

# BÀI 3.

# XÁC THỰC THÔNG ĐIỆN

---

Bùi Trọng Tùng,  
Viện Công nghệ thông tin và Truyền thông,  
Đại học Bách khoa Hà Nội

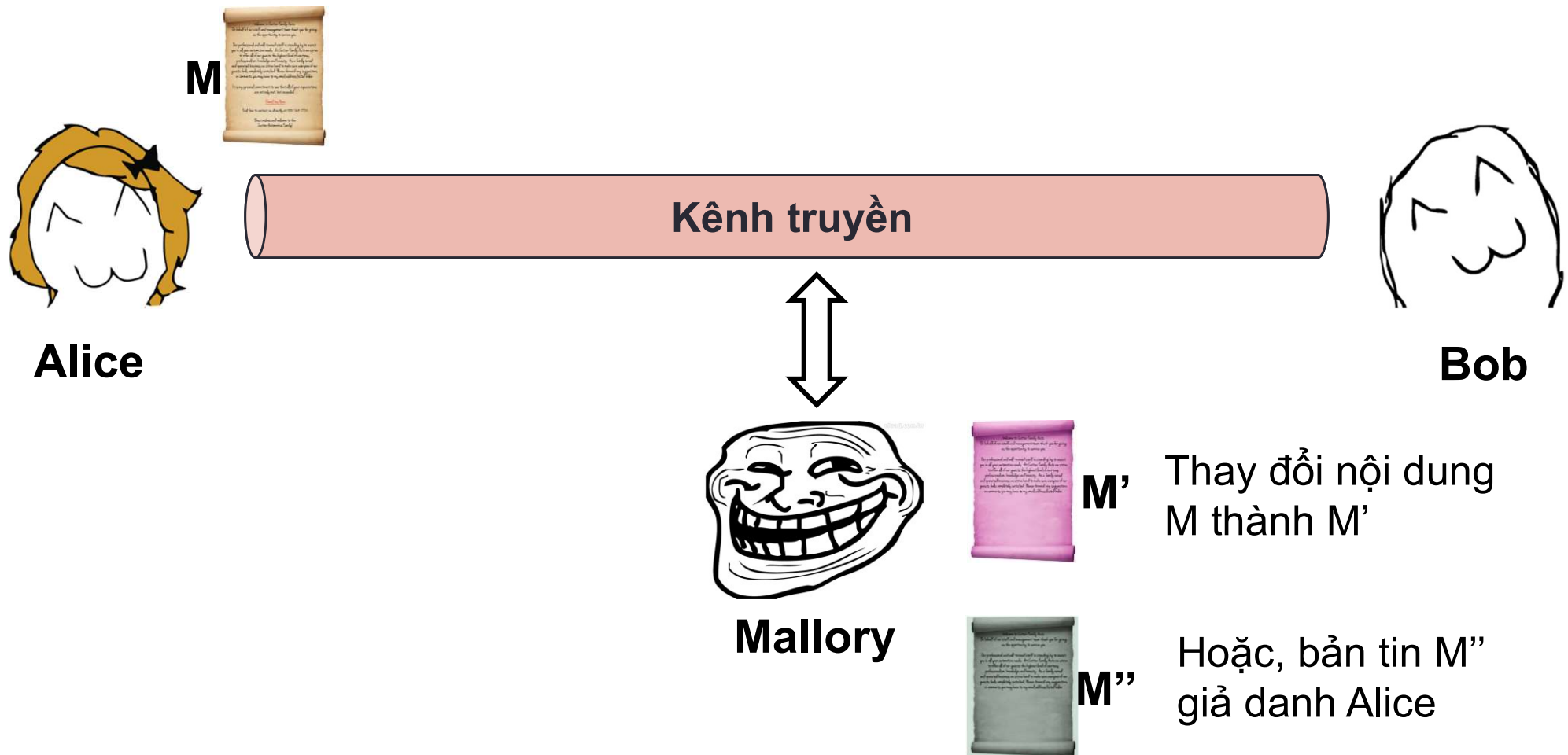
# Nội dung

- Các vấn đề xác thực thông điệp
- Mã xác thực thông điệp (MAC)
- Hàm băm và hàm băm HMAC

# 1. ĐẶT VẤN ĐỀ

---

# 1. Đặt vấn đề



# Xác thực thông điệp

- Bản tin phải được xác minh:
  - Nội dung toàn vẹn: bản tin không bị sửa đổi
    - ✓ Bao hàm cả trường hợp Bob cố tình sửa đổi
  - Nguồn gốc tin cậy:
    - ✓ Bao hàm cả trường hợp Alice phủ nhận bản tin
    - ✓ Bao hàm cả trường hợp Bob tự tạo thông báo và “vu khống” Alice tạo ra thông báo này
  - Đúng thời điểm
- Các dạng tấn công điển hình vào tính xác thực: Thay thế (Substitution), Giả danh (Masquerade), tấn công phát lại (Replay attack), Phủ nhận (Repudiation)

# Tấn công thay thế

- Chặn thông điệp, thay đổi nội dung và chuyển tiếp cho bên kia

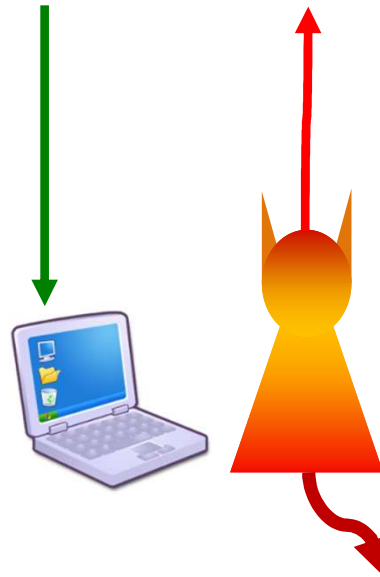
**“Tôi là Alice.  
Số tài khoản của tôi  
là 123. Hãy chuyển  
tiền cho tôi!”**



Alice



Bob

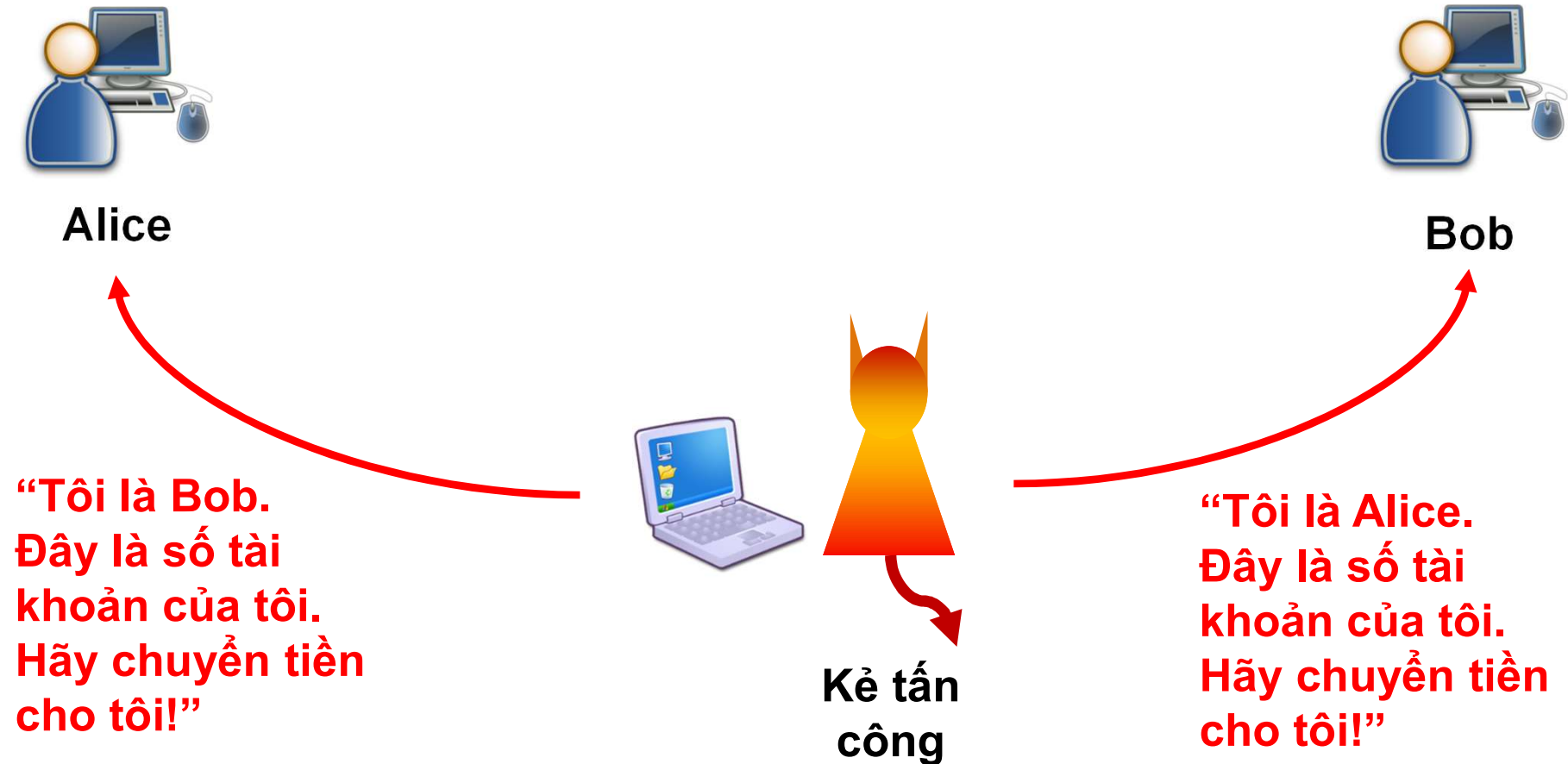


Kẻ tấn công

**“Tôi là Alice.  
Số tài khoản của tôi là 456. Hãy  
chuyển tiền cho  
tôi!”**

# Tấn công giả danh

- Kẻ tấn công mạo danh một bên và chuyển các thông điệp cho bên kia.



# Tấn công phủ nhận gửi

- Bên gửi phủ nhận việc đã gửi đi một thông tin



Alice

**“Tôi là Alice.  
Hãy chuyển tiền của tôi vào tài  
khoản 123!”**



Bob

**“Tôi là Bob.  
Tôi đã chuyển tiền của cô vào tài  
khoản 123.”**



**“Không.  
Tôi chưa bao giờ yêu cầu chuyển  
tiền của tôi vào tài khoản 123!”**





## 2. MÃ XÁC THỰC THÔNG ĐIỆN (MAC)

---

BTT2

## Slide 9

---

**BTT2**

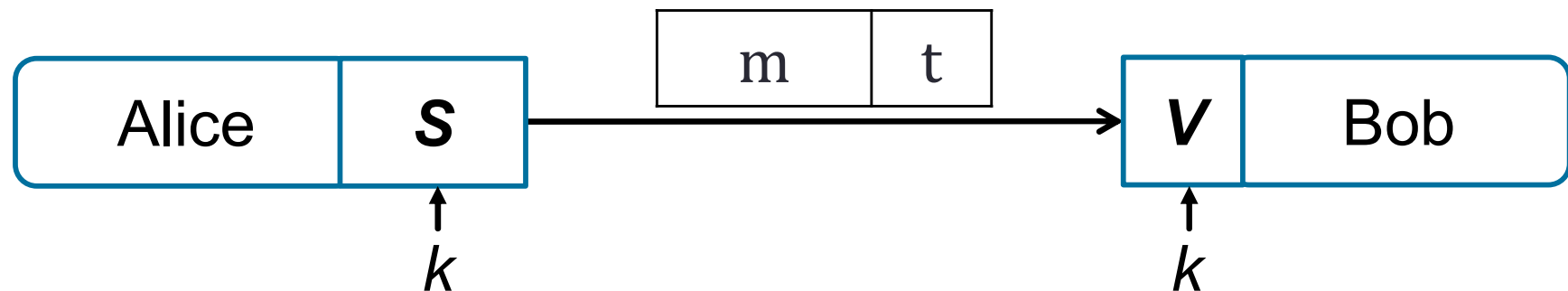
Rollback resistance

Rollback attack

Bui Trong Tung, 21/01/2021

# Message Authentication Code

- Hai bên đã trao đổi một cách an toàn khóa mật  $k$
- Hàm  $MAC = (S, V)$  là một cặp thuật toán
- Sinh mã:  $t = S(k, m)$ 
  - Đầu ra: kích thước cố định, không phụ thuộc kích thước của  $m$
- Xác minh:  $V(k, m, t)$ 
  - Nếu  $t = S(k, m)$  thì  $V = \text{true}$
  - Ngược lại  $V = \text{false}$



# MAC – Ví dụ 1

## Khách hàng chuyển khoản

1. Chia sẻ khóa  $k$
2.  $m = \text{SoTK} \parallel \text{money}$
3.  $t = S(k, m)$

$m$	$t$
-----	-----

$V(k, m, t) = \text{True}$

**V** Ngân hàng

$m'$	$t'$
------	------

$V(k, m', t') = \text{False}$

## Kẻ tấn công

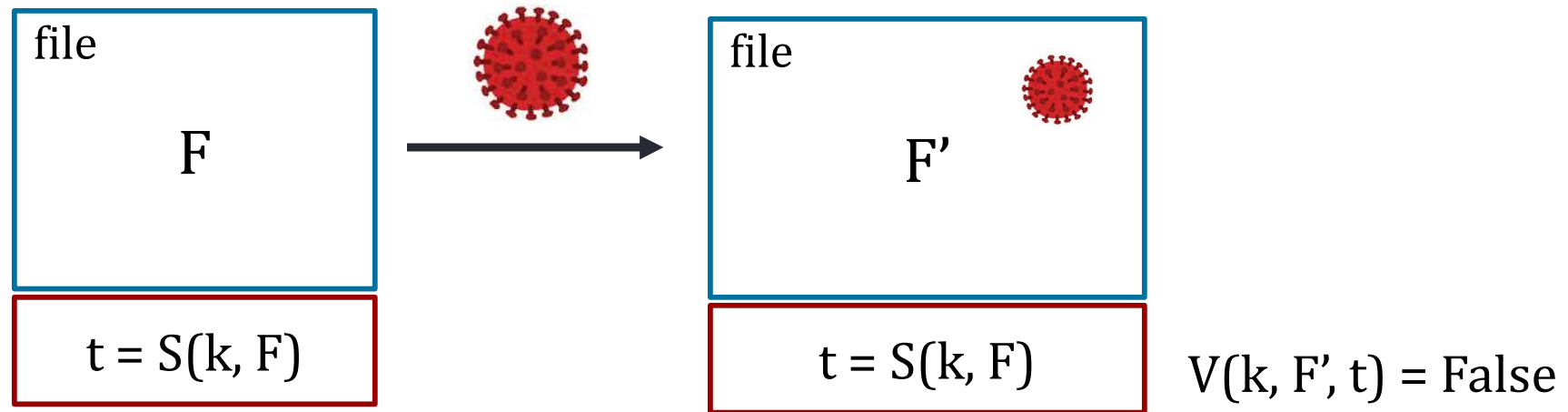
1. Không biết  $k$
2. Tạo  $m' = \text{SoTK}' \parallel \text{money}$
3. Tạo  $t'$

Mã MAC cho phép  
phát hiện thông tin bị  
sửa đổi

Thay đổi số tài  
khoản nhận tiền

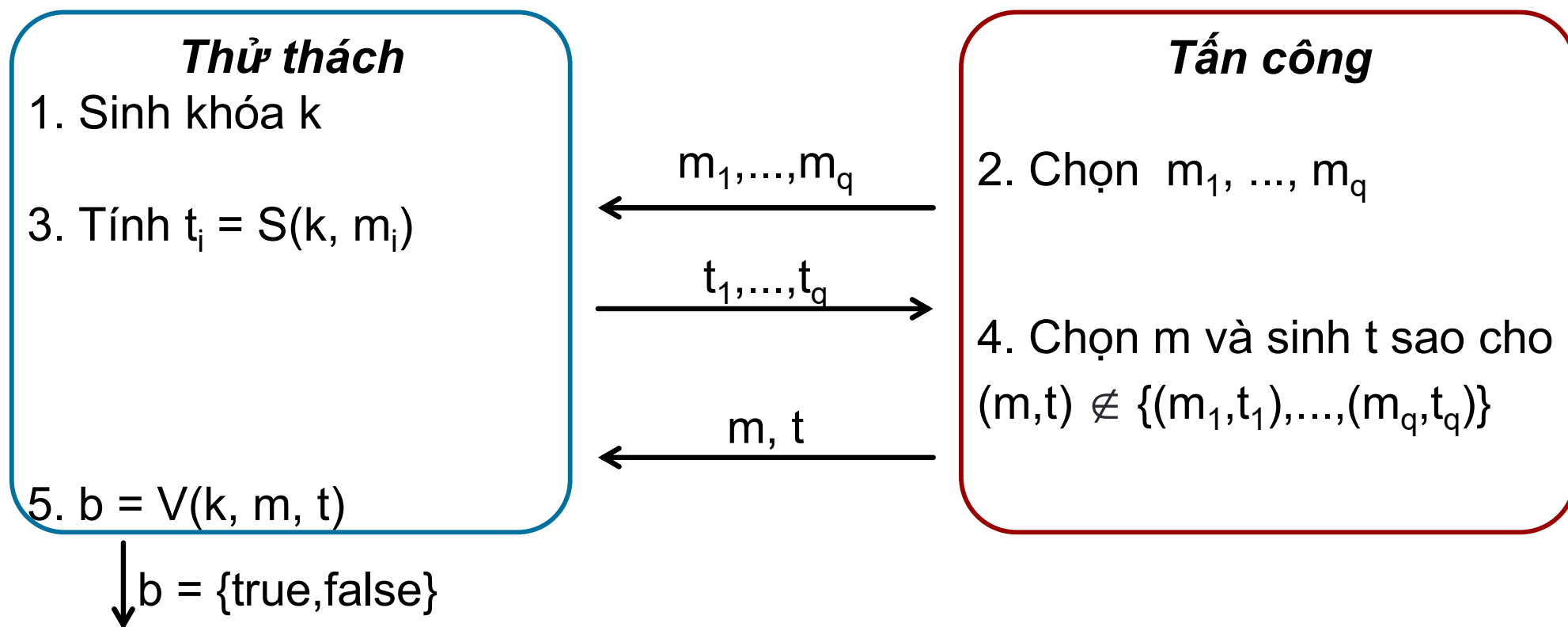
# MAC – Ví dụ 2: Phần mềm Tripwire

- Khi cài đặt, tính giá trị MAC của các file cần bảo vệ



- Khi máy tính khởi động, các file được kiểm tra mã MAC  
→ Cho phép phát hiện các file bị sửa đổi (ví dụ do nhiễm virus)

# An toàn của MAC



- MAC là an toàn nếu với mọi thuật toán tấn công hiệu quả thì xác suất  $P(b = \text{true}) \leq \epsilon$   
→ kẻ tấn công không thể tạo giá trị  $t$  hợp lệ nếu không có khóa  $k$

# An toàn của MAC

- MAC còn an toàn không nếu tồn tại thuật toán hiệu quả cho một trong các tình huống sau:
  - (1) Tìm được  $m^*$  sao cho  $S(?, m^*) = t$  với  $t$  chọn trước
  - (2) Tìm được  $m^*$  sao cho  $S(?, m^*) = S(?, m)$  với  $m$  chọn trước
  - (3) Tìm được  $m$  và  $m^*$  sao cho  $S(?, m^*) = S(?, m)$
- Hoặc giá trị  $t$  có kích thước 10 bit

# Một ví dụ khác

- Một hàm MAC được tạo ra như sau:

$$S'(k, m1 \parallel m2) = S(k, m1) \parallel S(k, m2)$$

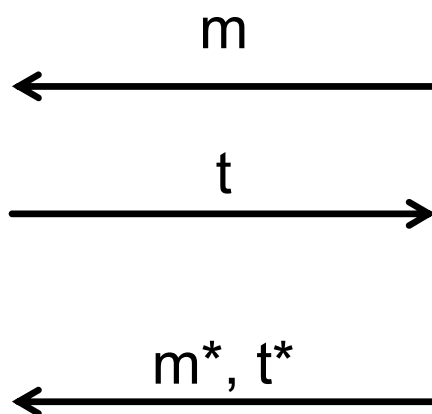
Trong đó  $S$  là hàm tạo mã MAC an toàn.  $S'$  có phải là hàm MAC an toàn hay không?

## ***Thử thách***

1. Sinh khóa  $k$

3. Tính  $t = S'(k, m)$   
 $= S(k, m1) \parallel S(k, m2)$   
 $= t1 \parallel t2$

5.  $b = V(k, m^*, t^*) = \text{true}$



## ***Tấn công***

2. Chọn  $m = m1 \parallel m2$

4. Chọn  $m^* = m2 \parallel m1$   
 $t^* = t2 \parallel t1$



# Độ an toàn của MAC

- Giả sử  $m_1$  và  $m_2$  là hai bản tin có mã MAC giống nhau:

$$S(k, m_1) = S(k, m_2)$$

→  $S(k, m_1 || W) = S(k, m_2 || W)$  với  $W$  bất kỳ

- Kịch bản tấn công:

1. Kẻ tấn công nhận được  $t_x = S(k, m_x)$  với  $x = 1, \dots, N$
2. Tìm cặp bản tin  $(m_i, m_j)$  có  $t_i = t_j$ . Nếu không tìm thấy thực hiện lại bước 1
3. Chọn bản tin  $W$  và tính  $t = S(k, m_i || W)$
4. Thay  $m_i || W$  bằng  $m_j || W$  có lợi cho kẻ tấn công

# Ví dụ tấn công vào tính độ

(1) Kẻ tấn công(Mr. Tung) tìm được 2 bản tin có mã MAC giống nhau:

m1: 'I will pay 1'

m2: 'I will pay 2'

Chọn W = '000\$ to Mr.Tung'

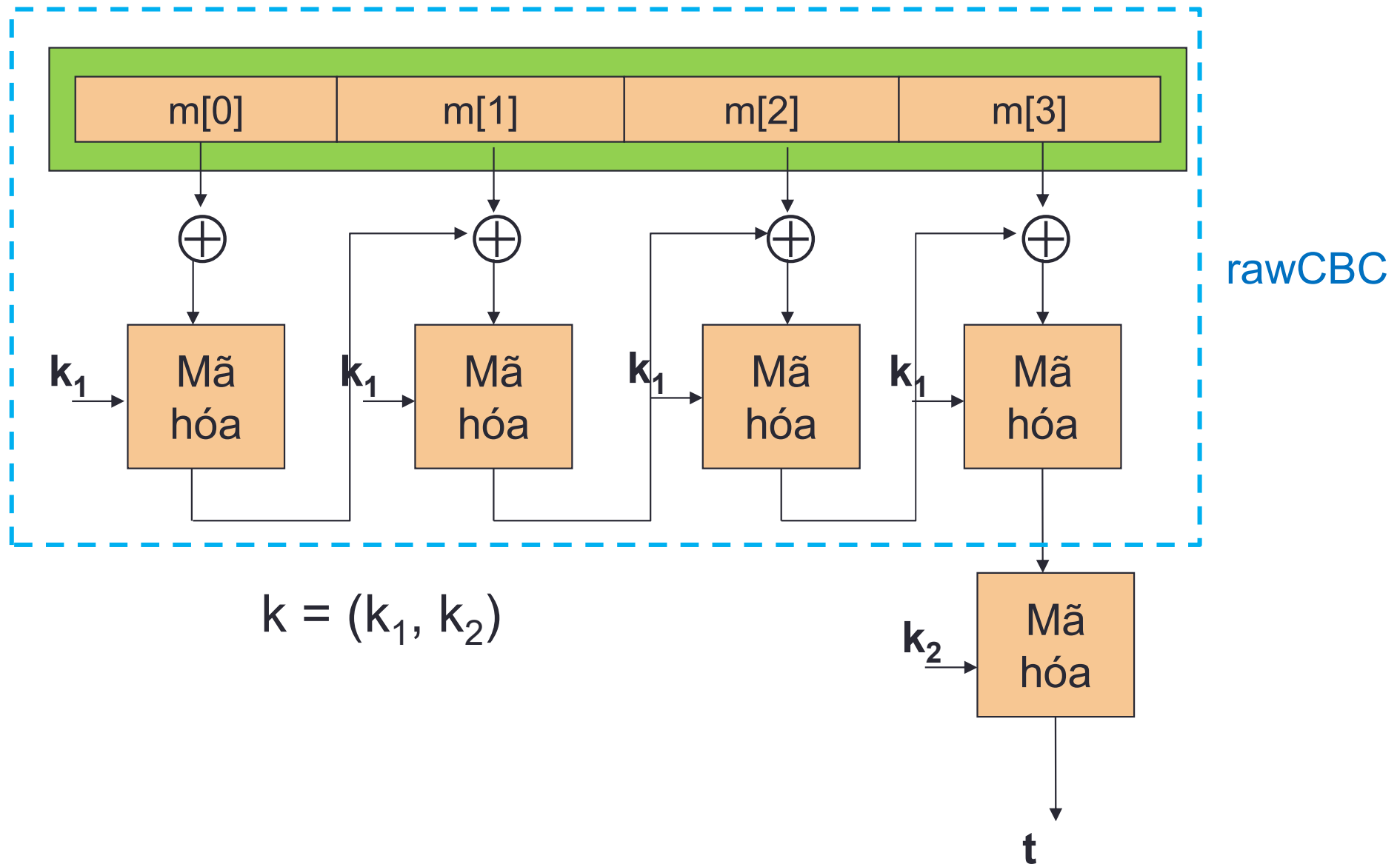
m1 || W = 'I will pay 1000\$ to Mr.Tung'

m2 || W = 'I will pay 2000\$ to Mr.Tung'

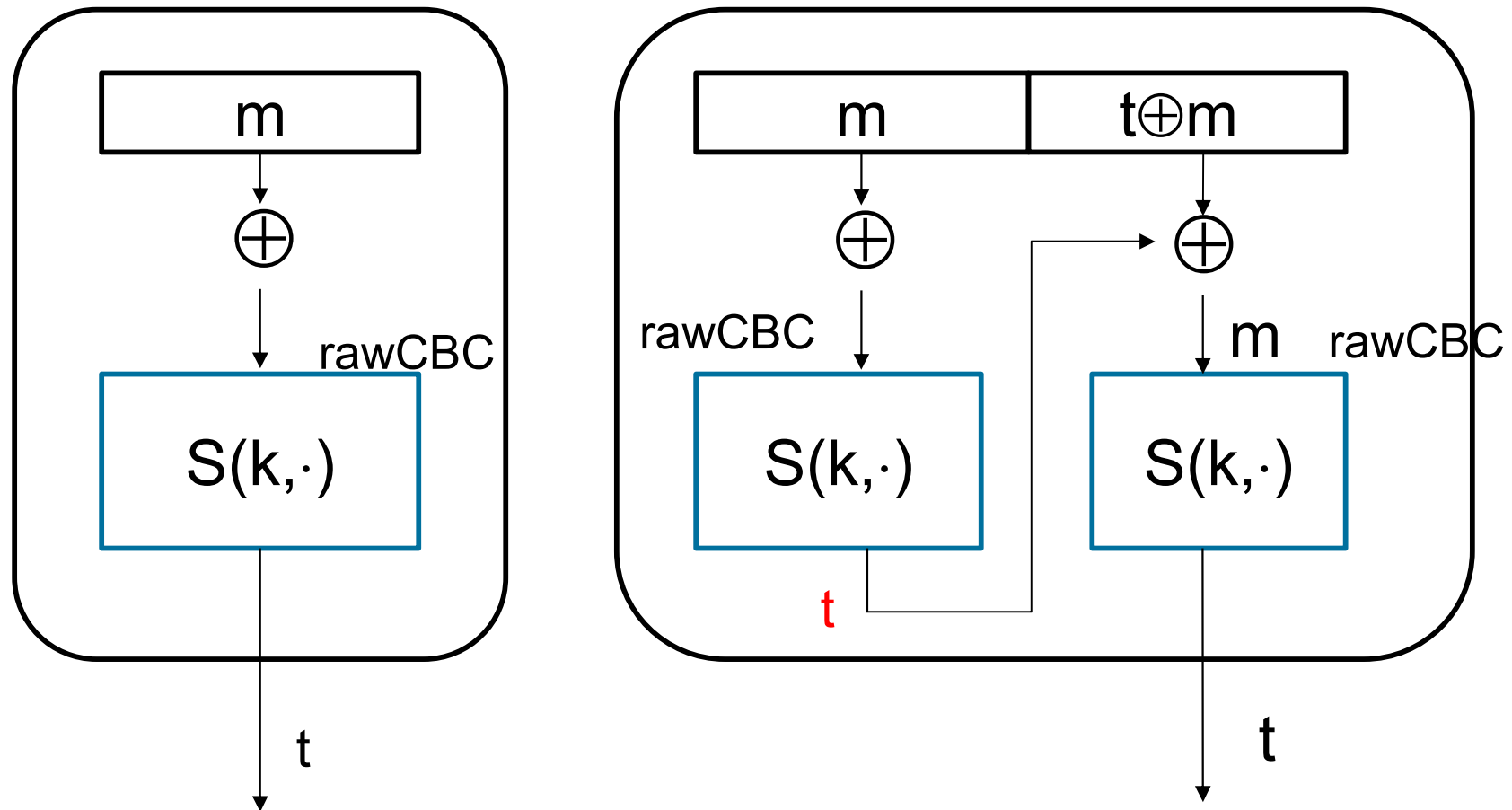
(2) Đánh lừa người dùng gửi bản tin 'I will pay 1000\$ to Mr.Tung' || S(k, 'I will pay 1000\$ to Mr.Tung') cho ngân hàng

(3) Thay thế bằng 'I will pay 2000\$ to Mr.Tung' || S(k, 'I will pay 1000\$ to Mr.Tung') → Ngân hàng chấp nhận

# Xây dựng MAC: CBC-MAC



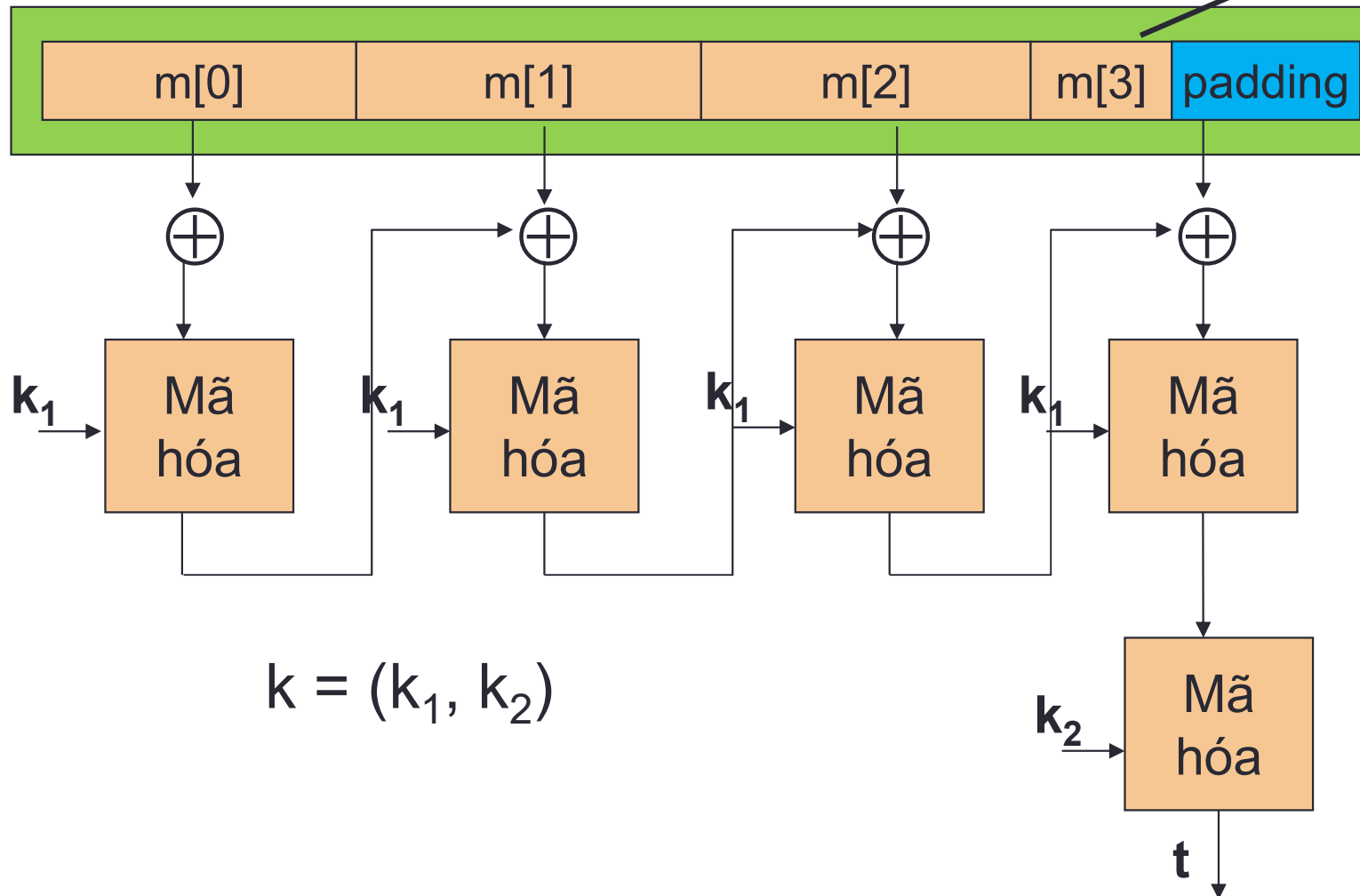
# rawCBC-Tấn công chọn trước bản rõ



Vấn đề:  $S(k, m || t \oplus m) = S(k, S(k, m) \oplus (t \oplus m)) =$   
 $S(k, t \oplus (t \oplus m)) =$   
 $S(k, m) = t$

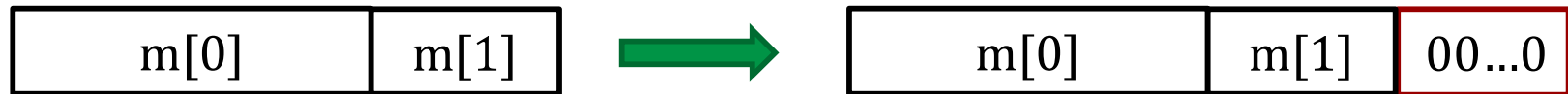
# Xây dựng MAC: CBC-MAC

Kích thước thông điệp không chia hết cho kích thước một khối?

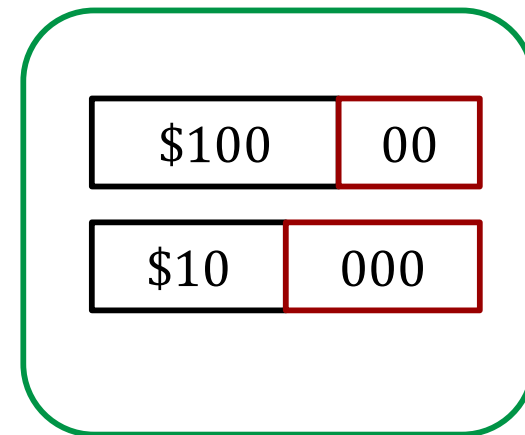


# Padding cho CBC-MAC

- Ý tưởng 1: Thêm vào các bit 0

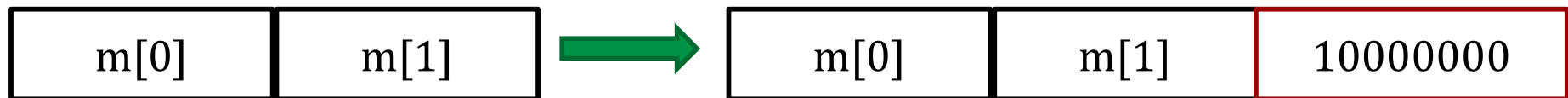


- Không an toàn. Ví dụ:



# Padding cho CBC-MAC

- Yêu cầu:  $M_i \neq M_j$  thì  $\text{pad}(M_i) \neq \text{pad}(M_j)$
- Chuẩn ISO/IEC 9797-1:
  - Sử dụng chuỗi padding bắt đầu bởi bit 1
  - Nếu kích thước thông điệp là bội số kích thước của khối, luôn thêm 1 khối padding



# An toàn của CBC-MAC

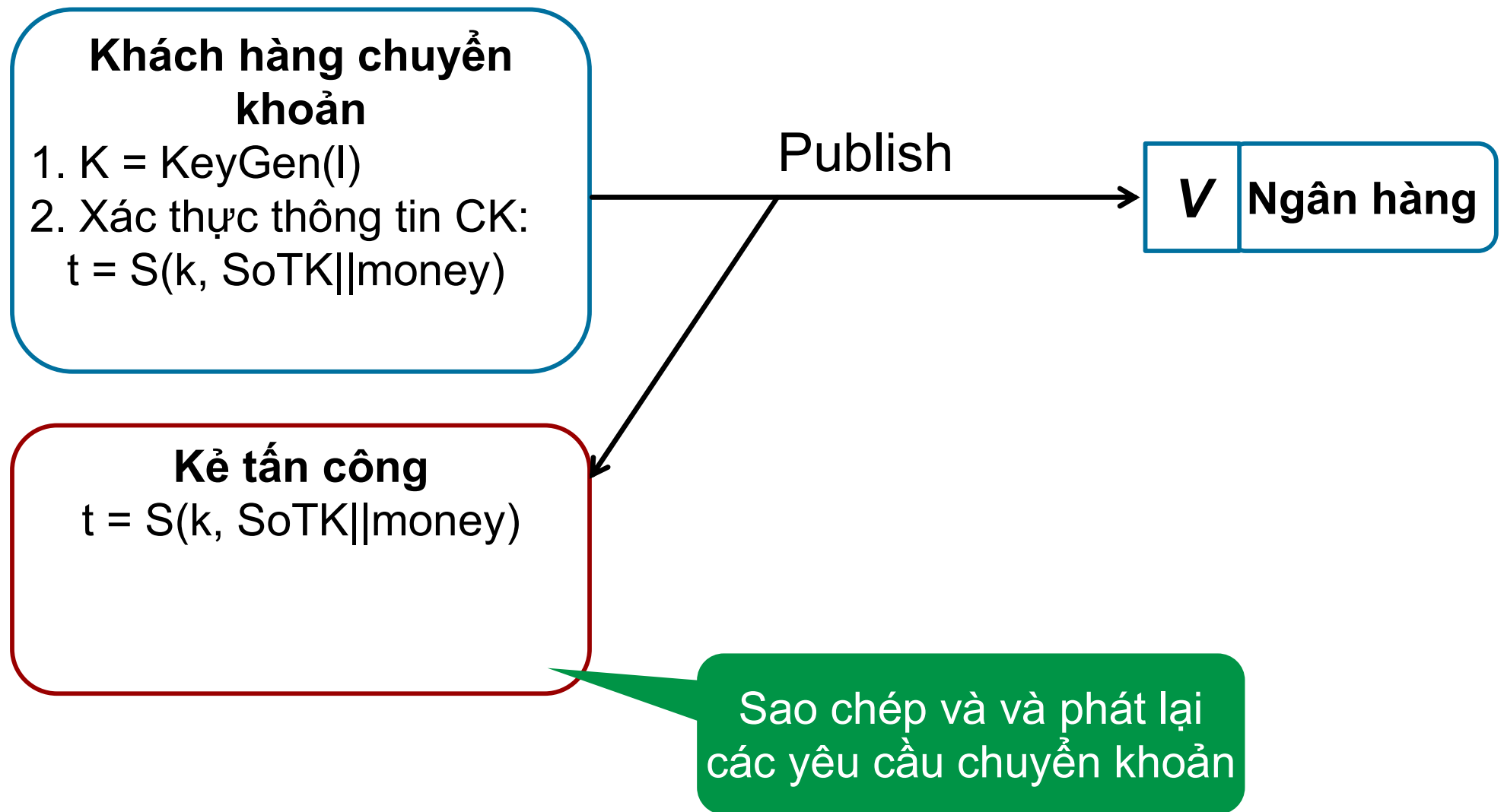
- Khóa được dùng nhiều lần  $\rightarrow$  giảm độ an toàn
- Nếu gọi:
  - $q$ : số bản tin được tính MAC cùng với khóa không đổi
  - $|X|$ : Số lượng giá trị có thể của  $t$
- Xác suất tấn công thành công  $\leq 2^*q^2 / |X|$
- Để xác suất tấn công là không đáng kể ( $\leq 2^{-80}$ ) thì sau bao nhiêu lần tính MAC phải đổi khóa?
- Ví dụ: Sử dụng AES-CBC-MAC:  
 $2^*q^2 / 2^{128} \leq 2^{-80} \rightarrow q = ?$



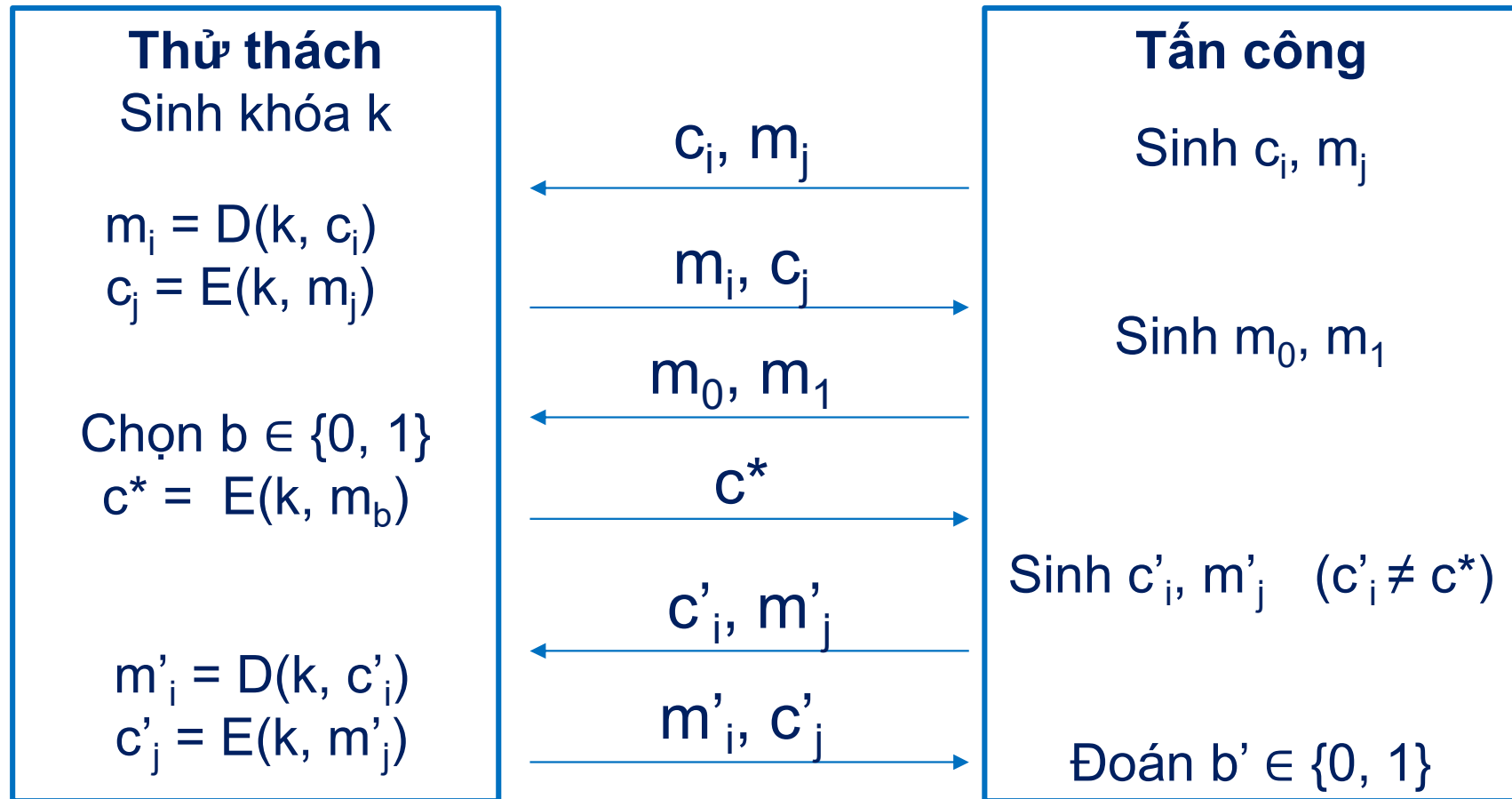
# Tấn công phát lại (Replay attack)

- Kẻ tấn công phát lại bản tin  $M$  đã được chứng thực trong phiên truyền thông trước đó
- Thiết kế MAC không chống tấn công phát lại  
→ cần thêm các yếu tố chống tấn công phát lại trong các giao thức truyền thông sử dụng MAC
- Một số kỹ thuật chống tấn công phát lại:
  - Giá trị dùng 1 lần (nonce):  $S(k, m \parallel \text{nonce})$
  - Tem thời gian:  $S(k, m \parallel \text{timestamp})$

# Tấn công phát lại



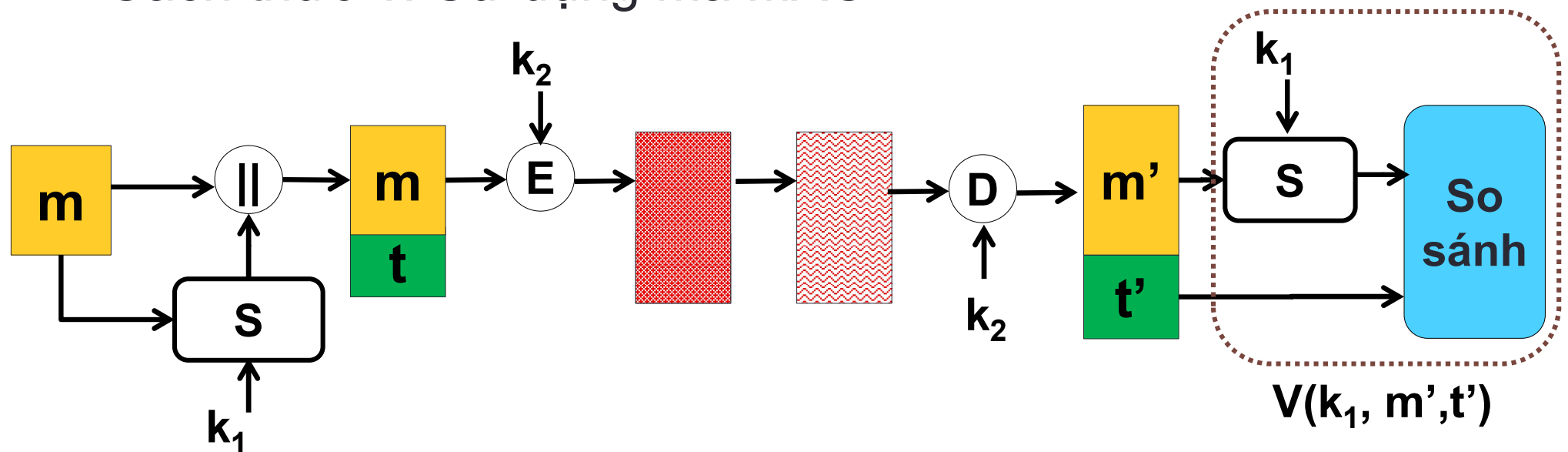
# Tấn công CCA – Nhắc lại



- Hệ mật chống lại được tấn công CCA (độ an toàn IND-CCA) nếu với mọi thuật toán tấn công hiệu quả thì  $P(b' = b) \leq \frac{1}{2} + \epsilon$

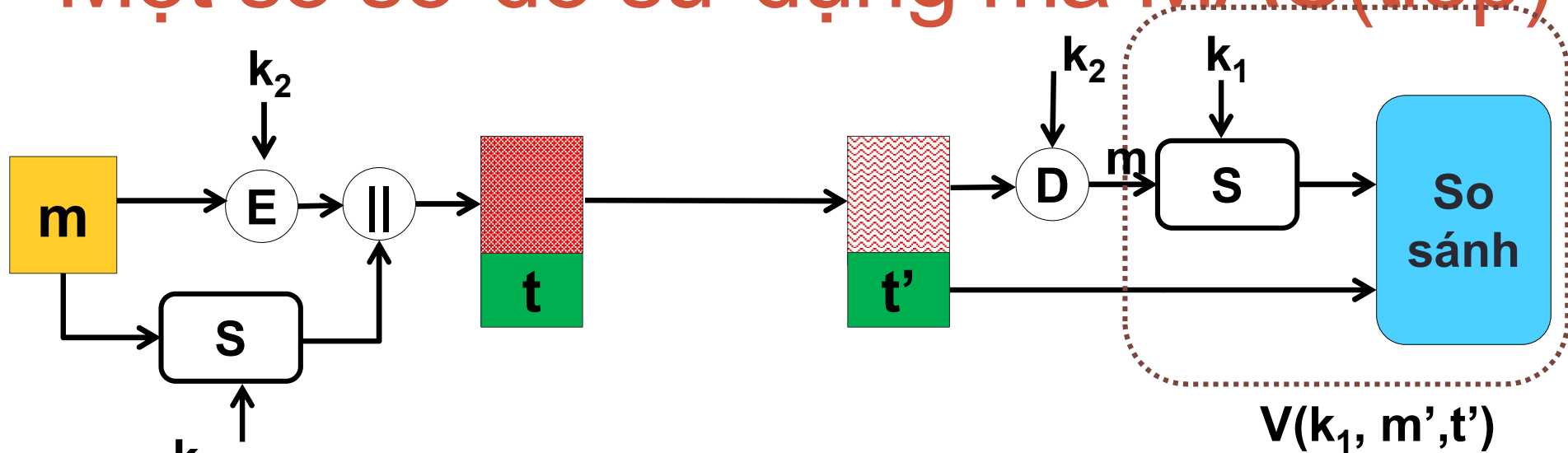
# Một số sơ đồ sử dụng mã MAC

- Các sơ đồ mật mã đã xem xét không chống lại được tấn công CCA(chosen-cipher attack)
  - Cách thức chung: kẻ tấn công sửa bản mã  $c^*$  thành  $c'_i$  và yêu cầu giải mã
- Mật mã có xác thực: sơ đồ mật mã đảm bảo đồng thời tính bí mật và toàn vẹn
- Cách thức 1: Sử dụng mã MAC

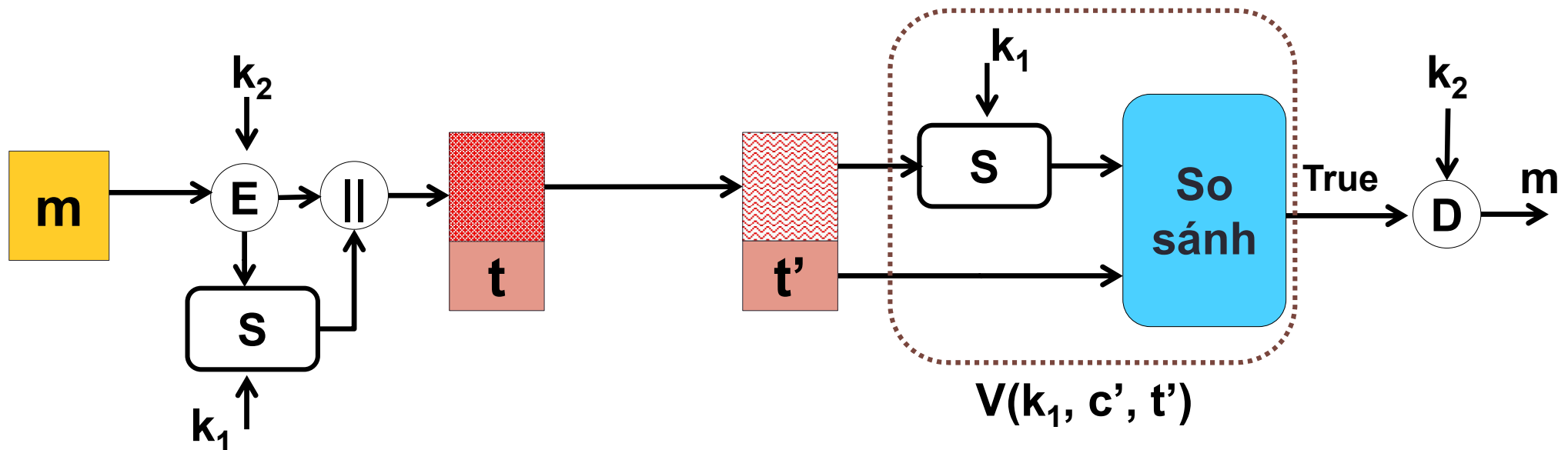


a) Xác thực bằng MAC, bảo mật bằng mật mã khóa đối xứng(SSL)

# Một số sơ đồ sử dụng mã MAC(tiếp)



b) Xác thực bằng MAC, bảo mật bằng mật mã khóa đối xứng (SSH)



c) Xác thực bằng MAC, bảo mật bằng mật mã khóa đối xứng (IPSec)<sup>28</sup>

# Nhận xét

## Sơ đồ a

- Xác thực toàn vẹn bản rõ
- Không xác thực toàn vẹn bản mật(không phát hiện tấn công thay thế bản mật)
- Không có thông tin về bản rõ từ MAC
- Chỉ đảm bảo an toàn CCA khi mã ở chế độ rand-CBC hoặc rand-CTR

## Sơ đồ b

- Xác thực toàn vẹn bản rõ
- Không xác thực toàn vẹn bản không phát hiện bản mật bị thay thế)
- MAC chứa thông tin bản rõ
- Có thể giảm sự an toàn mã mật

## Sơ đồ c

- Xác thực toàn vẹn bản rõ
- Xác thực toàn vẹn bản mật(có thể phát hiện bản mật bị thay thế)
- MAC không chứa thông tin bản rõ
- Luôn đảm bảo an toàn CCA

# Cách thức 2: Thêm dữ liệu xác thực

- Là các phương pháp mật mã

Hàm mã hóa  $E: K \times M \times N \rightarrow C$

Hàm giải mã  $D: K \times C \times N \rightarrow M \cup \{\perp\}$

Từ chối giải mã các bản mã không hợp lệ

- Trong đó  $N$  là một dữ liệu được thêm vào để xác thực toàn vẹn của bản tin
- Một số chuẩn:
  - GCM: Mã hóa ở chế độ CTR sau đó tính CW-MAC
  - CCM: Tính CBC-MAC sau đó mã hóa ở chế độ CTR (802.11i)
  - EAX: Mã hóa ở chế độ CTR sau đó tính CMAC

## 3.HÀM BẮM

---



# Khái niệm

- *Hàm băm H*: thực hiện phép biến đổi:
  - Đầu vào: bản tin có kích thước bất kỳ
  - Đầu ra: giá trị *digest*  $h = H(m)$  có kích thước  $n$  bit cố định (thường nhỏ hơn rất nhiều so với kích thước bản tin đầu vào)
- Chỉ thay đổi 1 bit đầu vào, làm thay đổi hoàn toàn giá trị đầu ra
- Ví dụ:
  - Đầu vào: “The quick brown fox jumps over the lazy **d**og”
  - Mã băm: 2fd4e1c67a2d28fced849ee1bb76e7391b93eb12
  - Đầu vào: “The quick brown fox jumps over the lazy **c**og”
  - Đầu ra: de9f2c7fd25e1b3afad3e85a0bd17d9b100db4b3

# Một hàm băm đơn giản

- Chia thông điệp thành các khối có kích thước n-bit

➤ Padding nếu cần

- Thực hiện XOR tất cả các khối → mã băm có kích thước n bit
- Tất nhiên, hàm băm này không đủ an toàn để sử dụng trong bài toán xác thực thông điệp

$$m = \begin{bmatrix} m_1 \\ m_2 \\ \dots \\ m_l \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & \dots & m_{1n} \\ m_{21} & m_{22} & \dots & m_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{l1} & m_{l2} & \dots & m_{ln} \end{bmatrix}$$
$$\begin{matrix} \oplus & \oplus & \oplus & \oplus \\ \Downarrow & \Downarrow & \Downarrow & \Downarrow \\ [c_1 & c_2 & \dots & c_n] = H(m) \end{matrix}$$

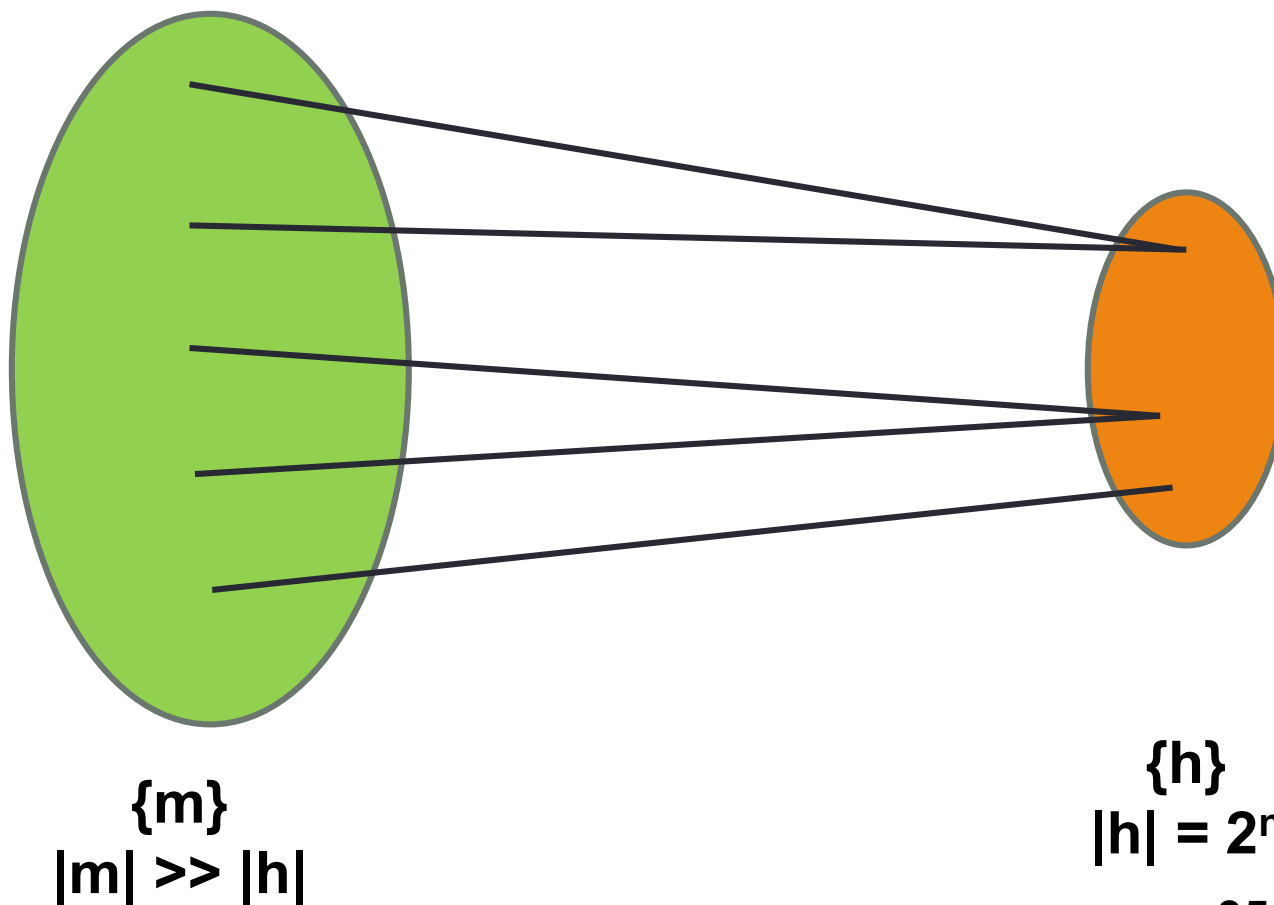
# Yêu cầu đối với hàm băm mật mã

1. Có thể áp dụng với thông điệp  $m$  với độ dài bất kỳ
2. Tạo ra giá trị băm  $h$  có độ dài cố định
3.  $H(m)$  dễ dàng tính được với bất kỳ  $m$  nào
4. Từ  $h$  rất khó tìm được  $m$  sao cho  $h = H(m)$ : tính một chiều
5. Biết trước  $m_1$  rất khó tìm được  $m_2$  sao cho  $H(m_1) = H(m_2) \rightarrow$  tính chống đụng độ yếu
6. Rất khó tìm được cặp  $(m_1, m_2)$  sao cho  $H(m_1) = H(m_2) \rightarrow$  tính chống đụng độ mạnh

# Tấn công vào tính 1 chiều

- Mã băm có kích thước  $n$  bit
- Cho trước  $h$  có kích thước  **$n$  bit**  $\rightarrow$  tìm được  $m$  sao cho  $H(m) = h$

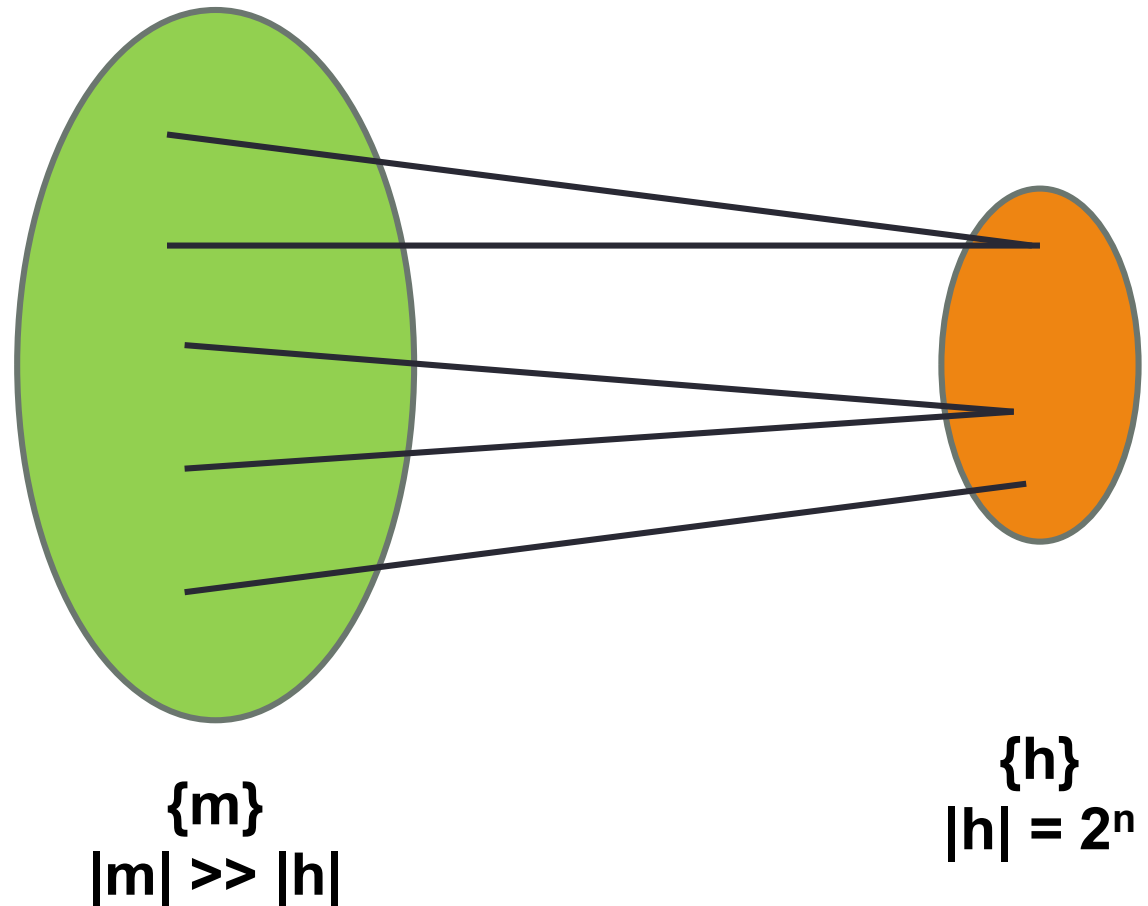
Chọn ra  $2^n$  bản tin ngẫu nhiên và tính mã băm của chúng.



# Tấn công vào tính độ yếu

- Mã băm có kích thước  $n$  bit
- Cho trước  $m_1 \rightarrow$  tìm được  $m_2$  sao cho  $H(m_2) = H(m_1)$

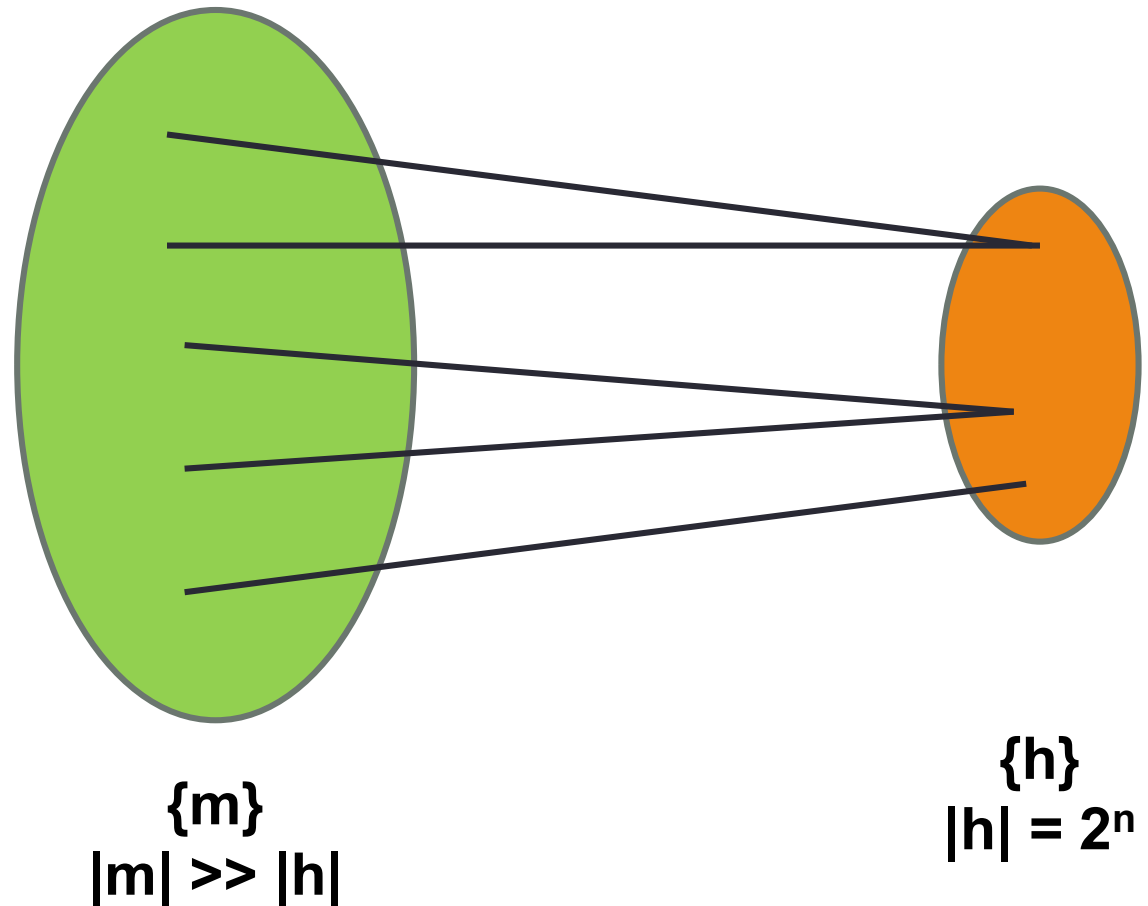
Chọn ra  $2^n$  bản tin  $m_2$  ngẫu nhiên và tính mã băm của chúng.



# Tấn công vào tính độ mạnh

- Mã băm có kích thước  $n$  bit
- Tìm được  $m_1, m_2$  sao cho  $H(m_2) = H(m_1)$

Chọn  $2^n + 1$  bản tin ngẫu nhiên và tính mã băm của chúng.



# Ví dụ

- Giả sử kẻ tấn công có khả năng tính toán 1.000.000 mã băm mỗi giây. Kích thước mã băm là bao nhiêu để kẻ tấn công thực hiện 100 năm, xác suất thành công cao nhất là  $2^{-80}$
- Lời giải:
  - Kích thước mã băm:  $n$  bit  $\rightarrow$  số mã băm có thể là  $2^n$
  - Số mã băm kẻ tấn công tính được trong 100 năm là:  $X$
  - Thỏa mãn:  $X/2^n \leq 2^{-80}$

# Một số hàm băm phổ biến

- MD5

- Kích thước digest: 128 bit
- Công bố thuật toán tấn công đụng độ (collision attack) vào 1995
- Năm 2005 tấn công thành công

- SHA-1

- Kích thước digest: 160 bit
- Công bố tấn công thành công vào năm 2015
- Hết hạn vào năm 2030

- SHA-2: 224/256/384/512 bit

- SHA-3: 224/256/384/512 bit



# MD5

- Bước 1: Padding dữ liệu sao cho bản tin đầu vào có độ dài  $L$  sao cho  $L \bmod 512 = 448$
- Bước 2: Biểu diễn độ dài của dữ liệu ban đầu dưới dạng 64 bit. Thêm giá trị độ dài này vào khối dữ liệu.
  - Coi dữ liệu là một chuỗi các khối 512 bit:  $Y_0, Y_1, \dots, Y_{K-1}$
  - Hoặc là một chuỗi các khối 32 bit :  $m_0, m_1, \dots, m_N$
- Bước 3: Khởi tạo các giá trị hằng số  $A, B, C, D$ 
  - $A = 0x67\ 45\ 23\ 01$
  - $B = 0xEF\ CD\ AB\ 89$
  - $C = 0x98\ BA\ DC\ FE$
  - $D = 0x10\ 32\ 54\ 76$

# MD5

- Bước 4: Thực hiện vòng lặp xử lý các khối 512 bit

$$CV_{q+1} = F(Y_q, CV_q)$$

- Xử lý khối dữ liệu  $Y_q$ : thực hiện 4 vòng lặp. Mỗi vòng lặp áp dụng hàm nén với  $T[1..64]$  là mảng hằng số xác định trước
- Cộng modulo  $2^{32}$  mỗi khối với giá trị  $CV_q$  để có  $CV_{q+1}$
- Bước 5: Kết quả xử lý khối 512 bit cuối cùng là giá trị băm của thông điệp

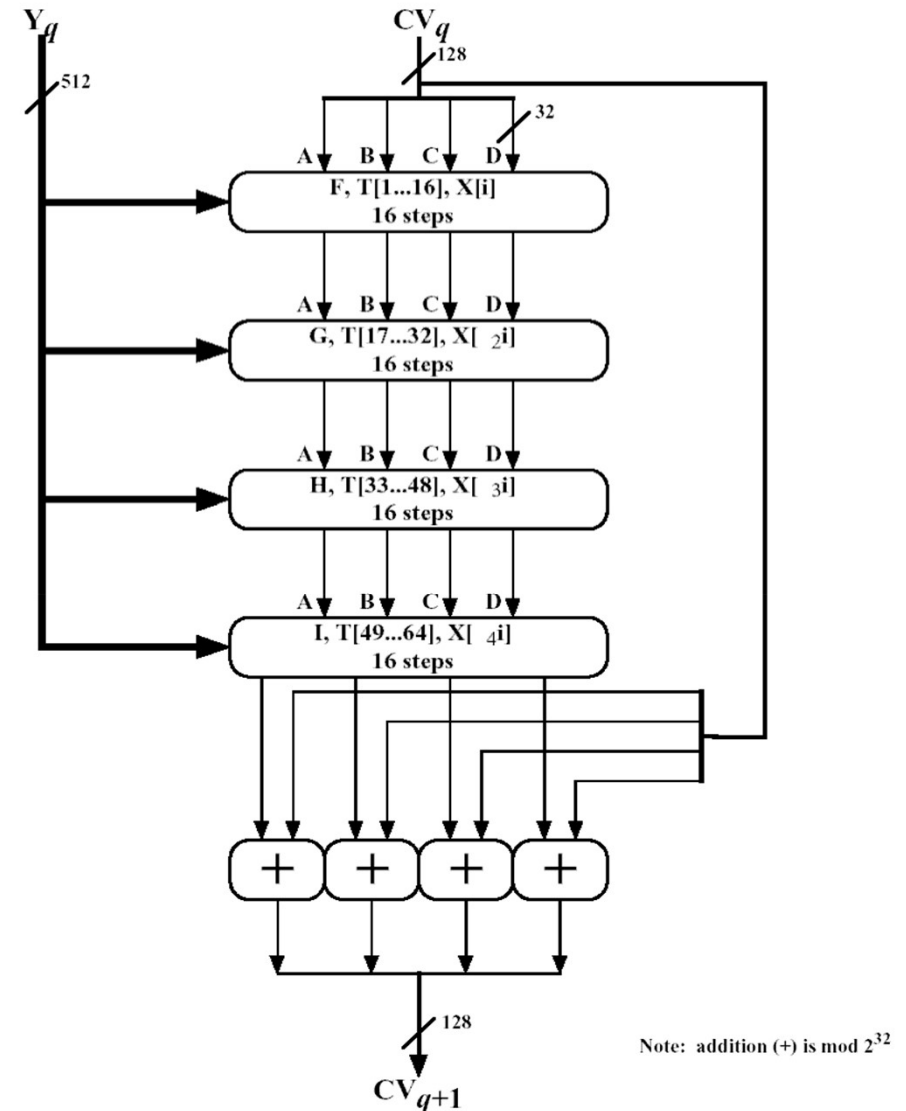

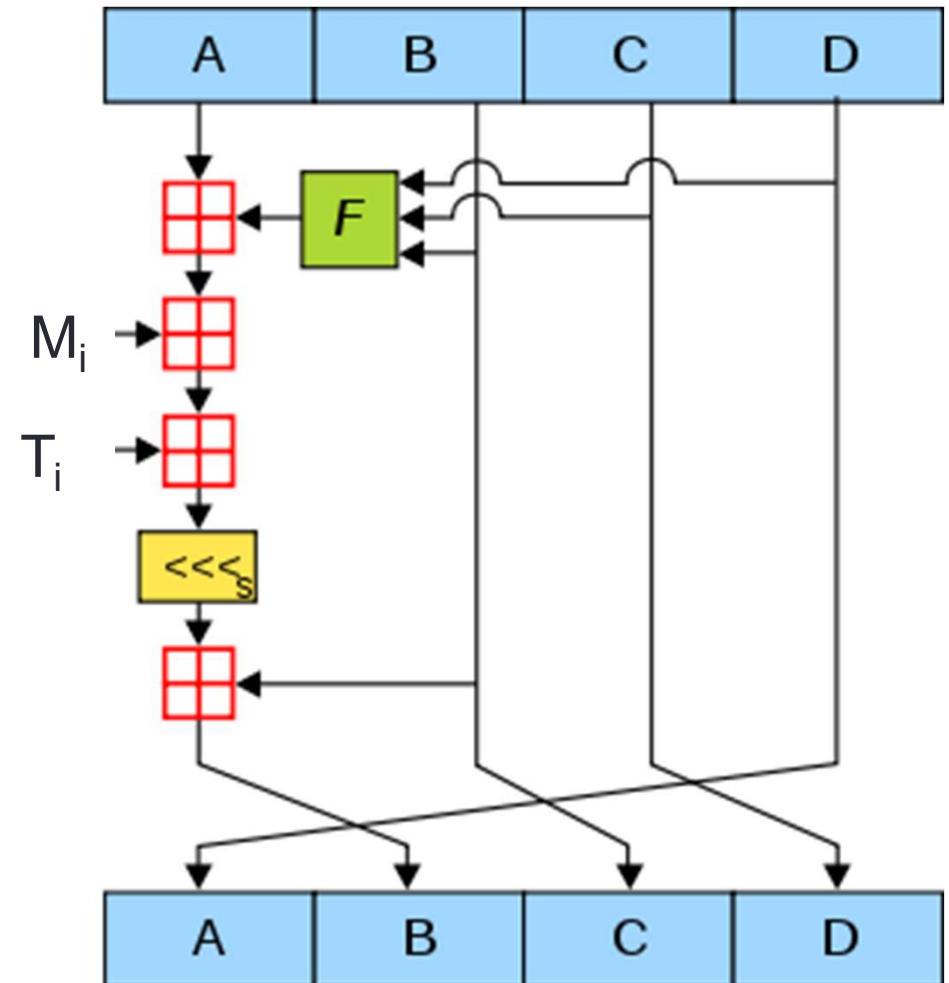


Figure 9.2 MD5 Processing of a Single 512-bit Block (MD5 Compression Function)

# Hàm nén trong MD5 (Tham khảo)

- Đầu vào:
    - CV: Khối 128 bit
      - $M_i$ : khối dữ liệu 32-bit
      - $T_i$ : Hằng số
  -  Cộng modulo  $2^{32}$
  - $\lll s$ : dịch trái  $s$  bit
  - $\wedge$ : AND,  $\vee$ : OR,  $\neg$ : NOT
- $$F1 = (B \wedge C) \vee (\neg B \wedge D)$$
- $$F2 = (B \wedge D) \vee (C \wedge \neg D)$$
- $$F3 = B \oplus C \oplus D$$
- $$F4 = C \oplus (B \vee \neg D)$$
- Thực hiện vòng lặp 16 bước



# SHA-1

- Bước 1: Padding dữ liệu sao cho bản tin đầu vào có độ dài  $L$  sao cho  $L \bmod 512 = 448$
- Bước 2: Biểu diễn độ dài của dữ liệu ban đầu dưới dạng 64 bit. Thêm giá trị độ dài này vào khối dữ liệu.
  - Coi dữ liệu là một chuỗi các khối 512 bit:  $Y_0, Y_1, \dots, Y_{K-1}$
  - Hoặc là một chuỗi các khối 32 bit :  $m_0, m_1, \dots, m_N$
- Bước 3: Khởi tạo các giá trị hằng số  $A, B, C, D, E$ 
  - $A = 0x67\ 45\ 23\ 01$
  - $B = 0xEF\ CD\ AB\ 89$
  - $C = 0x98\ BA\ DC\ FE$
  - $D = 0x10\ 32\ 54\ 76$
  - $E = 0xC3\ D2\ E1\ F0$

# SHA-1

- Bước 4: Thực hiện vòng lặp xử lý các khối 512 bit

$$CV_{q+1} = F(Y_q, CV_q)$$

- Xử lý khối dữ liệu  $Y_q$ : thực hiện 4 vòng lặp. Mỗi vòng lặp áp dụng hàm nén với  $K$  là hằng số xác định trước
- Cộng modulo  $2^{32}$  mỗi khối với giá trị  $CV_q$  để có  $CV_{q+1}$
- Bước 5: Kết quả xử lý khối 512 bit cuối cùng là giá trị băm của thông điệp

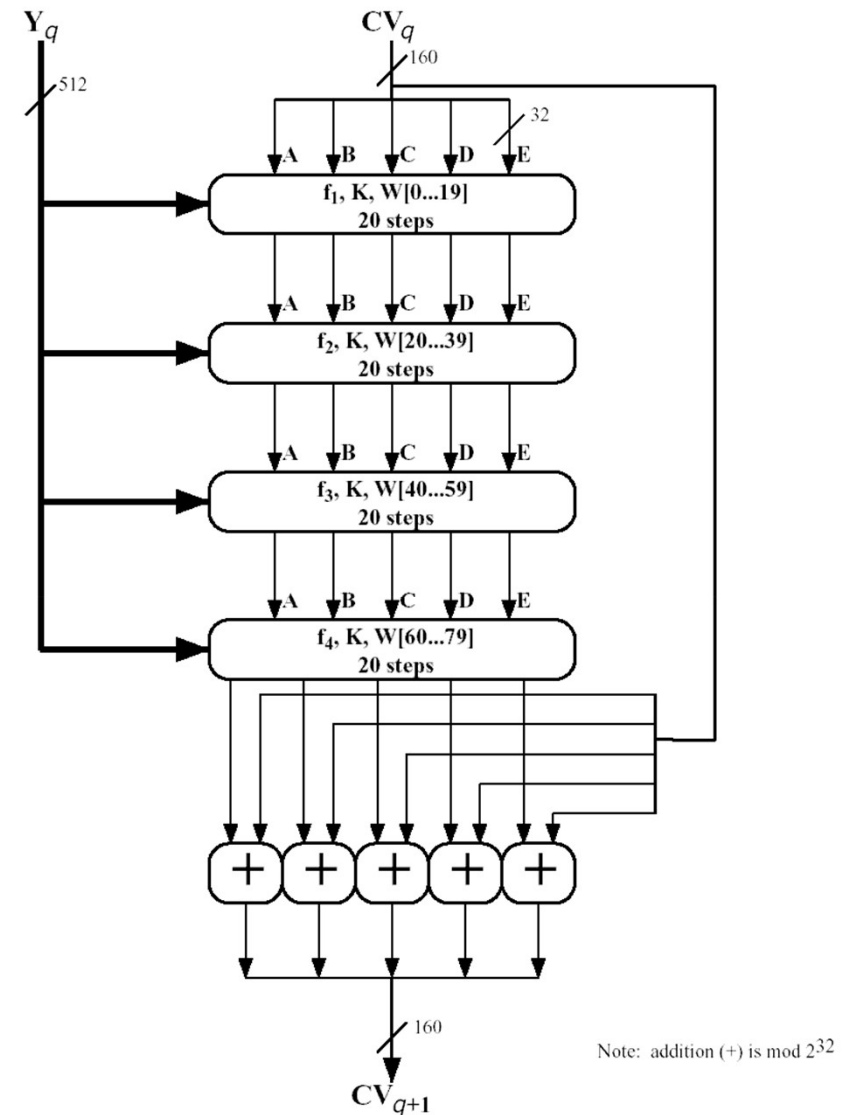


Figure 9.5 SHA-1 Processing of a Single 512-bit Block (SHA-1 Compression Function)

# Hàm nén trong SHA-1 (Tham khảo)

- Đầu vào:
  - CV: Khối 160 bit
  - $W_t$ : Khối dữ liệu mở rộng 32 bit
  - $K_t$ : Hằng số

- $\boxplus$ : Cộng modulo  $2^{32}$
- $\lll 5(30)$ : dịch trái 5(30) bit
- $\wedge$ : AND,  $\vee$ : OR,  $\neg$ : NOT

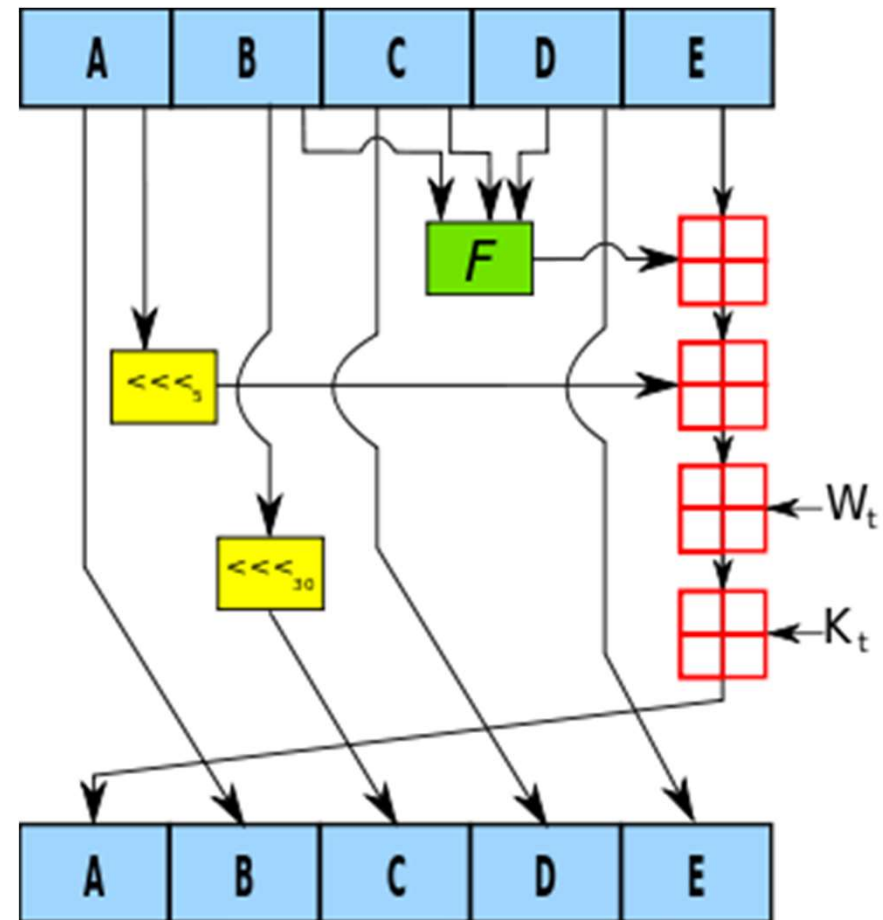
$$F1 = (B \wedge C) \vee (\neg B \wedge D)$$

$$F2 = B \oplus C \oplus D$$

$$F3 = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$$

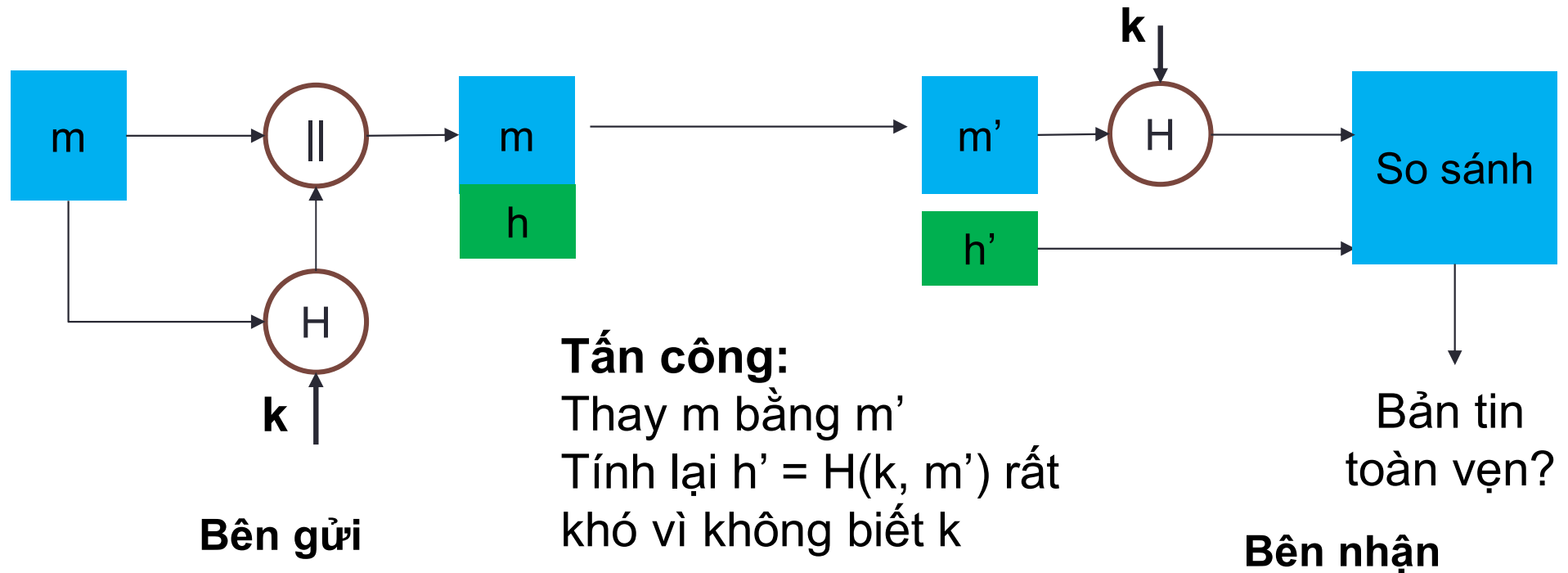
$$F4 = B \oplus C \oplus D$$

- Thực hiện vòng lặp 20 bước



# Sử dụng mã băm

- Xác thực toàn vẹn bản tin: chỉ phát hiện được lỗi ngẫu nhiên trong quá trình truyền



- Xác thực toàn vẹn bản tin khi có tấn công chủ động: băm bản tin cùng với khóa bí mật đã chia sẻ trước

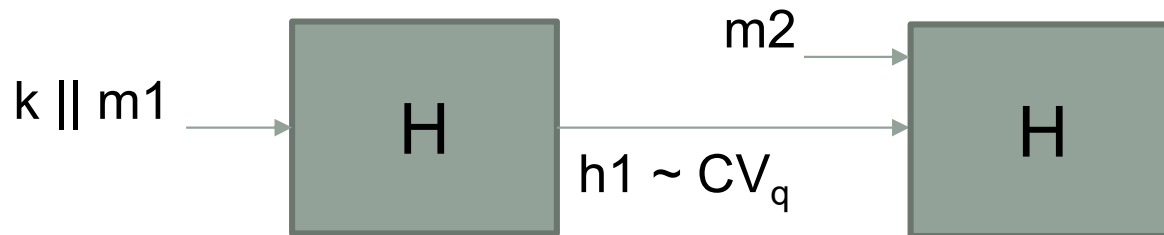
# Các ý tưởng sử dụng hàm băm

- Ý tưởng 1:
  - $m = m1 \parallel m2$
  - $H(m) = H(m1) \parallel H(m2)$
- Ý tưởng 2:
  - $t = S(k, m)$
  - $H(m) = H(t)$
- Ý tưởng 3:
  - $H(k, m) = H(k \parallel m)$ ,  $k$  là khóa bí mật chia sẻ trước



# $H(k \parallel m)$ có an toàn?

- Tấn công mở rộng kích thước (Length extension attack)
- Có thể tính được  $H(k \parallel (m1 \parallel m2))$  nếu biết  $\text{length}(k)$ ,  $\text{length}(m1)$  và  $h1 = H(k \parallel m1)$ 
  - Kẻ tấn công có thể thay  $m1$  bằng  $m1 \parallel m2$  và tính được mã băm dù không biết  $k$
- Các hàm băm bị ảnh hưởng: MD5, SHA-1, SHA-2



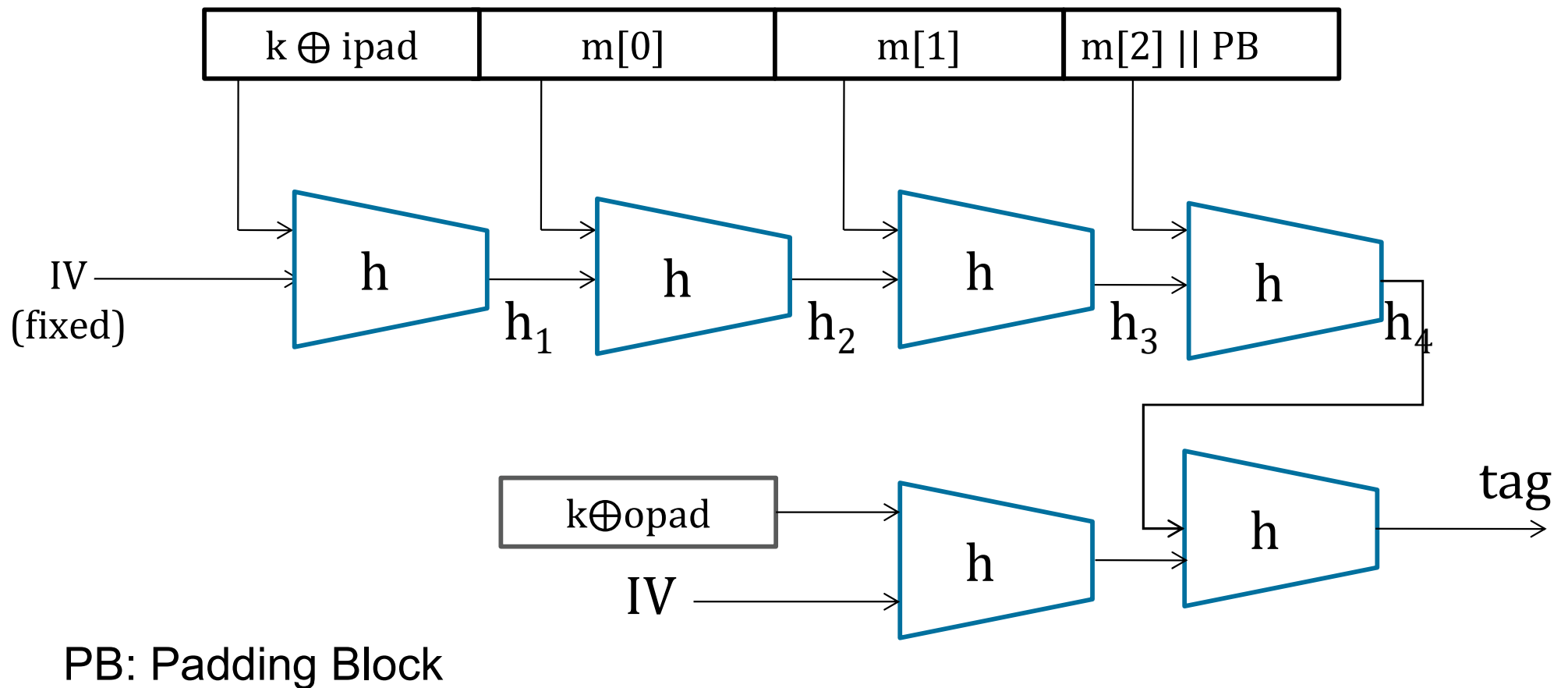
$$m1 \parallel H(k \parallel m1) \\ \rightarrow (m1 \parallel m2) \parallel H(k \parallel m1 \parallel m2)$$

- Thay đổi:  $H(m \parallel k)$
- An toàn hơn: sử dụng HMAC

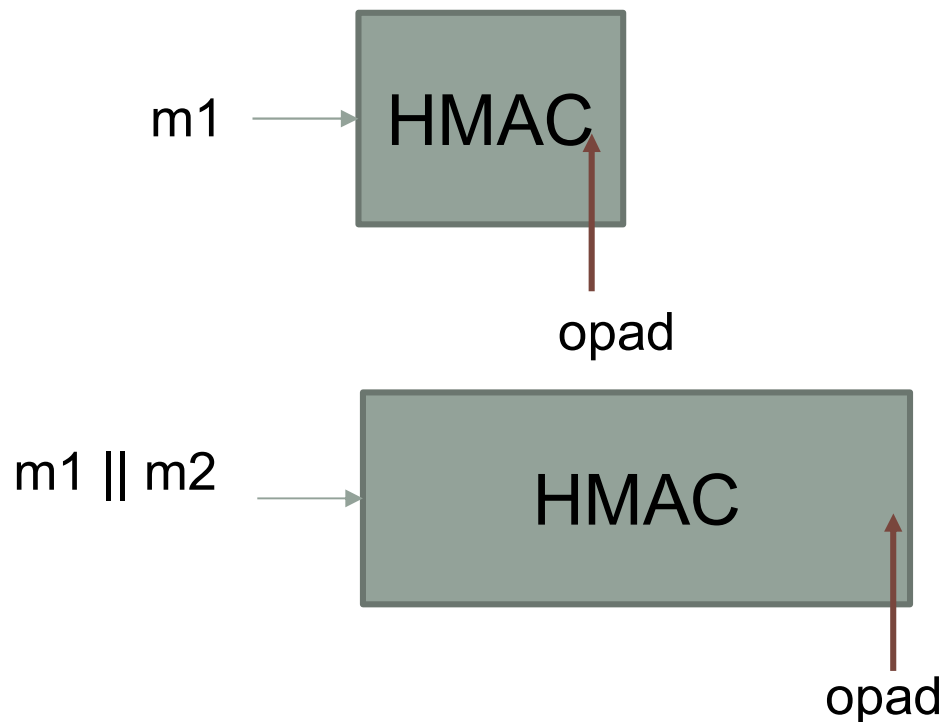
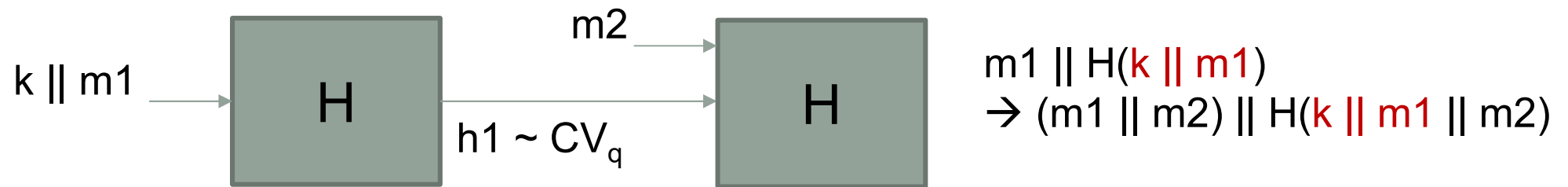
$$m1 \parallel H(m1 \parallel k) \\ \rightarrow (m2 \parallel m1) \parallel H(m2 \parallel m1 \parallel k)$$

# HMAC

- Hashed MAC: xây dựng MAC dựa trên hàm băm



# HMAC chống lại length extension attack



# HMAC và MAC

- HMAC có tính chống đụng độ chắc chắn hơn do sử dụng hàm băm
- Tốc độ tính toán của HMAC nhanh hơn
- Kích thước giá trị tag được tạo bởi HMAC lớn hơn → an toàn hơn trước các tấn công

## 4. TẦN CÔNG VÀO TÍNH ĐỤNG ĐỘ

---

# Tấn công vét cạn

- Đụng độ trong MAC: tồn tại  $m_1 \neq m_2$  mà  $S(k, m_1) = S(k, m_2)$
- Đụng độ trong hàm băm: tồn tại  $m_1 \neq m_2$  mà  $H(m_1) = H(m_2)$
- Nhắc lại: sự tồn tại các bản tin đụng độ dẫn đến các nguy cơ tấn công vào sơ đồ xác thực
- Tìm ra bản tin đụng độ: có thể thực hiện vét cạn  
→ số bản tin cần tính tối thiểu là bao nhiêu sẽ chắc chắn thành công?

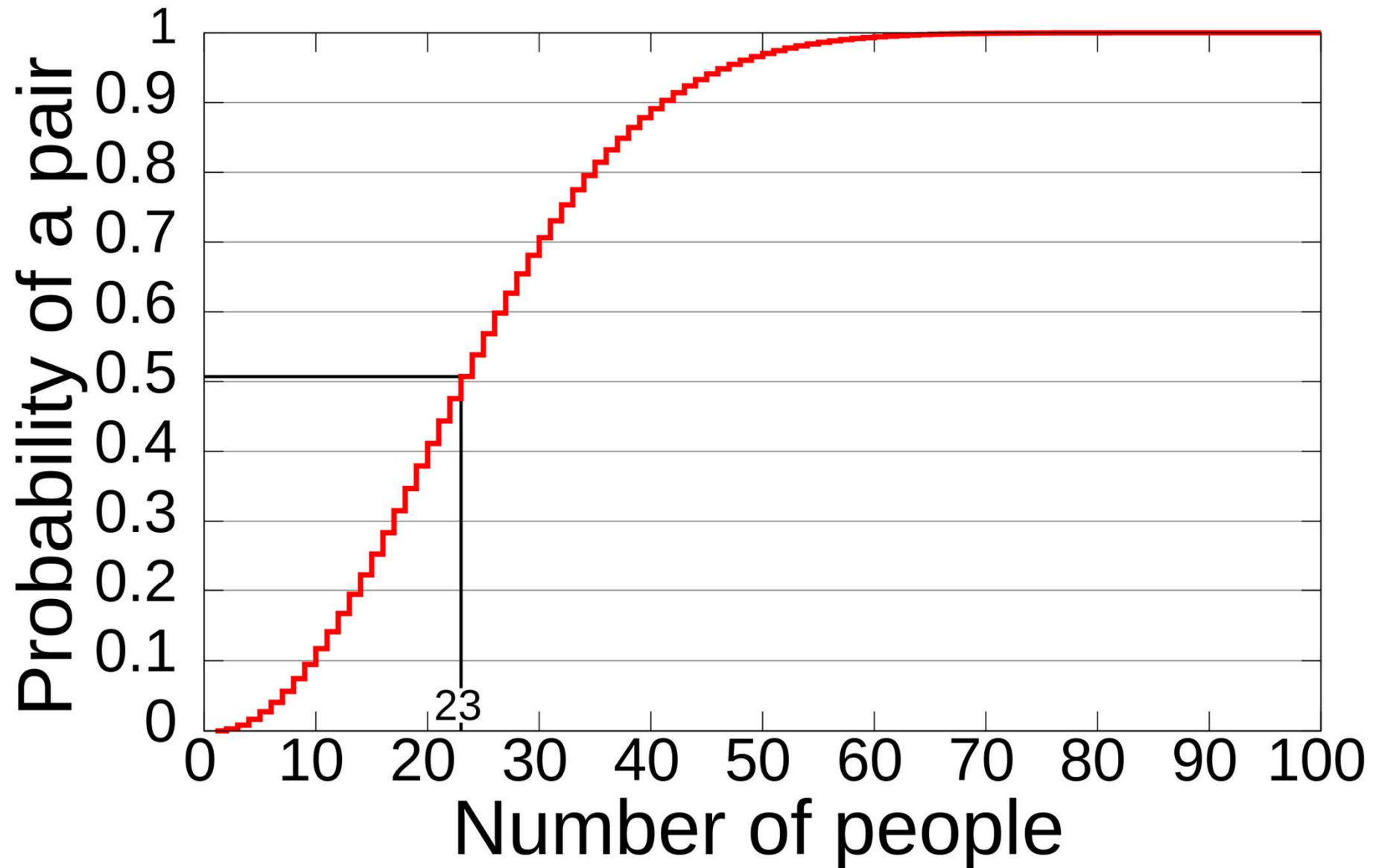
# Nghịch lý ngày sinh (Birthday paradox)

- Bài toán: Khi chọn  $n$  người bất kỳ, xác suất để có tối thiểu 2 người có trùng ngày sinh là bao nhiêu?
- Số cách chọn ra  $n$  người bất kỳ:  $365^n$
- Số cách chọn ra  $n$  người không có cặp nào trùng ngày sinh:  $365 \times 364 \times \dots \times (365 - (n - 1)) = C^n_{365}$
- Xác suất để chọn ra  $n$  người không có cặp nào trùng ngày sinh

$$Q = \frac{365 \times 364 \times \dots \times (365 - (n - 1))}{365^n}$$

- Xác suất cần tính:  $P = 1 - Q$
- $n = ?$  để  $P > 0.5$  (cứ 2 lần chọn thì có 1 lần thỏa mãn)

# Nghịch lý ngày sinh





# Tấn công dựa trên nghịch lý ngày sinh (Birthday paradox attack)

- Kiểm tra  $2^{n/2}$  bản tin có xác suất tìm ra các bản tin đụng độ là  $\sim 0.5$
- Cách thức tấn công:
  - Bước 1: Chọn ra  $2^{n/2}$  bản tin ngẫu nhiên
  - Bước 2: Tính mã băm/MAC cho các bản tin
  - Bước 3: Kiểm tra sự tồn tại của các bản tin đụng độ. Nếu không có, quay lại bước 1.
  - Kỳ vọng: thành công sau 2 lần thử

# CASE STUDY: BẢO MẬT TRÊN IPHONE

---

# Bảo mật trên iPhone

- Triết lý bảo mật của Apple:
  - Nếu điện thoại trong tay bạn, bạn có thể truy cập mọi thứ
  - Nếu điện thoại trong tay người khác, nó chỉ là một “cục gạch”
- Apple triển khai tất cả các cơ chế bảo mật trên chip Secure Enclave Processor (SEP)
  - Secure Enclave Processor và TCB của thiết bị
- Mọi phần còn lại của thiết bị được coi là không tin cậy
  - Bộ nhớ là không tin cậy, vì vậy mọi dữ liệu được mã hóa
  - CPU phải gửi yêu cầu Secure Enclave để giải mã
  - Một vài dữ liệu bí mật chỉ có thể đọc bởi Secure Enclave

# Bảo mật trên iPhone

- Khóa chủ  $K_{\text{phone}}$  được sử dụng để mã hóa các khóa mật mã khác
- $K_{\text{phone}}$  được mã hóa bởi  $K_{\text{user}}$ , sinh từ mật khẩu của người dùng
  - $K_{\text{user}} = \text{PBKDF}(\text{password}, \text{hardcode})$
  - hardcode có kích thước 256 bit được sinh và lưu trữ trên chip SEP
- Làm cách nào kẻ tấn công có thể đánh cắp được dữ liệu mà không có mật khẩu người dùng?