotr | Chưa |
| SHA-512/384 | 512/ 384 | 512 | 1024 | $2^{128} - 1$ | 64 | 80 | +,and, or,xor, shr,rotr | Chưa |

# Sử dụng SHA

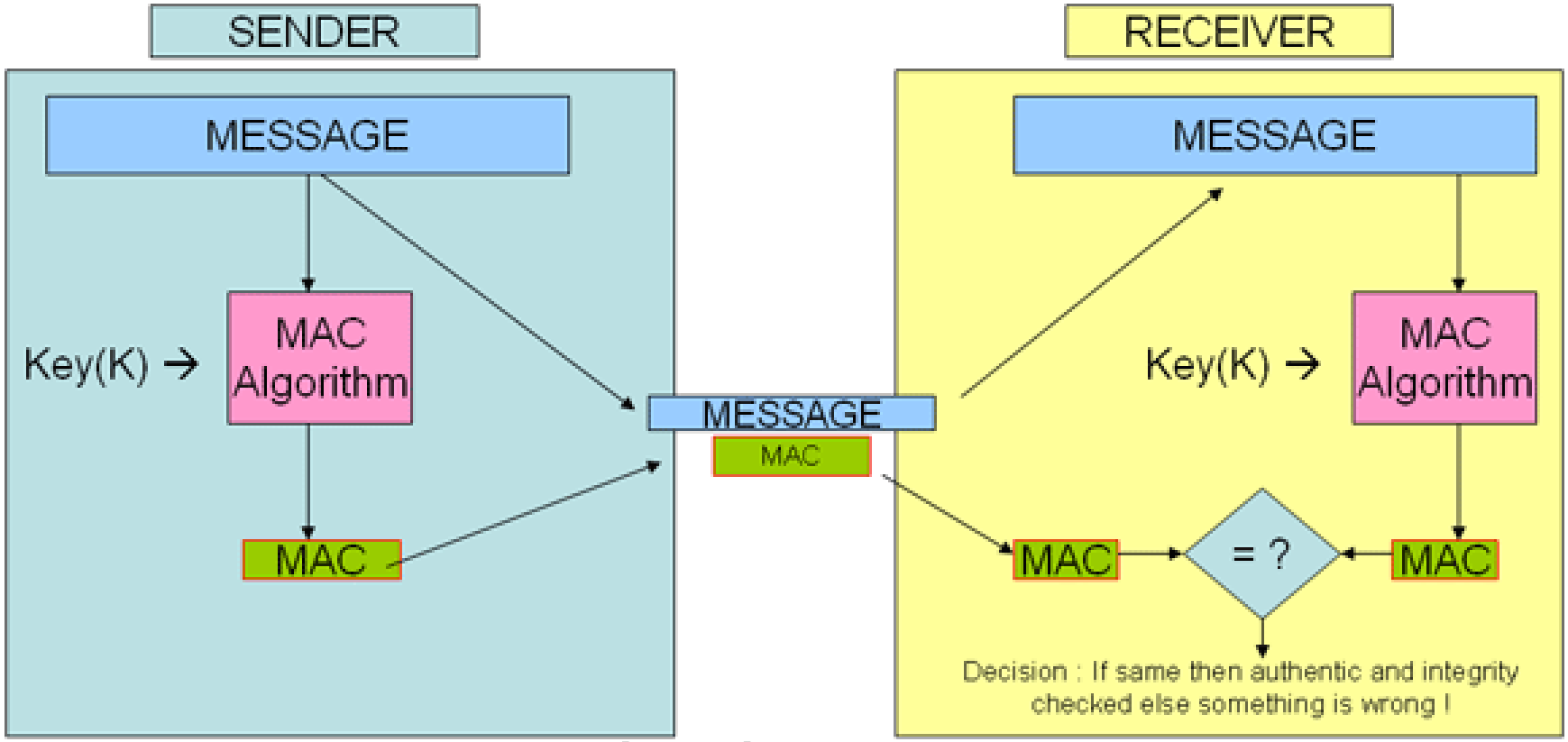
| Loại ƯD | Sử dụng thông thường | | Suite B | |
|---|---|---|---|---|
| Thuật toán | Đến 2010 | Sau 2010 | Secret | Top Secret |
| SHA-1 | √ | | | |
| SHA-224 | √ | √ | | |
| SHA-256 | √ | √ | √ | |
| SHA-384 | √ | √ | √ | √ |
| SHA-512 | √ | √ | | |

Nguồn: NIST Cryptographic Standards Status Report

April 4, 2006

Bill Burr
Manager, Security Technology Group
NIST
william.burr@nist.gov

# Message authentication code (MAC)




Mục đích: xác định nguồn gốc của thông tin

MAC – Message Authentication Code
(rsh) 2007

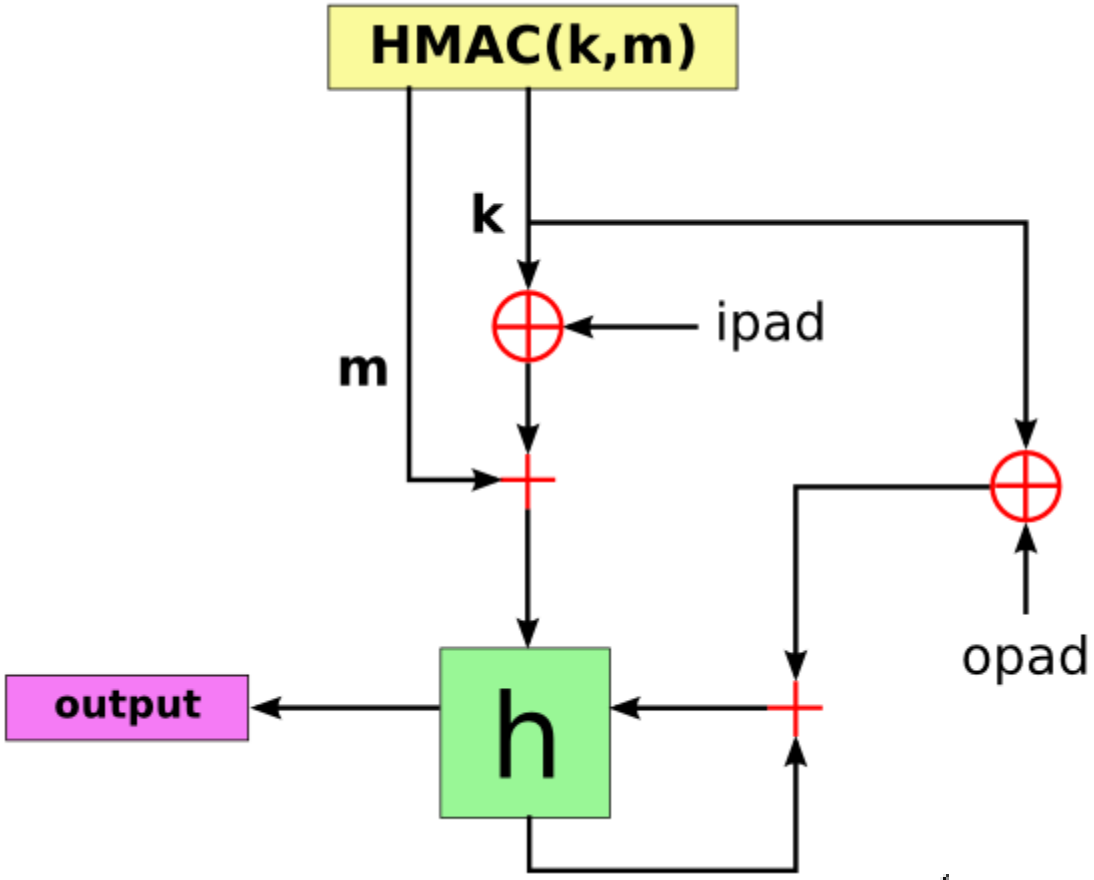
# MAC và chữ ký điện tử

- Phát sinh MAC và kiểm tra MAC sử dụng chung khóa bí mật (secret key)
- Người gửi và người nhận phải thỏa thuận trước khóa bí mật (giống mã hóa đối xứng)
- Không hỗ trợ việc chống từ chối trách nhiệm (non-repudiation)

# Message authentication code (MAC)

- MAC có thể được tạo ra từ hàm băm mật mã (HMAC) hay từ giải thuật mã hóa theo khối (OMAC, CBC-MAC, PMAC)

# Keyed-hash message authentication code




$$\text{opad} = \texttt{0x5c5c5c...5c5c}$$

$$\text{ipad} = \texttt{0x363636...3636}$$

$$\mathrm{HMAC}_K(m) = h\Big( (K \oplus \text{opad}) \| h\big( (K \oplus \text{ipad}) \| m \big) \Big)$$

Mihir Bellare, Ran Canetti, Hugo Krawczyk (1996 )

# Sử dụng hàm băm mật mã trong chữ ký điện tử


KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

# Sử dụng tạo chữ ký điện tử (đính kèm)

- Giải pháp cơ bản
- ANSI X9.31
- PKCS #1 v1.5
- Bellare-Rogaway FDH
- Bellare-Rogaway PSS

# Giải pháp cơ bản

- *M*: thông điệp cần ký
- $\mu(M)$ = Hash(*M*)
- Ký trực tiếp trên $\mu(M)$
- Với cùng 1 thông điệp *M*, chữ ký (của cùng 1 người) luôn luôn giống nhau ➔ An toàn?
- Mang tính minh họa trong giảng dạy
- Không nên dùng trong thực tế

# ANSI X9.31

(Digital Signatures Using Reversible Public-Key Cryptography for the Financial Services Industry, 1998)

- $\mu(M)$ = 6b bb … bb ba || Hash($M$) || 3$x$ cc

  với $x$ = 3 nếu dùng for SHA-1,

  $x$ = 1 nếu dùng RIPEMD-160

- Được hỗ trợ trong nhiều chuẩn
  - IEEE P1363, ISO/IEC 14888-3
  - US NIST FIPS 186-1

# PKCS #1 v1.5

(RSA Encryption Standard, 1991)

- $\mu(M)$ = 00 01 ff … ff 00 || HashAlgID || Hash($M$)
- Được sử dụng rộng rãi
  - SSL certificate
  - S/MIME
- Được đưa vào chuẩn IEEE P1363a; còn tiếp tục dùng trong PKCS #1 v2.0

# Bellare-Rogaway FDH

(Full Domain Hashing, ACM CCCS ’93)

- $\mu(M) = 00 \,||\, \text{Full-Length-Hash}(m)$
- Được đưa vào chuẩn IEEE P1363a

# Bellare-Rogaway PSS

(Probabilistic Signature Scheme, Eurocrypt '96)

- $\mu(M) = 00 \,||\, H \,||\, G(H) \oplus [salt \,||\, 00 \ldots 00]$

  với

  $H = \text{Hash}(salt, M)$,

  *salt* là chuỗi giá trị ngẫu nhiên,

  *G* là hàm biến đổi H thành chuỗi bit có độ dài phù hợp để XOR với i [*salt* || 00 … 00]
- Được đưa vào chuẩn IEEE P1363a; ANSI X9.31

# Sử dụng tạo chữ ký điện tử (khôi phục được nội dung)

- Giải pháp cơ bản
- ISO/IEC 9796-1
- ISO/IEC 9796-2
- Bellare-Rogaway PSS-R

# Giải pháp cơ bản

- $\mu(M) = M$
- Minh họa trong giảng dạy
- Không an toàn trong thực tế

# ISO/IEC 9796-1

(Digital Signature Scheme Giving Message Recovery, 1991)

- $\mu(M) = s^*(m_{l-1})\ s'(m_{l-2})\ m_{l-1}\ m_{l-2}$
  $s(m_{l-3})\ s(m_{l-4})\ m_{l-3}\ m_{l-4}$ ...
  $s(m_3)\ s(m_2)\ m_3\ m_2$
  $s(m_1)\ s(m_0)\ m_0$

  với $m_i$ là 4 bit thứ $i$ của $M$

  $s^*$, $s'$ và $s$ là các hoán vị (cố định)
- *Không an toàn đối với tấn công bằng phép nhân* (multiplicative forgery [CHJ99], [Grieu 1999])
  - Có thể dùng nếu $M$ là giá trị hash

# ISO/IEC 9796-2

(Digital Signature Scheme Giving Message Recovery — Mechanisms Using a Hash Function, 1997)

- $\mu(M)$ = 4b bb bb … bb ba || $M$ || Hash($M$) || bc

- Không an toàn với phương pháp tấn công bằng phép nhân (multiplicative forgery) nếu giá trị hash gồm từ 64 bit trở xuống [CNS99]
  - Vẫn có thể sử dụng an toàn với các giá trị hash có nhiều hơn 64 bit

# Bellare-Rogaway PSS-R

(Probabilistic Signature Scheme with Recovery, 1996)

- $\mu(M) = 00 \,||\, H \,||\, G(H) \oplus [salt \,||\, 00 \ldots 01 \,||\, M]$

  $H = \text{Hash}(salt, M)$,

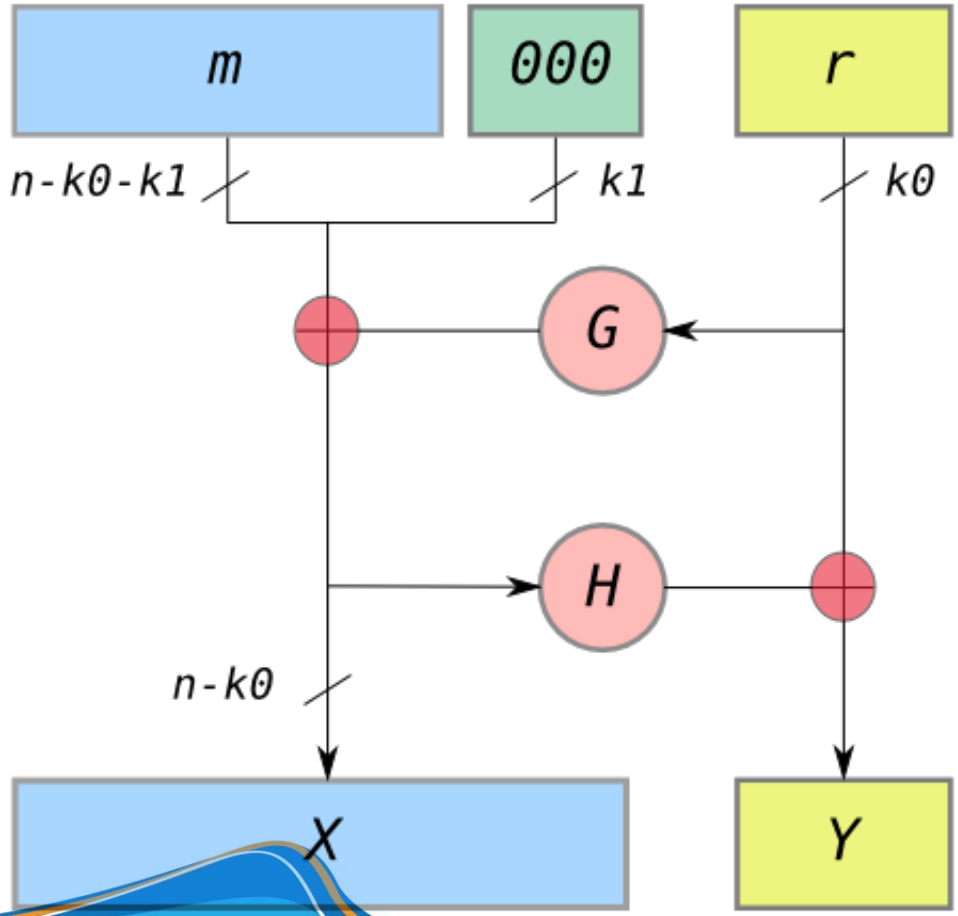
  *salt* là chuỗi giá trị ngẫu nhiên,

  *G* là hàm biến đổi H thành chuỗi bit có độ dài phù hợp để XOR với $[salt \,||\, 00 \ldots 01 \,||\, M]$
- Được đưa vào chuẩn IEEE P1363a; ISO/IEC 9796-2

# Sử dụng hàm băm mật mã trong mã hóa bất đối xứng


KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

# Optimal Asymmetric Encryption Padding




Tác giả:

- Bellare và Rogaway (1994)

Khi mã hóa:

- $X = m00..0 \oplus G(r)$
- $Y = r \oplus H(X)$

Khi giải mã:

- $r = Y \oplus H(X)$
- $m00..0 = X \oplus G(r)$

# Optimal Asymmetric Encryption Padding

- Trong .Net, OAEP sử dụng SHA1 với độ dài của giá trị băm là 160 bit = 20 byte
- Do đó, khi mã hóa bất đối xứng dùng RSA với OAEP, độ dài của chuỗi dữ liệu cần mã hóa (tính bắng byte) tối đa là:

  $n$ – 2 * 20 byte – 2= $n$ – 42 byte

  với n là độ dài (tính bằng byte) của modulus (ví dụ 512 bit = 32 byte)

# Optimal Asymmetric Encryption Padding

- Trong .Net, OAEP sử dụng SHA1 với độ dài của giá trị băm là 160 bit = 20 byte
- Do đó, khi mã hóa bất đối xứng dùng RSA với OAEP, độ dài của chuỗi dữ liệu cần mã hóa (tính bắng byte) tối đa là:

  $n$ – 2 * 20 byte – 2= $n$ – 42 byte

  với n là độ dài (tính bằng byte) của modulus (ví dụ 512 bit = 32 byte)

# Một số ứng dụng thực tế của hàm băm


KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

# Sử dụng trong chứng nhận (certification)

- Chữ ký điện tử
  - Sử dụng hàm băm:
    - Nén thông tin
    - Kết hợp thông tin ngẫu nhiên
  - Yêu cầu về bảo mật:
    - An toàn đối với tấn công second pre-image
    - Phải hợp lệ trong ít nhất 5 năm (theo SOX), hoặc 7 năm (theo HIPPA)
- Một số ví dụ khác:
  - PKI
  - Time-stamping (nhãn thời gian)

# Sử dụng trong định danh chứng thực người dùng (authentication)

- Kerberos
  - Sử dụng hàm băm:
    - Tính khóa bí mật của người dùng
    - Kiểm soát tính toàn vẹn của thông điệp trong protocol
  - Yêu cầu về bảo mật:
    - An toàn đối với tấn công second pre-image
    - Phải hợp lệ trong 1 phiên làm việc
- Một số ví dụ khác:
  - IEEE 802.1X-EAP
  - APOP

# Sử dụng trong liên lạc an toàn

- IPSec
  - Sử dụng hàm băm:
    - Chứng thực (authentication) trong quá trình trao đổi khóa
    - Kiểm soát tính toàn vẹn của thông điệp trong protocol
  - Yêu cầu về bảo mật:
    - An toàn đối với tấn công second pre-image
    - Phải hợp lệ trong 1 phiên làm việc
- Một số ví dụ khác:
  - SSL/TLS
  - SSH

# Sử dụng trong email

- S/MIME
  - Sử dụng hàm băm:
    - Dùng trong chữ ký điện tử
  - Yêu cầu về bảo mật:
    - An toàn đối với tấn công second pre-image
    - Phải hợp lệ trong thời gian dài nếu cần dùng làm bằng chứng
- Một số ví dụ khác:
  - PGP (pretty good privacy)

# Một số ứng dụng khác

- Kiểm tra tính toàn vẹn của phần mềm/dữ liệu khi download.
- Đối sánh CSDL (Database matching)
- …

# Mật khẩu người dùng

- Lưu trong CSDL: username + password
  - Kiểm tra: so sánh password của người dùng nhập vào và password đã lưu trong CSDL
  - ➔ An toàn? Admin biết password của người dùng!

- Lưu trong CSDL: username + hash (password)
  - Kiểm tra: so sánh

    hash (password người dùng nhập)

    = hash (password đã lưu)?
  - ➔ An toàn hơn
  - Còn vấn đề gì đáng lo ngại hay không?

# Mật khẩu người dùng

- Lưu trong CSDL:

  username + salt + H với H = hash (password, salt)

  - Kiểm tra: so sánh

    hash (password người dùng nhập, salt) = H ?