HÀM BĂM KHÁNG XUNG ĐỘT

https://class.coursera.org/crypto-preview/class/index

- *Giới thiệu*
- *Tấn công dùng nghịch lý ngày sinh*
- *Sơ đồ Merkle-Damgard*
- *Xây dựng hàm nén*
- *HMAC: MAC dựa trên SHA256*
- *Timing Attack cho MAC*

HÀM BĂM KHÁNG XUNG ĐỘT

https://class.coursera.org/crypto-preview/class/index

- *Giới thiệu*
- *Tấn công dùng nghịch lý ngày sinh*
- *Sơ đồ Merkle-Damgard*
- *Xây dựng hàm nén*
- *HMAC: MAC dựa trên SHA256*
- *Timing Attack cho MAC*

Nhắc lại: Toàn vẹn thông điệp

MAC xây dựng dựa trên PRF:

- ECBC-MAC, CMAC : Thường dùng với AES (Ví dụ, 802.11i)
- NMAC: làm cơ sở cho HMAC
- PMAC: một MAC song song

MAC ngẫu nhiên:

- Carter-Wegman MAC: dựa trên one-time MAC nhanh

Tiếp theo:

- xây dựng MAC dựa trên tính kháng xung đột

Tính kháng xung đột

Định nghĩa. Xét hàm băm $H: M \to T$ với $|M| >> |T|$. Một xung đột cho H là một cặp $m_0, m_1 \in M$ thỏa mãn :

$$H(m_0) = H(m_1) \quad \text{và} \quad m_0 \neq m_1$$

Định nghĩa. Hàm H được gọi là kháng xung đột nếu với mọi thuật toán "hiệu quả" (tường minh) A:

$$\text{Adv}_{CR}[A,H] = \Pr[\ A \text{ output xung đột cho } H\ ]$$

là "không đáng kể".

Ví dụ:

- SHA-256: Output là 256 bit.

Xây dựng MAC từ hàm kháng xung đột

Xây dựng. Xét $I = (S, V)$ là MAC cho thông điệp ngắn trên (K, M, T) (Ví dụ, AES). Xét hàm băm $H: M_{big} \rightarrow M$.

Ta định nghĩa $I_{big} = (S_{big}, V_{big})$ trên (K, M_{big}, T) như sau:

$$S_{big}(k,m) = S(k,H(m)) \quad ; \quad V_{big}(k, m, t) = V(k, H(m), t)$$

Định lý. Nếu I là một MAC an toàn và H hàm kháng xung đột, vậy thì I_{big} là MAC an toàn.

Ví dụ:

- $S(k, m) = \text{AES}_{\text{2-block-cbc}}(k, \text{SHA-256}(m))$ là một MAC an toàn.

Xây dựng MAC từ hàm kháng xung đột

$$S_{big}(k,m) = S(k,H(m)) \quad ; \quad V_{big}(k, m, t) = V(k, H(m), t)$$

Tính kháng xung đột là cần:

- Nếu kẻ tấn công có thể tìm được $m_0 \neq m_1$ sao cho $H(m_0) = H(m_1)$,
- vậy thì MAC không còn an toàn trước tấn công chọn 1 bản rõ:
  - bước 1: kẻ tấn công truy vấn $t \leftarrow\!\!- S(k, m_0)$
  - bước 2: output (m_1 , t) là cặp thông điệp/tag giả mạo

Bảo vệ sự toàn vẹn của file dùng hàm băm kháng xung đột

Gói phần mềm:

Không gian chung
(chỉ đọc)

$H(F_1)$ $H(F_2)$
...
$H(F_n)$

Toàn vẹn:

- Khi người dùng download file, chị ta có thể kiểm tra nội dung có khớp với mã băm
- H kháng xung đột ⇒ kẻ tấn công không thể sửa gói phần mềm mà không bị phát hiện
- Không cần khóa (mọi người đều có thể kiểm tra tính toàn vẹn), nhưng cần không gian lưu trữ công khai

HÀM BĂM KHÁNG XUNG ĐỘT

https://class.coursera.org/crypto-preview/class/index

- *Giới thiệu*
- *Tấn công dùng nghịch lý ngày sinh*
- *Sơ đồ Merkle-Damgard*
- *Xây dựng hàm nén*
- *HMAC: MAC dựa trên SHA256*
- *Timing Attack cho MAC*

Thuật toán tấn công hàm băm

- Xét hàm băm H: $M \rightarrow \{0,1\}^n$ với $|M| >> 2n$
- Thuật toán sau cho phép tìm xung đột sau $O(2^{n/2})$ lần băm.

Thuật toán:

1. Chọn $2^{n/2}$ thông điệp ngẫu nhiên trong M: $m_1, \ldots, m_{2^{n/2}}$

   (với xác suất chúng phân biệt là cao)

2. For i = 1, …, $2^{n/2}$:

   tính $t_i = H(m_i) \in \{0,1\}^n$.

3. Tìm xung đột $(t_i = t_j)$. Nếu không thấy thì quay lại bước 1.

- Khả năng thành công của thuật toán này như thế nào?

Nghịch lý ngày sinh nhật

Định lý. Xét các số nguyên $r_1, \ldots, r_n \in \{1,\ldots,B\}$ có phân phối giống nhau.

Khi $n = 1.2 \times B^{1/2}$ thì ta có

$$\Pr[\ \exists i \neq j:\ \ r_i = r_j\ ] \geq \ ½$$

Chứng minh. $\Pr[\exists i \neq j : r_i = r_j] = 1 - \Pr[\forall i \neq j : r_i \neq r_j]$

$$= 1 - \left(\frac{B-1}{B}\right)\left(\frac{B-2}{B}\right)\cdots\left(\frac{B-n+1}{B}\right)$$

$$= 1 - \prod_{i=1}^{n-1}\left(1 - \frac{i}{B}\right) \geq 1 - \prod_{i=1}^{n-1} e^{-i/B}$$

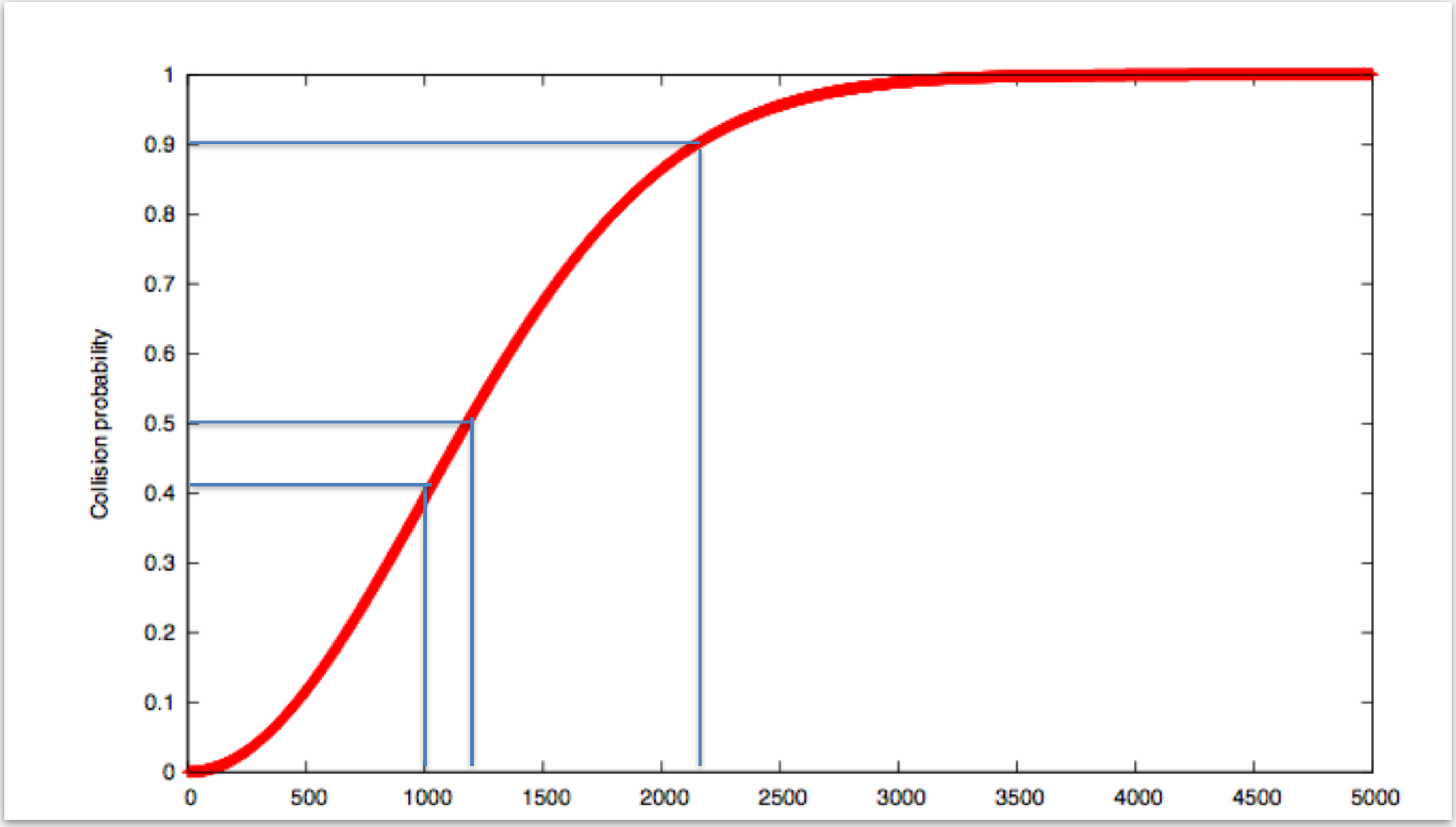
$e^x \geq 1 - x$

$$= 1 - e^{-1/B \sum_{i=1}^{n-1} i}$$

$$\geq 1 - e^{-n^2/(2B)}$$

$$\geq 1 - e^{-0.72} = 0.53$$

$B = 10^6$


Collision probability
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
samples n

Thuật toán tấn công hàm băm H: $M \rightarrow \{0,1\}^n$

Thuật toán:

1. Chọn $2^{n/2}$ thông điệp ngẫu nhiên trong M: $m_1, \ldots, m_{2^{n/2}}$

   (xác suất chúng phân biệt nhau là cao)

2. For $i = 1, \ldots, 2^{n/2}$:

   tính $t_i = H(m_i) \in \{0,1\}^n$.

3. Tìm xung đột $(t_i = t_j)$. Nếu không thấy thì quay lại bước 1.

- Kỳ vọng số vòng lặp cần thực hiện gần bằng 2
- Thời gian chạy: $O(2^{n/2})$ và không gian $O(2^{n/2})$

Một số hàm băm kháng xung đột

Crypto++5.6.0 [Wei Dai]

AMD Opteron, 2.2 GHz (Linux)

hàm	mã băm (số bit)	tốc độ (MB/giây)	thời gian tấn công
SHA-1	160	153	2^{80}
SHA-256	256	111	2^{128}
SHA-512	512	99	2^{256}
Whirlpool	512	57	2^{256}

* thuật toán tốt nhất tìm xung đột cho SHA-1 cần 2^{51} lần tính mã băm.

Thuật toán lượng tử

	Thuật toán cổ điển	Thuật toán lượng tử
Tấn công vét cạn hệ mã khối $E: K \times X \longrightarrow X$	$O(\lvert K \rvert)$	$O(\lvert K \rvert^{1/2})$
Tìm xung đột cho hàm băm $H: M \longrightarrow T$	$O(\lvert T \rvert^{1/2})$	$O(\lvert T \rvert^{1/3})$

HÀM BĂM KHÁNG XUNG ĐỘT

https://class.coursera.org/crypto-preview/class/index

- *Giới thiệu*
- *Tấn công dùng nghịch lý ngày sinh*
- *Sơ đồ Merkle-Damgard*
- *Xây dựng hàm nén*
- *HMAC: MAC dựa trên SHA256*
- *Timing Attack cho MAC*

Nhắc lại: Hàm băm kháng xung đột

Định nghĩa. Xét hàm băm $H: M \to T$ với $|M| >> |T|$. Một xung đột cho H là một cặp $m_0, m_1 \in M$ thỏa mãn :

$$H(m_0) = H(m_1) \quad \text{và} \quad m_0 \neq m_1$$

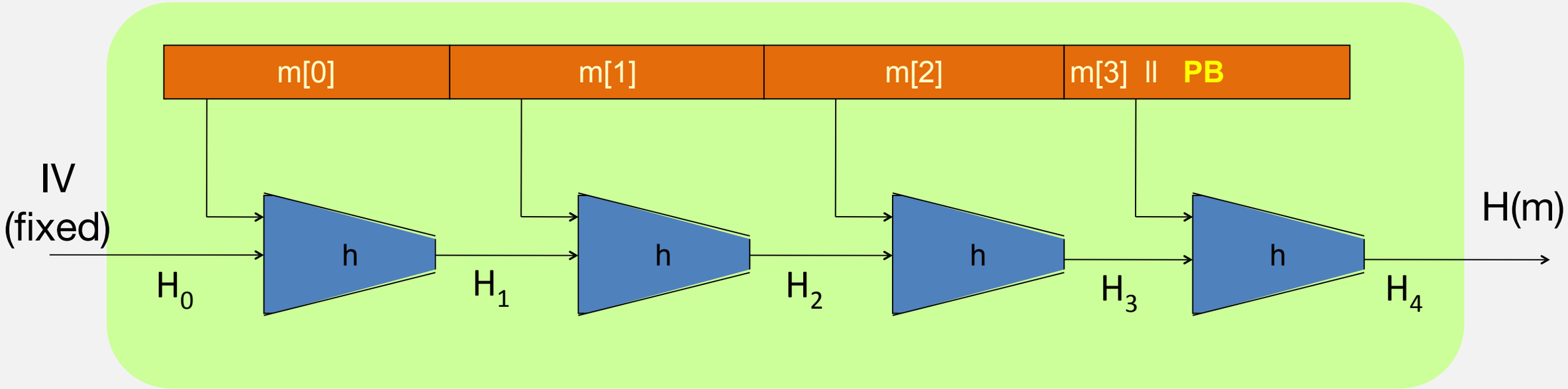
Mục đích:

- Xây dựng hàm băm kháng xung đột

Bước đầu tiên:

- Cho một hàm băm kháng xung đột cho thông điệp kích thước nhỏ,
- hãy xây dựng hàm băm cho thông điệp kích thước lớn.

Xây dựng theo sơ đồ Merkle-Damgard

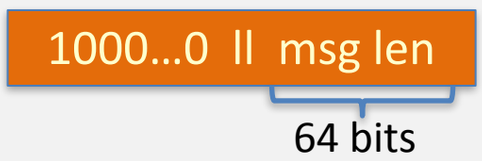


Cho trước hàm nén $h: T \times X \longrightarrow T$

Xây dựng:

- Hàm băm $H: X^{\leq L} \longrightarrow T$.

Padding:



- Nếu không đủ không gian cho padding, vậy thì thêm block mới.

Tính kháng xung đột cho sơ đồ MD

Định lý. Nếu h là kháng xung đột, vậy thì H xây dựng theo sơ đồ trước cũng là kháng xung đột.

Chứng minh.

- xung đột cho H $\Rightarrow$ xung đột cho h

Làm thế nào xây dựng được hàm băm kháng xung đột cho thông điệp kích thước nhỏ?

HÀM BĂM KHÁNG XUNG ĐỘT

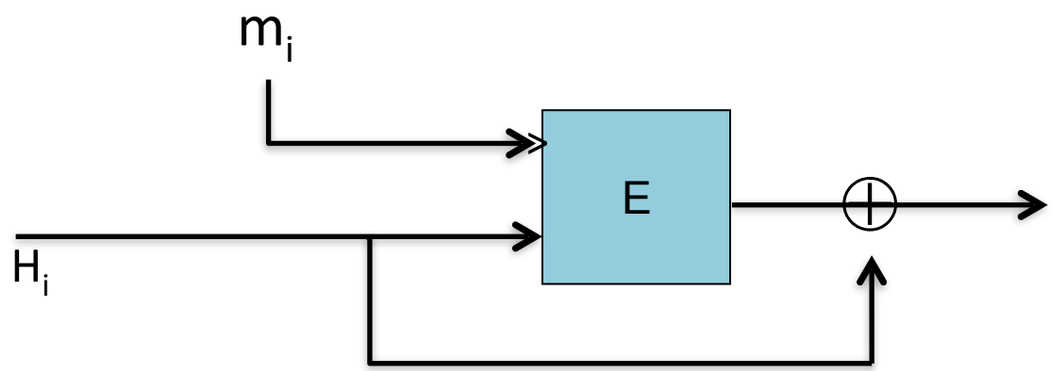
https://class.coursera.org/crypto-preview/class/index

- *Giới thiệu*
- *Tấn công dùng nghịch lý ngày sinh*
- *Sơ đồ Merkle-Damgard*
- *Xây dựng hàm nén*
- *HMAC: MAC dựa trên SHA256*
- *Timing Attack cho MAC*

Hàm nén từ mã khối

Xây dựng. Xét hệ mã khối $E : K \times \{0,1\}^n \longrightarrow \{0,1\}^n$. Hàm nén **Davies-Meyer** xây dựng bởi:

$$h(H, m) = E(m, H) \oplus H$$



Định lý. Giả sử E là một hệ mã lý tưởng (tập gồm $|K|$ hoán vị ngẫu nhiên).

Tìm một xung đột $h(H, m) = h(H', m')$ mất $O(2^{n/2})$ lần tính (E,D).

Hãy chọn đáp án đúng

- Giả sử ta định nghĩa

$$h(H, m) = E(m, H)$$

- Vậy thì hàm $h(.,.)$ không kháng xung đột:
  - để tìm xung đột (H,m) và (H',m') ta chọn ngẫu nhiên (H,m,m')
  - và xây dựng H' như sau:

1. $H'=D(m', E(m,H))$
2. $H'=E(m', D(m,H))$
3. $H'=E(m', E(m, H))$
4. $H'=D(m', D(m,H))$

Các cách xây dựng khác

- Để đơn giản, ta xét $E: \{0,1\}^n \times \{0,1\}^n \longrightarrow \{0,1\}^n$

- Miyaguchi-Preneel: $h(H, m) = E(m, H) \oplus H \oplus m$ **(Whirlpool)**

  $h(H, m) = E(H \oplus m, m) \oplus m$

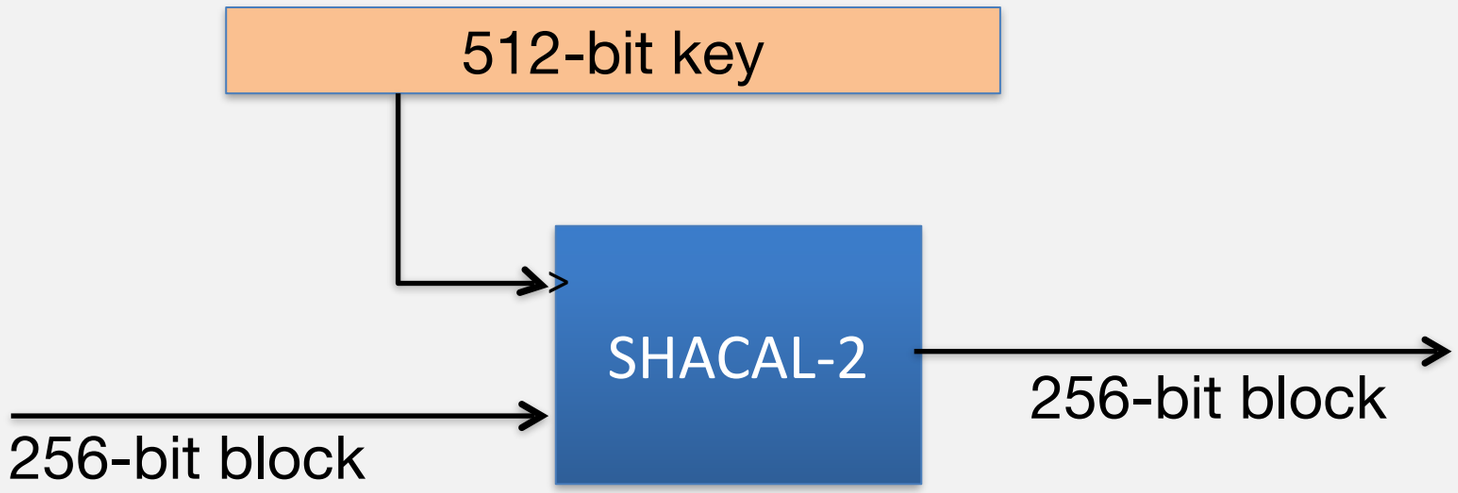
  có 12 biến thể như vậy

- Các biến thể khác là không an toàn, ví dụ

  $h(H, m) = E(m, H) \oplus m$ (Bài tập)

Case study: SHA-256

- Merkle-Damgard function
- Davies-Meyer compression function
- Block cipher: SHACAL-2



Hàm nén có thể chứng minh an toàn

- Chọn một số nguyên tố ngẫu nhiên p kích thước 2000-bit và các số ngẫu nhiên $1 \le u, v \le p$.
- Với mỗi $m, h \in \{0,\ldots,p\text{-}1\}$ ta định nghĩa

$$h(H, m) = u^H \cdot v^m \pmod p$$

Sự kiện. Tìm xung đột cho $h(.,.)$ là khó như giải bài toán "discrete-log" modun p.

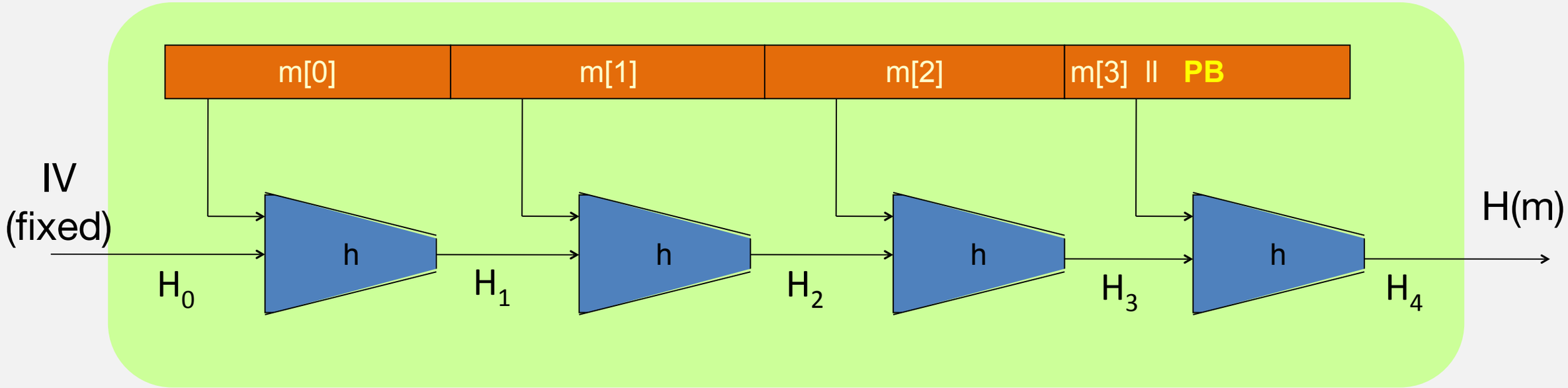
Vấn đề: hàm nén này chậm.

HÀM BĂM KHÁNG XUNG ĐỘT

https://class.coursera.org/crypto-preview/class/index

- *Giới thiệu*
- *Tấn công dùng nghịch lý ngày sinh*
- *Sơ đồ Merkle-Damgard*
- *Xây dựng hàm nén*
- *HMAC: MAC dựa trên SHA256*
- *Timing Attack cho MAC*

Nhắc lại: Xây dựng theo sơ đồ Merkle-Damgard



Định lý. *h* kháng xung đột ⇒ *H* kháng xung đột.

Câu hỏi:

- Liệu chúng ta có thể sử dụng H(.) trực tiếp để xây dựng MAC ?

MAC từ hàm băm theo sơ đồ Merkle-Damgard

Xây dựng thử nghiệm:

- Xét $H: X^{\leq L} \longrightarrow T$ là một hàm băm kháng xung đột theo sơ đồ Merkle-Damgard
- Ta xây dựng

$$S(k, m) = H(k \,\|\, m)$$

- MAC này là không an toàn bởi vì:

1. Cho $H(k \,\|\, m)$ có thể tính $H(w \,\|\, k \,\|\, m \,\|\, PB)$ với mọi w.
2. Cho $H(k \,\|\, m)$ có thể tính $H(k \,\|\, m \,\|\, w)$ với mọi w.
3. Cho $H(k \,\|\, m)$ có thể tính $H(k \,\|\, m \,\|\, \text{PB} \,\|\, w)$ với mọi w.
4. Mọi người đều có thể tính $H(k \,\|\, m)$ với mọi m.

Phương pháp chuẩn: HMAC (Hash-MAC)

Được dùng rộng rãi trên Internet

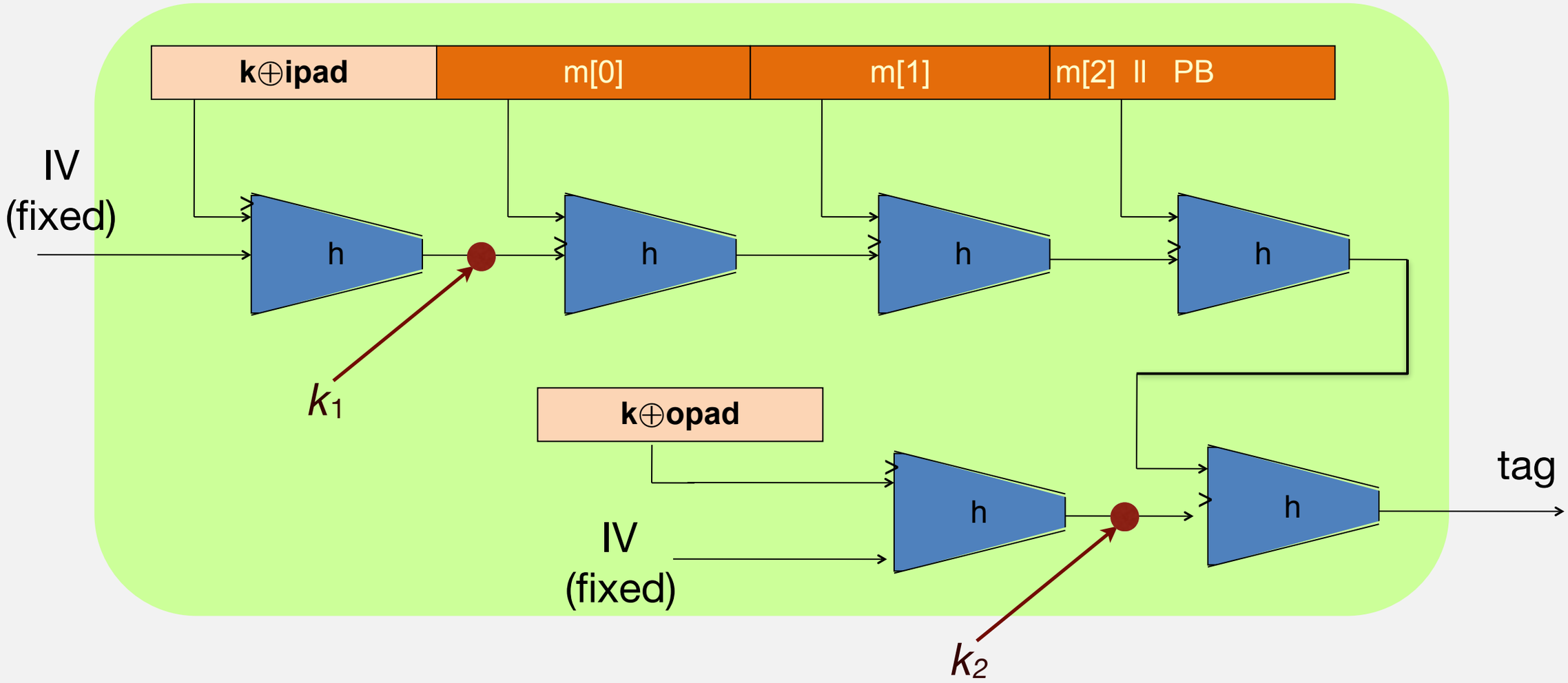
Hàm băm *H*

- Ví dụ: SHA-256 ; output là 256 bits

Ta xây dựng MAC từ hàm băm:

HMAC: $S(k, m) = H(\ k \oplus \text{opad} \ \|\ H(\ k \oplus \text{ipad} \| m\ )\ )$

Hình mô tả HMAC



Tương tự như NMAC PRF

- khác biệt chính: hai khóa k_1 và k_2 phụ thuộc nhau.

Tính chất của HMAC

- Xây dựng từ cài đặt của SHA-256

- HMAC được giả sử là một PRF an toàn
  - Có thể chứng minh với một số giả sử PRF về h(.,.)
  - Chặn về an toàn tương tự như NMAC:

    Khi $q^2/|T|$ là "không đáng kể".

- Trong TLS: có hô trợ HMAC-SHA1-96

HÀM BĂM KHÁNG XUNG ĐỘT

https://class.coursera.org/crypto-preview/class/index

- *Giới thiệu*
- *Tấn công dùng nghịch lý ngày sinh*
- *Sơ đồ Merkle-Damgard*
- *Xây dựng hàm nén*
- *HMAC: MAC dựa trên SHA256*
- *Timing Attack cho MAC*

Chú ý: Timing Attacks vào hàm kiểm tra

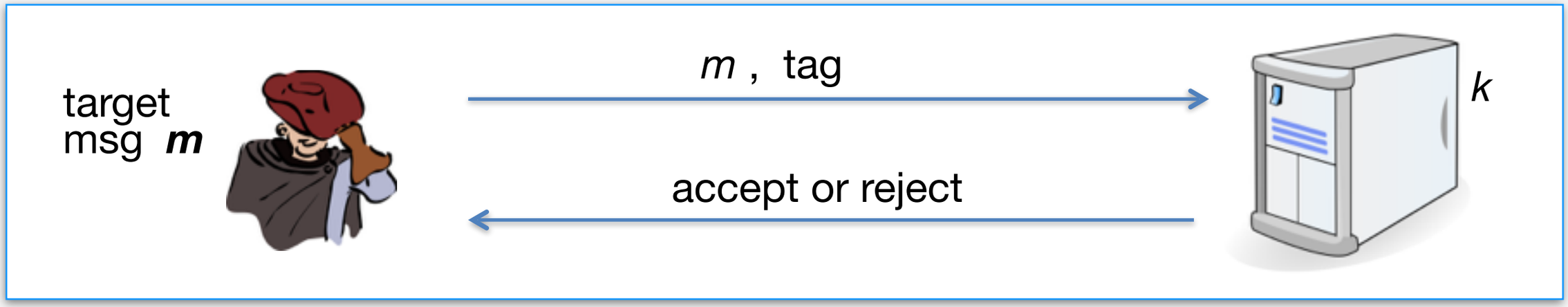
Ví dụ. Keyczar crypto library (Python) [Đã đơn giản hóa]

```
def Verify(key, msg, sig_bytes):
      return HMAC(key, msg) == sig_bytes
```

Vấn đề:

- Cài đặt của phép toán '==' so sánh tuần tự từng byte
- Sẽ trả lại false ngay khi gặp byte khác nhau đầu tiên.

Chú ý: Timing Attacks vào hàm kiểm tra



Timing attack:

- Để tính tag cho một thông điệp *m* ta thực hiện:

  1. Truy vấn server để lấy một tag ngẫu nhiên
  2. Lặp lại mọi khả năng của byte đầu tiên và gửi đến server, dừng khi hàm verification chạy nhanh hơn so với thời gian thực hiện bước 1
  3. Lặp lại với mọi byte trong tag cho đến khi tìm được tag.

Chống Timing Attack #1

Phương pháp

- Đảm bảo rằng phép toán so sánh sẽ luôn thực hiện với thời gian bằng nhau trên mọi dữ liệu

```
return false if  sig_bytes  has wrong length
result = 0
for x, y in zip( HMAC(key,msg) , sig_bytes):
          result |= ord(x) ^ ord(y)
return result == 0
```

Vấn đề:

- Không đảm bảo được việc trình biên dịch không sửa lại đoạn mã trong khi tối ưu

Chống Timing Attack #2

Phương pháp

- Đảm bảo rằng phép toán so sánh sẽ luôn thực hiện với thời gian bằng nhau trên mọi dữ liệu

```
def Verify(key, msg, sig_bytes):
        mac = HMAC(key, msg)
        return HMAC(key, mac) == HMAC(key, sig_bytes)
```

Đảm bảo

- Kẻ tấn công không biết giá trị nào đang được so sánh.

Đừng tự cài đặt crypto !