

# Mã Hóa Ứng dụng

Đoàn Trình Dục/ Đoàn Trình Dục

Mail: duc.duantrinh@stu.edu.vn

2019



# NỘI DUNG CHÍNH



Giới Thiệu



Các Khái Niệm Cơ bản



Mã Hóa Đối Xứng



Mã Hóa Bất Đối Xứng



Hash Function



Thám Mã



Độ An Toàn

# GIỚI THIỆU

# Giới Thiệu

- **Mã hóa** là phương pháp để biến thông tin (file, hình ảnh...) từ định dạng bình thường sang dạng thông tin không thể hiểu được nếu không có phương tiện **giải mã**.
- Giải mã là phương pháp để đưa từ dạng thông tin đã được mã hóa về dạng thông tin ban đầu, quá trình ngược của mã hóa.

# CÁC KHÁI NIỆM CƠ BẢN

## Các Khái Niệm Cơ Bản

**Một hệ thống mã hóa bao gồm các thành phần:**

- Thông tin trước khi mã hóa, ký hiệu là  $P$
- Thông tin sau khi mã hóa, ký hiệu là  $C$
- Chìa khóa, ký hiệu là  $K$
- Phương pháp mã hóa/giải mã, ký hiệu là  $E/D$ .
- Quá trình mã hóa được tiến hành bằng cách áp dụng hàm toán học  $E$  lên thông tin  $P$ , vốn được biểu diễn dưới dạng số, để trở thành thông tin đã mã hóa  $C$ .
- Quá trình giải mã được tiến hành ngược lại: áp dụng hàm  $D$  lên thông tin  $C$  để được thông tin đã giải mã  $P$ .

# Các Khái Niệm Cơ Bản

- **Mật mã học** là chuyên ngành khoa học của Khoa học máy tính nghiên cứu về các nguyên lý và phương pháp mã hóa. Hiện nay người ta đưa ra nhiều chuẩn an toàn cho các lĩnh vực khác nhau của công nghệ thông tin.
- **Thám mã** nghiên cứu các nguyên lý và phương pháp giải mã thường là không biết khóa. Thông thường khi đưa các mã mạnh ra làm chuẩn phổ biến công khai các mã đó được các kẻ thám mã cũng như những người phát triển mã tìm hiểu nghiên cứu.
- **Lý thuyết mã** bao gồm cả mật mã và thám mã để đánh giá một mã mạnh hay không.

# MÃ HÓA ĐỔI XỨNG/KHÓA BÍ MẬT

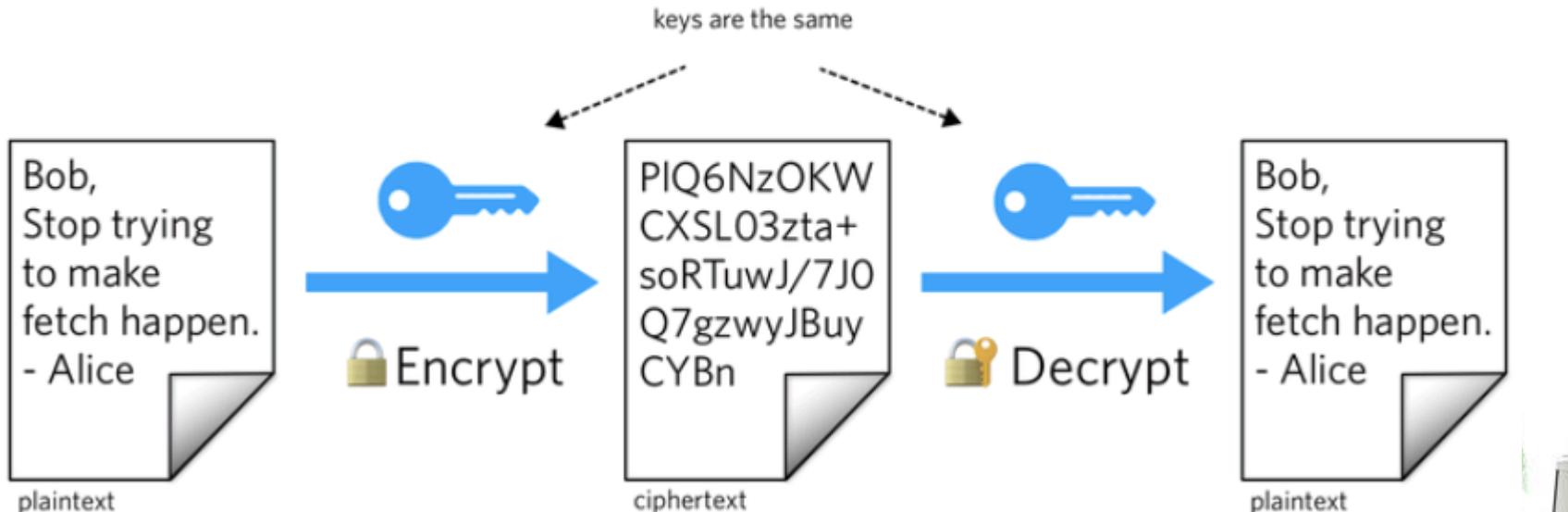
# Mã Hóa Đổi Xứng

- ❖ Mã hóa đổi xứng là loại mã hóa sử dụng một khóa K cho việc mã hóa và giải mã
- ❖ Với X là văn bản cần được mã hóa (plaintext)
- ❖ Với K là khóa bí mật
- ❖ Với E là thuật toán mã hóa
- ❖ Với D là thuật toán giải mã
- ❖ Với Y là bản mã hóa của X thì
- ❖  $Y = E_k(X)$  và  $X = D_k(Y)$

# Mã Hóa Đổi Xứng

## ❖ Mô hình mã hóa đổi xứng

### Symmetric Cryptography



# Mã Hóa Đổi Xứng

- ❖ Một mã đổi xứng có các đặc trưng là cách xử lý thông tin của thuật toán mã hóa, giải mã, tác động của khóa vào bản mã, độ dài của khóa.
- ❖ Mỗi liên hệ giữa bản rõ, khóa và bản mã thông qua thuật toán càng phức tạp càng tốt.
- ❖ Thuật toán mã hóa mạnh: Có cơ sở toán học vững chắc đảm bảo rằng dù công khai thuật toán, nhưng việc thám mã là rất khó khăn và phức tạp nếu không biết khóa.
- ❖ Khoá được giữ bí mật: Chỉ có người gửi và người nhận biết. Có kênh an toàn để phân phối khoá giữa các người sử dụng chia sẻ khóa. Mỗi liên hệ giữa khóa và bản mã là không nhận biết được.

## Mã Hóa Đổi Xứng

Trong mã hóa đổi xứng được chia làm 2 loại thuật toán:

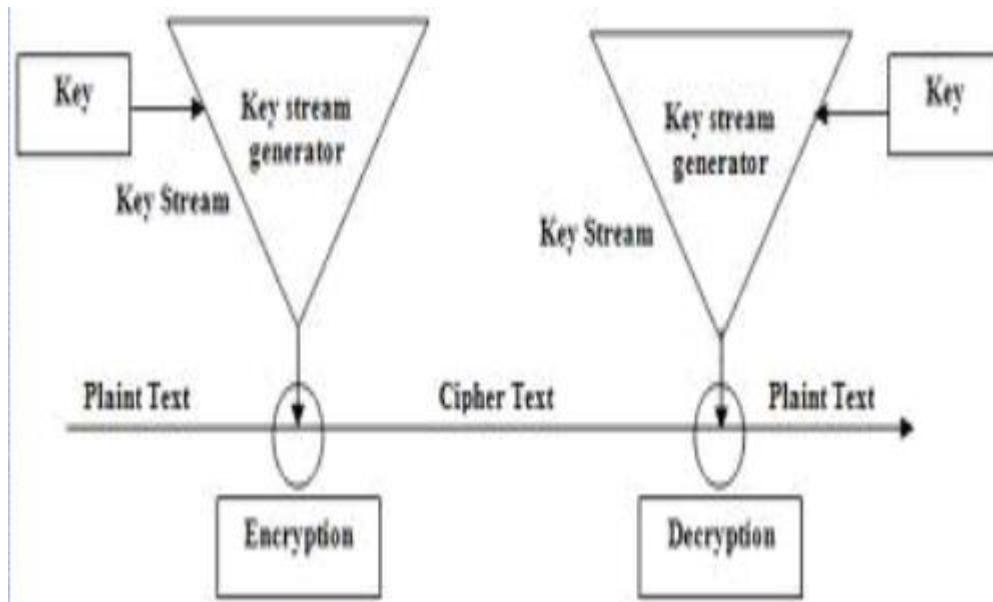
- ❖ Thuật toán block algorithms
- ❖ Thuật toán stream algorithms

# Mã Hóa Đổi Xứng

- ❖ **Stream cipher** là phương pháp mã hóa cho văn bản trong đó thuật toán mã hóa được áp dụng cho mỗi Bit dữ liệu từng bit một
- ❖ Sử dụng một khóa key tạo ngẫu nhiên một Key stream
- ❖  $C_1, C_2, C_3, C_4 \dots \dots$  Với  $C_i = P_i \text{ XOR } K_i$
- ❖ Stream cipher key phải có chu kỳ tạo key lớn
- ❖ Stream cipher key phải có giá trị ngẫu nhiên xuất hiện tương đương nhau
- ❖ Stream cipher key không được sử dụng lại stream key

# Mã Hóa Đổi Xứng

- ❖ Hệ thống Stream cipher

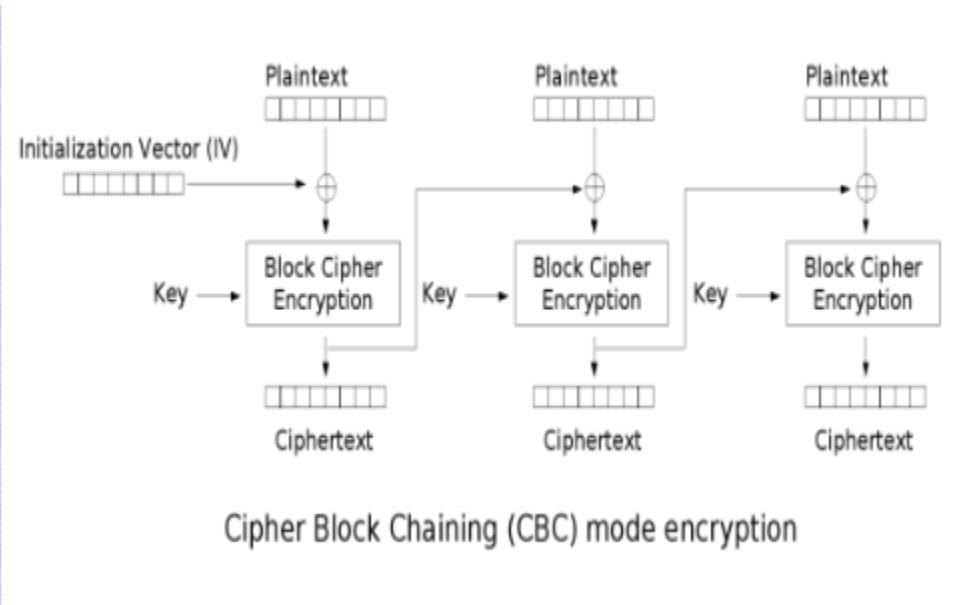


# Mã Hóa Đổi Xứng

- ❖ **Block cipher** là phương pháp mã hóa cho văn bản trong đó thuật toán mã hóa được áp dụng cho mỗi block dữ liệu
- ❖ Mỗi đoạn văn bản được chia thành các block và mỗi block có n bit dữ liệu
- ❖ Nếu số bit ít hơn n thì sẽ được thêm một số bit cho đủ n bit được gọi là kĩ thuật padding
- ❖ Nếu số bit văn bản không là bội số của n thì block cuối cùng sẽ được thêm vào một số bit để đủ n bit
- ❖ Các thuật toán sẽ có số bit trong block cố định ví dụ: DES là 64 bit

# Mã Hóa Đối Xứng

## ❖ Hệ thống block cipher



# MÃ HÓA BẤT ĐỐI XỨNG/KHÓA CÔNG KHAI

# Mã Hóa Bất Đối Xứng

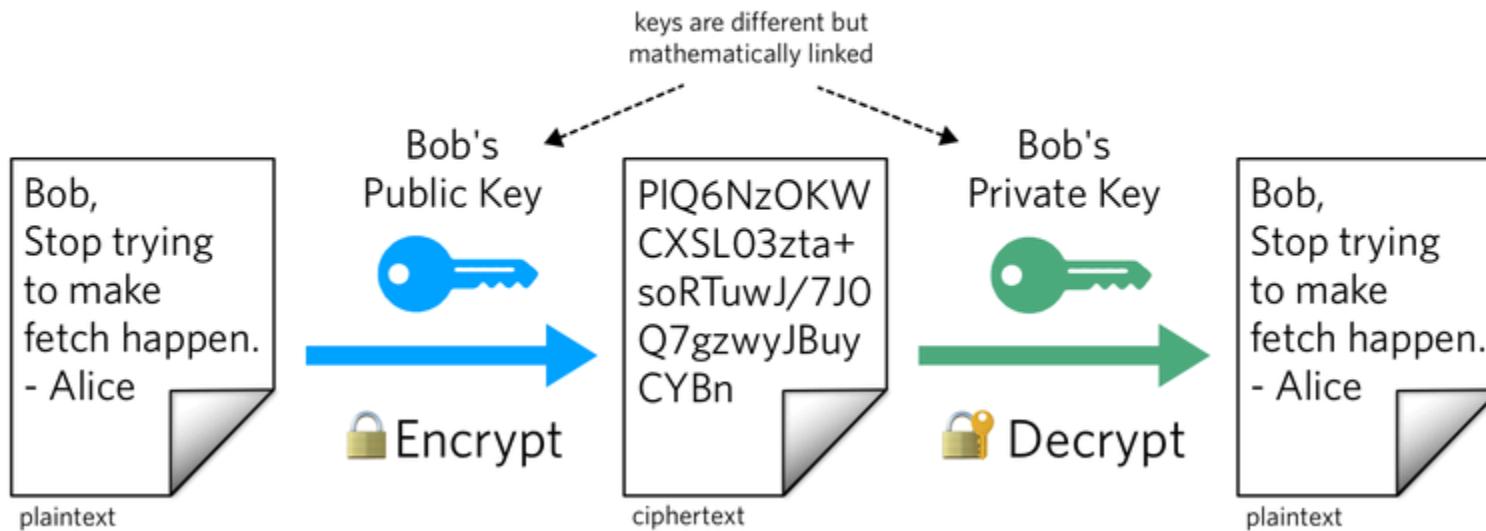
- ❖ Mật mã hóa khóa công khai là một dạng mật mã hóa cho phép người sử dụng trao đổi các thông tin mật mà không cần phải trao đổi các khóa chung bí mật trước đó.
- ❖ Điều này được thực hiện bằng cách sử dụng một cặp khóa có quan hệ toán học với nhau là khóa công khai và khóa cá nhân.
- ❖ Thuật ngữ **mật mã hóa khóa bất đối xứng** thường được dùng đồng nghĩa với **mật mã hóa khóa công khai** mặc dù hai khái niệm không hoàn toàn tương đương. Có những thuật toán mật mã hóa bất đối xứng không có tính chất khóa công khai và bí mật như đề cập ở trên mà cả hai khóa (cho mã hóa và giải mã) đều cần phải giữ bí mật

## Mã Hóa Bất Đối Xứng

- ❖ Mã hóa đối xứng là loại mã hóa sử dụng một khóa K public cho việc mã hóa và khóa K private để giải mã
- ❖ Với X là văn bản cần được mã hóa (plaintext)
- ❖ Với  $K_p$  là khóa share và  $K_s$  là khóa bí mật
- ❖ Với E là thuật toán mã hóa
- ❖ Với Y là bản mã hóa của X thì
- ❖  $Y = E_{k_p}(X)$  và  $X = E_{k_s}(E_{k_p}(X))$

# Mã Hóa Bất Đôi Xứng

## Public Key Cryptography



# Mã Hóa Bất Đối Xứng

Hệ thống mật mã hóa khóa công khai có thể sử dụng với các mục đích:

- **Mã hóa:** giữ bí mật thông tin và chỉ có người có khóa bí mật mới giải mã được.
- **Tạo chữ ký số:** cho phép kiểm tra một văn bản có phải đã được tạo với một khóa bí mật nào đó hay không.
- **Thỏa thuận khóa:** cho phép thiết lập khóa dùng để trao đổi thông tin mật giữa 2 bên.
- Thông thường, các kỹ thuật mật mã hóa khóa công khai đòi hỏi khối lượng tính toán nhiều hơn các kỹ thuật mã hóa khóa đối xứng nhưng những lợi điểm mà chúng mang lại khiến cho chúng được áp dụng trong nhiều ứng dụng.

# HASH FUNCTION

Hash function

# HASH Function

- ❖ Hash function là giải thuật nhằm sinh ra các **giá trị băm** tương ứng với mỗi **khối dữ liệu** (có thể là một chuỗi ký tự, một đối tượng trong lập trình hướng đối tượng, v.v....).
- ❖ **Giá trị băm** đóng vai gần như một **khóa** để phân biệt các **khối dữ liệu**, tuy nhiên, người ta chấp **hiện tượng trùng khóa** hay còn gọi là **đụng độ** và cố gắng cải thiện giải thuật để giảm thiểu sự đụng độ đó.
- ❖ Hàm băm thường được dùng trong nhằm giảm **chi phí tính toán** khi tìm một **khối dữ liệu** trong một tập hợp (nhờ việc so sánh các **giá trị băm** nhanh hơn việc so sánh những *khối dữ liệu* có kích thước lớn).
- ❖ Hàm băm mật mã hiện nay được sử dụng rộng rãi([https](https://), block chain)

# HASH Function

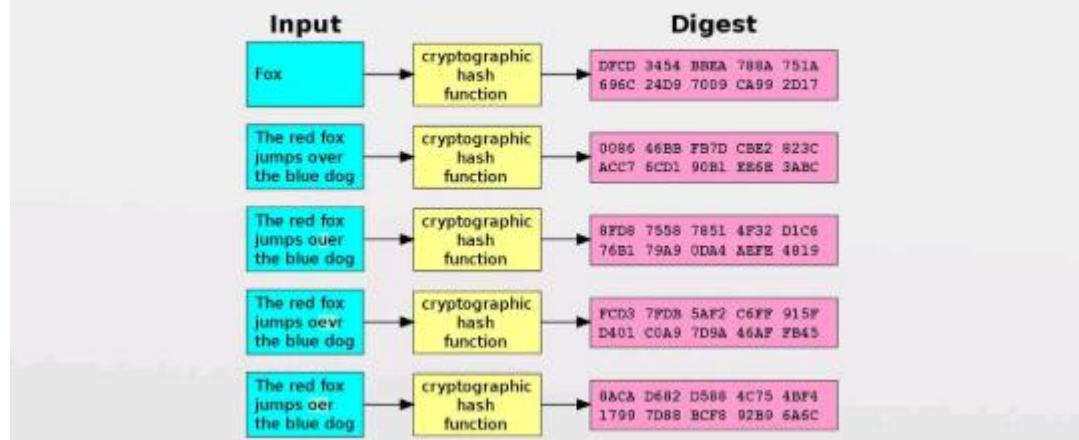
## Một số tính chất của Hash function

- ❖ Tính toán nhanh.
- ❖ Các khoá được phân bố đều.
- ❖ Ít xảy ra đụng độ.
- ❖ Xử lý được các loại kiểu dữ liệu khác nhau.
- ❖ Hash function là một hàm một chiều
- ❖ Một số hàm băm được sử dụng hiện nay như:  
MD5, SHA-224, SHA-256, SHA-384, SHA-512, ...

# HASH Function

Minh họa về hàm Băm

## Cryptographic hash function



# THÁM MÃ

# Thám Mã

**Có 2 cách tấn công chính về mã hóa**

- ❖ **Tấn công dùng thuật toán:** dựa trên thuật toán và một số đặc trưng chung về bản rõ hoặc một số mẫu bản rõ/bản mã. Kiểu tấn công này nhằm khai phá các đặc trưng của thuật toán để tìm bản rõ cụ thể hoặc tìm khóa.
- ❖ **Tấn công duyệt toàn bộ:** kẻ tấn công tìm cách thử mọi khóa có thể trên bản mã cho đến khi nhận được bản rõ. Trung bình cần phải thử một nửa số khóa.

# Thám Mã

- ❖ Biết thuật toán và bản mã, dùng phương pháp thống kê, xác định bản rõ.
- ❖ Biết thuật toán, biết được bản mã/bản rõ tấn công tìm khóa.
- ❖ Chọn bản rõ và nhận được bản mã, biết thuật toán tấn công tìm khóa.
- ❖ Chọn bản mã và có được bản rõ tương ứng, biết thuật toán tấn công tìm khóa...

# Thám Mã

- ❖ Về mặt lý thuyết phương pháp duyệt tổng thể là luôn thực hiện được, do có thể tiến hành thử từng khoá, mà số khoá là hữu hạn.
- ❖ Phản ứng lớn công sức của các tấn công đều tỷ lệ thuận với kích thước khoá. Khóa càng dài thời gian tìm kiếm càng lâu và thường tăng theo hàm mũ.
- ❖ Ta có thể giả thiết là kẻ thám mã có thể dựa vào đặc trưng về ngữ cảnh để nhận biết được bản rõ.

# Thám Mã

Key Size (bits)	Number of Alternative Keys	Time required at 1 encryption/ $\mu$ s	Time required at $10^6$ encryptions/ $\mu$ s
32	$2^{32} = 4.3 \times 10^9$	$2^{31} \mu$ s = 35.8 minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55} \mu$ s = 1142 years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127} \mu$ s = $5.4 \times 10^{24}$ years	$5.4 \times 10^{18}$ years
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167} \mu$ s = $5.9 \times 10^{36}$ years	$5.9 \times 10^{30}$ years
26 characters (permutation)	$26! = 4 \times 10^{26}$	$2 \times 10^{26} \mu$ s = $6.4 \times 10^{12}$ years	$6.4 \times 10^6$ years

# ĐỘ AN TOÀN



# Độ An toàn

- ❖ **An toàn không điều kiện:** ở đây không quan trọng máy tính mạnh như thế nào, có thể thực hiện được bao nhiêu phép toán trong một giây, bản mã không thể bị bẻ, vì bản mã không cung cấp đủ thông tin để xác định duy nhất bản rõ.
- ❖ Chưa có thuật toán mã hóa nào được coi là an toàn không điều kiện.

# Độ An toàn

❖ **An toàn tính toán:** với nguồn lực máy tính giới hạn và thời gian có hạn (chẳng hạn thời gian tính toán không quá tuổi của vũ trụ) mã hóa coi như không thể bị bẻ. Trong trường hợp này coi như mã hóa an toàn về mặt tính toán. Nói chung từ nay về sau, một thuật toán mã hóa an toàn tính toán được coi là an toàn.

# Mã Hóa Ứng dụng-Chương 2



Đoàn Trình Dục/ Đoàn Trình Dục

Mail: duc.dolantrinh@stu.edu.vn

2019



# NỘI DUNG CHÍNH



Giới Thiệu Mã Hóa Cổ Điển



Mã Hóa Đối Xứng



Các Hệ Thống Mã Cổ Điển



Phân Tích Các Hệ Mã Cổ Điển



Thám Mã



Xây Dựng Hệ Mã Cổ Điển Trên Python

# **GIỚI THIỆU VỀ MÃ HÓA CỔ ĐIỂN**

# Giới Thiệu Mã Hóa Cỗ Điển

- ❖ Mật mã học cổ điển là một dạng của mật mã học đã được sử dụng trong lịch sử phát triển của loài người nhưng ngày nay đã trở nên lạc hậu do các phương thức mã hóa này quá đơn giản và những kẻ tấn công có thể dễ dàng bẻ khóa thông qua nhiều phương thức như tấn công vét cạn, hay thống kê,...
- ❖ Ví dụ: như dùng máy tính thử hết mọi trường hợp hay dựa trên tấn công thống kê (dựa trên tần suất xuất hiện của các chữ cái).
- ❖ Nói chung, mật mã học cổ điển hoạt động trên cơ sở bảng chữ cái (chẳng hạn các ký tự từ "A" tới "Z" trong tiếng Anh), và chúng được thực hiện bằng tay hay một số máy móc cơ khí đơn giản.
- ❖ Mã hóa cổ điển là mã hóa đối xứng/ mã khóa bí mật

# Giới Thiệu Mã Hóa Cỗ Điện

- Mã hóa cỗ điện có 2 phương pháp chính để thực hiện:
- Phương pháp thế (Substitution cipher)
- Phương pháp hoán vị(permuation cipher)
- Mã hóa cỗ điện là những phương pháp cơ bản để nghiên cứu và phát triển mã hóa đối xứng trong hiện đại

# CEASAR CIPHER

## Mã Hóa Ceasar

- ❖ Đây là mã thế được biết sớm nhất, được sáng tạo bởi Julius Ceasar. Lần đầu tiên được sử dụng trong quân sự. Việc mã hóa được thực hiện đơn giản là thay mỗi chữ trong bản rõ bằng chữ thứ ba tiếp theo trong bảng chữ cái.
- ❖ Ví dụ:
  - Meet me after the toga party
  - PHHW PH DIWHU WKH WRJD SDUWB
- ❖ Công thức của mã hóa Ceasar:
$$C = E(p) = (p + k) \bmod 26$$
$$P = D(C) = (c - k) \bmod 26$$
- ❖ Không gian khóa nhỏ 26 khóa

# BẢNG MÃ CHỮ ĐƠN



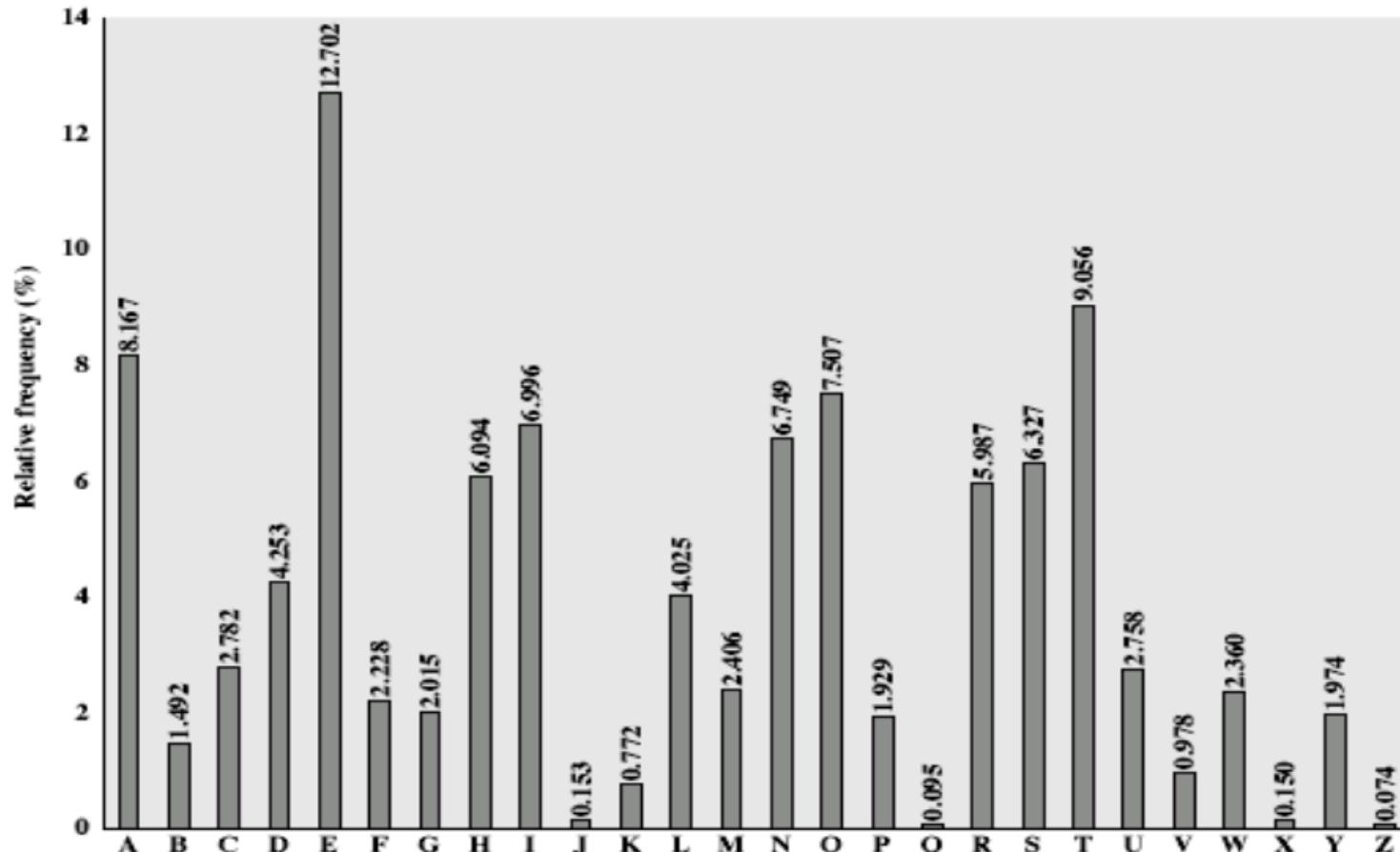
## Bảng Mã Chữ Đơn

- ❖ Trong một mã mỗi chữ của bản rõ được ánh xạ đến một chữ khác nhau của bản mã
- ❖ Như vậy độ dài khoá ở đây là 26 và số khoá có thể có là 26!.
- ❖ Ví dụ:
  - Plaintext : abcdefghijklmnopqrstuvwxyz
  - Ciphertext: DKVQFIBJWPESCXHTMYAUOLRGZN
  - Plaintext: If we wish to replace letters
  - Ciphertext :WIRFRWAJUHYFTSDVFSUUUFYA

## Bảng Mã Chữ Đơn

- ❖ Dựa vào các đặc trưng về tần suất xuất hiện của các chữ cái trong tiếng Anh chữ E được sử dụng nhiều nhất, sau đó đến các chữ T, R, N, I, O, A, S. Một số chữ rất ít dùng như: Z, J, K, Q, X; các bộ chữ thường dùng "th lrd s m shphrd shll nt wnt".
- ❖ Tần suất tương đối của các chữ ở plaintext và cipher text ko đổi. Điều đó được phát hiện bởi các nhà khoa học Ai cập từ thế kỷ thứ 9.

# Bảng Mã Chữ Đơn



## Bảng Mã Chữ Đơn

- Thám mã trên bảng chữ đơn: Tính toán tần suất của các chữ trong bản mã So sánh với các giá trị đã biết Tìm kiếm các chữ đơn hay dùng (A,I,E...), bộ đôi (NO) và bộ ba (RST); và các bộ ít dùng J,K, X,Z. Trên bảng chữ đơn cần xác định các chữ dùng các bảng bộ đôi và bộ ba trợ giúp

# Bảng Mã Chữ Đơn

- ❖ Ví dụ: UZQSOVUOHHXMOPVGPOZPEVSGZWSZOPFPE  
SXUDBMETSXAIZVUEPHZHMDZSHZOWSFPAP  
PDTSPQUZWYMXUZUHSXEPLYEP  
OPDZSZUFPOUDTMOHMQ
- ❖ Tính tần suất các chữ
- ❖ Đoán P và Z là e và t.
- ❖ Khi đó ZW là th và ZWP là the.
- ❖ Suy luận tiếp tục ta có bản rõ: it was disclosed yesterday  
that several informal but direct contacts have been made  
with political representatives in moscow

# MÃ PLAYFAIR



# Mã PlayFair

- ❖ Sáng tạo bởi Charles Wheatstone vào năm 1854 và mang tên người bạn là Baron Playfair
- ❖ Ma trận khoá Playfair: Cho trước một từ làm khoá, với điều kiện trong từ khoá đó không có chữ cái nào bị lặp. Ta lập ma trận Playfair là ma trận cỡ  $5 \times 5$  dựa trên từ khoá đã cho và gồm các chữ trên bảng chữ cái, được sắp xếp theo thứ tự như sau: Trước hết viết các chữ của từ khoá vào các hàng của ma trận bắt từ hàng thứ nhất.  
❖ Nếu ma trận còn trống, viết các chữ khác trên bảng chữ cái chưa được sử dụng vào các ô còn lại. Có thể viết theo một trình tự qui ước trước, chẳng hạn từ đầu bảng chữ cái cho đến cuối.

# Mã PlayFair

- ❖ Sáng tạo bởi Charles Wheatstone vào năm 1854 và mang tên người bạn là Baron Playfair
- ❖ Ma trận khoá Playfair: Cho trước một từ làm khoá, với điều kiện trong từ khoá đó không có chữ cái nào bị lặp. Ta lập ma trận Playfair là ma trận cỡ  $5 \times 5$  dựa trên từ khoá đã cho và gồm các chữ trên bảng chữ cái, được sắp xếp theo thứ tự như sau:
  - ❑ Trước hết viết các chữ của từ khoá vào các hàng của ma trận bắt từ hàng thứ nhất.
  - ❑ Nếu ma trận còn trống, viết các chữ khác trên bảng chữ cái chưa được sử dụng vào các ô còn lại. Có thể viết theo một trình tự qui ước trước, chẳng hạn từ đầu bảng chữ cái cho đến cuối.

# Mã PlayFair

- ❖ Ví dụ: ma trận khóa MONARCHY
- ❖ Chữ I,J được viết vào 1 ô

M	O	N	A	R
C	H	Y	B	D
E	F	G	I,J	K
L	P	Q	S	T
U	V	W	X	Z

# Mã PlayFair

- ❖ Chia bản rõ thành từng cặp chữ. Nếu một cặp nào đó có hai chữ như nhau, thì ta chèn thêm một chữ lọc chẵng hạn X. Ví dụ, trước khi mã “balloon” biến đổi thành “balxloxon”.
- ❖ Nếu cả hai chữ trong cặp đều rơi vào cùng một hàng, thì mã mỗi chữ bằng chữ ở phía bên phải nó trong cùng hàng của ma trận khóa, chẵng hạn “ar” biến đổi thành “rm”

M	O	N	A	R
C	H	Y	B	D
E	F	G	I,J	K
L	P	Q	S	T
U	V	W	X	Z

# Mã PlayFair

- ❖ Nếu cả hai chữ trong cặp đều rơi vào cùng một cột, thì mã mỗi chữ bằng chữ ở phía bên dưới nó trong cùng cột của ma trận khóa, chẳng hạn “mu” biến đổi thành “cm”.
- ❖ Trong các trường hợp khác, mỗi chữ trong cặp được mã bởi chữ cùng hàng với nó và cùng cột với chữ cùng cặp với nó trong ma trận khóa. Chẳng hạn, “hs” mã thành “bp”, và “ea” mã thành “im” hoặc “jm” (tùy ý)

M	O	N	A	R
C	H	Y	B	D
E	F	G	I,J	K
L	P	Q	S	T
U	V	W	X	Z

# Mã PlayFair

- ❖ Ta có  $26 \times 26$  cặp chữ, nên mỗi chữ cái có thể được mã hóa bởi 7 chữ cái khác nhau nên tần suất các chữ cái trên mảng mã tiếng anh giống nhau
- ❖ Muốn thống kê tần suất ta phải có  $26 \times 26$  cặp bảng thám mã lớn hơn nhiều so với bảng chữ chỉ 26
- ❖ Nó vẫn có thể bẻ khóa khi cho trước vài trăm chữ

# MÃ VEGENERE



# Mã VEGENERE

- ❖ Một hướng khác làm tăng độ an toàn cho mã trên bảng chữ là sử dụng nhiều bảng chữ để mã. Ta sẽ gọi chúng là các mã thế đa bảng. Mỗi chữ có thể được mã bằng bất kỳ chữ nào trong bản mã tùy thuộc vào ngữ cảnh khi mã hoá. Làm như vậy để trải đều tần suất các chữ xuất hiện trong bản mã. Do đó làm mất bớt cấu trúc của bản rõ được thể hiện trên bản mã và làm cho thám mã đa bảng khó hơn.
- ❖ Ta sử dụng từ khoá để chỉ rõ chọn bảng nào được dùng cho từng chữ trong bản tin. Sử dụng lần lượt các bảng theo từ khoá đó và lặp lại từ đầu sau khi kết thúc từ khoá. Độ dài khoá là chu kỳ lặp của các bảng chữ. Độ dài càng lớn và nhiều chữ khác nhau được sử dụng trong từ khoá thì càng khó thám mã.

# Mã VEGENERE

- ❖ Mã thế đa bảng đơn giản nhất là mã Vigenere.
- ❖ Thực chất quá trình mã hóa Vigenere là việc tiến hành đồng thời dùng nhiều mã Ceasar cùng một lúc trên bản rõ với nhiều khoá khác nhau.
- ❖ Giả sử khoá là một chữ có độ dài d được viết dạng  $K = K_1 K_2 \dots K_d$ , trong đó  $K_i$  nhận giá trị nguyên từ 0 đến 25. Tần suất các chữ trong bản mã dẫn tương đối đều.

# Mã VEGENERE

Các bước để tiến hành mã hóa Vegenere

- ❖ Viết bản rõ ra
- ❖ Viết từ khoá lặp nhiều lần phía trên tương ứng của nó
- ❖ Sử dụng mỗi chữ của từ khoá như khoá của mã Ceasar Mã chữ tương ứng của bản rõ với bước nhảy tương ứng.
- ❖ Ví dụ: từ khoá deceptive key:  
deceptivedeceptivedeceptive
- ❖ plaintext: wearediscoveredsaveyourself

# Mã VEGENERE

- ❖ Để mã chữ w đầu tiên ta tìm chữ đầu của khóa là d, như vậy w sẽ được mã trên bảng chữ tịnh tiến 3 (tức là a tịnh tiến thành d). Do đó chữ đầu w được mã bởi chữ Z.
- ❖ Chữ thứ hai trong từ khóa là e, có nghĩa là chữ thứ hai trong bản rõ sẽ được tịnh tiến 4 (từ a tịnh tiến đến e). Như vậy thứ hai trong bản rõ e sẽ được mã bởi chữ i.
- ❖ Tương tự như vậy cho đến hết bản rõ.

# Mã VEGENERE

- ❖ Như vậy có chữ mã khác nhau cho cùng một chữ của bản rõ. Suy ra tần suất của các chữ bị là phẳng, nghĩa là tần suất xuất hiện các chữ trên bản mã tương đối đều nhau .
- ❖ Tuy nhiên chưa mât hoàn toàn, do độ dài của khoá có hạn, nên có thể tạo nên chu kỳ vòng lặp. Kẻ thám mã bắt đầu từ tần suất của chữ để xem có phải đây là mã đơn bảng chữ hay không. Giả sử đây là mã đa bảng, sau đó xác định số bảng chữ trong từ khoá và lần tìm từng chữ. Như vậy cần tăng độ dài từ khoá để tăng số bảng chữ dùng khi mã để “là” tần suất của các chữ.

# PHƯƠNG PHÁP KASISKI



# Phương Pháp Kasiski

- ❖ Phương pháp phát triển bởi Babbage và Kasiski.
- ❖ Ta thấy các chữ nhu nhau trên bản rõ và cách nhau một khoảng đúng bằng độ dài từ khoá (chu kỳ), thì sẽ được mã bằng cùng một chữ.
- ❖ Như vậy từ độ lặp của các chữ trong bản mã có thể cho phép xác định chu kỳ. Tất nhiên không phải khi nào cũng tìm được độ dài từ khoá.
- ❖ Sau đó tìm các chữ trong từ khoá bằng cách tấn công từng bảng chữ đơn với cùng kỹ thuật dựa trên các bảng tần suất của các bộ chữ như trước.

# MÃ HÓA VEGENERE VỚI KHÓA TỰ ĐỘNG



# Mã Khóa Vigenere với khóa tự động

- ❖ Lý tưởng nhất là ta có khoá dài như bản tin.
- ❖ Do đó Vigenere đề xuất khoá tự động sinh cho bằng độ dài bản tin như sau: từ khoá được nối tiếp bằng chính bản rõ để tạo thành khoá.
- ❖ Sau đó dùng mã Vigenere để mã bản rõ đã cho.Khi đó biết từ khoá có thể khôi phục được một số chữ ban đầu của bản rõ.
- ❖ Sau đó tiếp tục sử dụng chúng để giải mã cho văn bản còn lại. Sự cải tiến này làm mất khái niệm chu kỳ, gây khó khăn cho việc thám mã, nhưng vẫn còn đặc trưng tần suất để tấn công.
- ❖ Ví dụ: với key:`deceptivewearediscoveredsav`
- ❖ plaintext: `wearediscoveredsaveyourself`
- ❖ ciphertext: `ZICVTWQNGKZEIIGASXSTSLVVWLA`

# Mã Khóa Vigenere với khóa tự động

- ❖ Nếu khoá thực sự ngẫu nhiên được dùng và có độ dài bằng bản rõ thì ta nói đó là bộ đệm một lần. Vì nó chỉ được dùng một lần và ngẫu nhiên, nên mã hoá sẽ an toàn. Mã sẽ không bẻ được vì bản mã không có liên quan thống kê gì với bản rõ, do bộ đệm được sinh ngẫu nhiên.
- ❖ Có thể nói mã bộ đệm một lần là an toàn tuyệt đối, vì với bản rõ bất kỳ và bản mã bất kỳ, luôn tồn tại một khoá để ánh xạ bản rõ đó sang bản mã đã cho.
- ❖ Về mặt lý thuyết, xác suất để mọi mẫu tin (có cùng độ dài với bản rõ) trên bảng chữ là mã của một bản rõ cho trước là như nhau. Khoá chỉ sử dụng một lần, nên các lần mã là độc lập với nhau.
- ❖ Vấn đề khó khăn của mã bộ đệm một lần là việc sinh ngẫu nhiên khoá và phân phối khoá an toàn. Do đó bộ đệm một lần ít được sử dụng và chỉ dùng trong trường hợp đòi hỏi bảo mật rất cao

# MÃ HOÁN VỊ



# Mã Hoán Vị

- ❖ Mã hoán vị: các chữ trong bản rõ không được thay thế bằng các chữ khác mà chỉ thay đổi vị trí, tức là việc mã hoá chỉ dịch chuyển vị trí tương đối giữa các chữ trong bản rõ.
- ❖ Bản mã có cùng phân bố tần suất xuất hiện các chữ như bản gốc nên dễ thám mã

# MÃ RAIL FENCE



# Mã Rail Fence

- ❖ Đây là mã hoán vị đơn giản. Viết các chữ của bản rõ theo đường chéo trên một số dòng. Sau đó đọc các chữ theo theo từng dòng sẽ nhận được bản mã. Số dòng chính là khoá của mã. Vì khi biết số dòng ta sẽ tính được số chữ trên mỗi dòng và lại viết bản mã theo các dòng sau đó lấy bản rõ bằng cách viết lại theo các cột.
- ❖ Ví dụ . “meet me after the toga party”

m e m a t r h t g p r y  
e t e f e t e o a a t  
**mematrhtgpryefeteoaat**

# MÃ DỊCH CHUYỂN DÒNG



# Mã Dịch Chuyển Dòng

- ❖ Mã có sơ đồ phức tạp hơn. Viết các chữ của bản tin theo các dòng với số cột xác định. Sau đó thay đổi thứ tự các cột theo một dãy số khoá cho trước, rồi đọc lại chúng theo các cột để nhận được bản mã. Quá trình giải mã được thực hiện ngược lại.

4	3	1	2	5	6	7
a	t	t	a	c	k	p
o	s	t	p	o	n	e
d	u	n	t	i	l	t
w	o	a	m	x	y	z

# **ĐẶC ĐIỂM CỦA MÃ CỔ ĐIỂN**



# Đặc Điểm Của Mã Cỗ Điển

- ❖ Phương pháp mã hoá cổ điển có thể dễ dàng bị giải mã bằng cách đoán chữ dựa trên phương pháp thống kê tần xuất xuất hiện các chữ cái trên mã và so sánh với bảng thống kê quan sát của bản rõ.
- ❖ Để dùng được mã hoá cổ điển thì bên mã hoá và bên giải mã phải thống nhất với nhau về cơ chế mã hoá cũng như giải mã.

# Mã Hóa Ứng dụng-Chương 3



Đoàn Trình Dục/ Đoàn Trình Dục

Mail: duc.duantrinh@stu.edu.vn

2019



# NỘI DUNG CHÍNH



Giới thiệu mã hóa đối xứng



Stream cipher



Block cipher



DES/AES



MODE



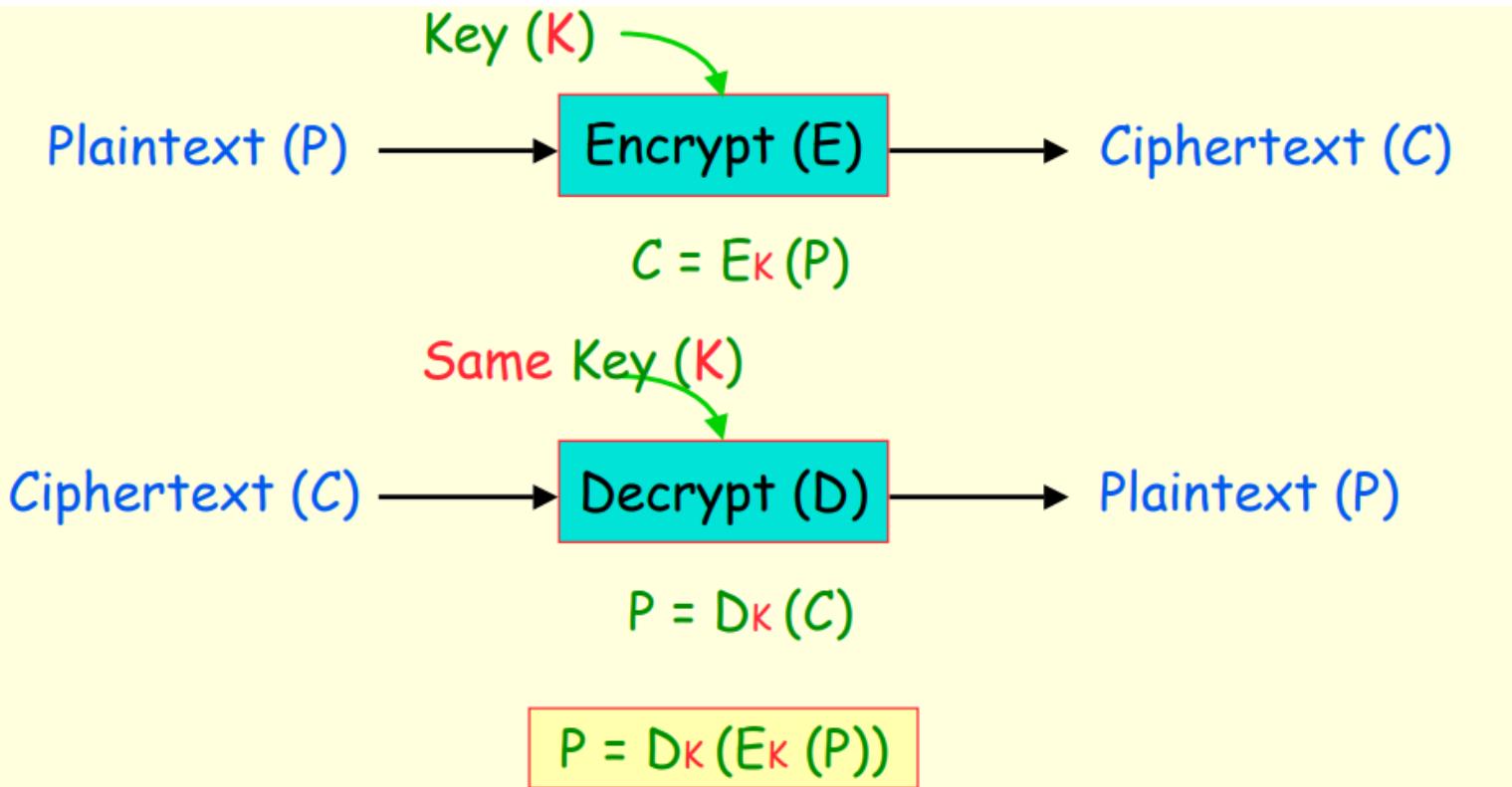
Ưu và nhược điểm

# GIỚI THIỆU

# Giới Thiệu

- ❖ Được biết với các tên gọi khác như Secret Key, Single Key, Private Key
- ❖ Bên gửi và bên nhận phải biết chung một key
- ❖ Yêu cầu giải quyết bài toán chia sẻ key
- ❖ Là thuật toán phổ biến để mã hóa File
- ❖ Các thuật toán mã hóa cổ điển + Mã hóa đối xứng hiện đại như: DES, Triple-DES, AES (Rijndael), IDEA, RC5, RC6, Blowfish,...

# Giới Thiệu

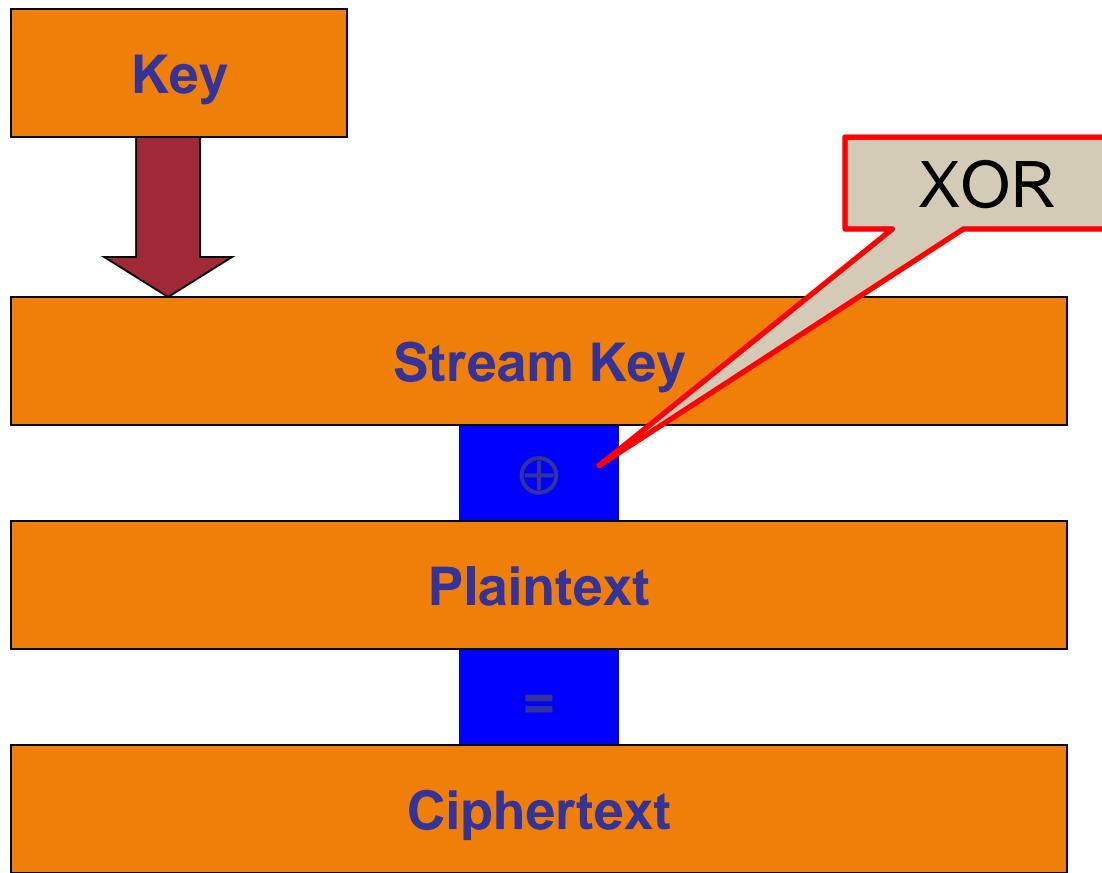


# STREAM CIPHER

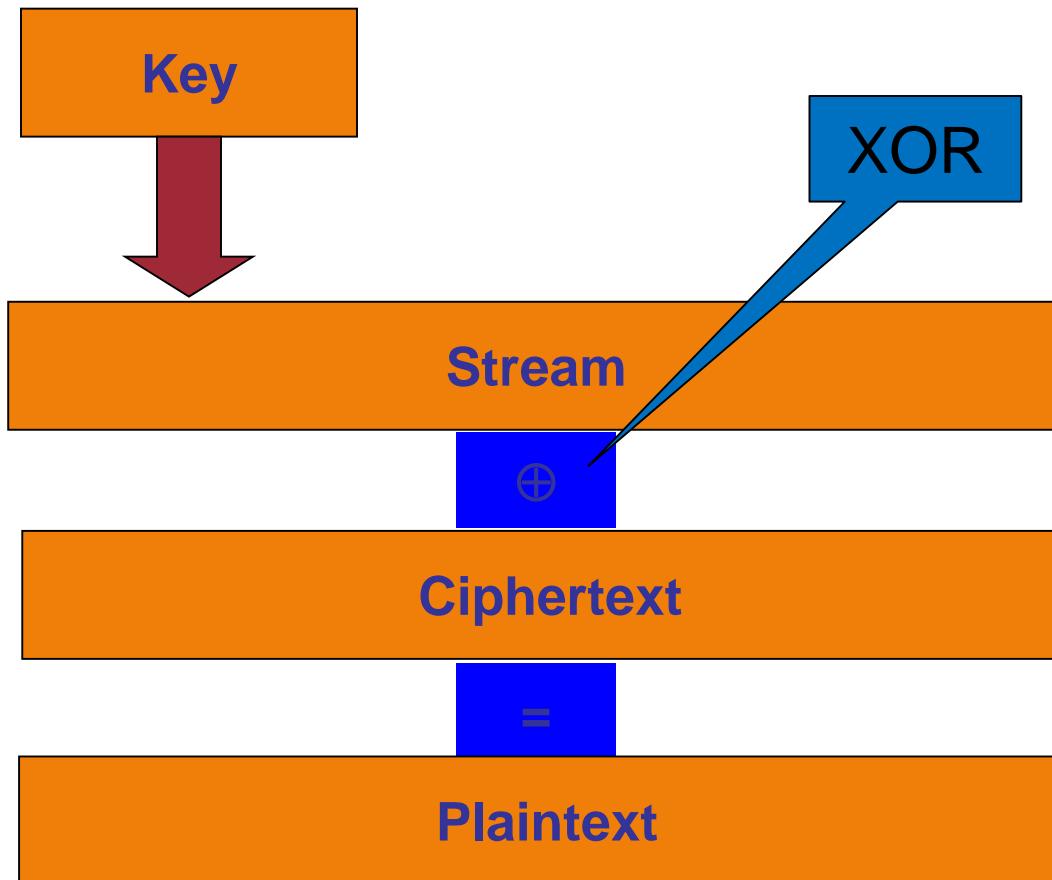
# Stream Cipher

- Stream Cipher là loại mã hóa luồng dữ liệu số một bit hoặc một byte mỗi lần.
- Bắt đầu với một khóa bí mật
- Từ key bí mật này tạo ra một luồng key
- Ví dụ như stream cipher cổ điển là mật mã Vigenère tự động và mật mã Vernam.
- Trình tạo bit-stream phải được xem là một thủ tục thuật toán, vì vậy bit-stream mã hóa có thể được sản xuất bởi cả hai phía người dùng.
- Bộ tạo bit-stream là một thuật toán được điều khiển bằng khóa và phải tạo ra một bit-stream mạnh về mặt mã hoá.

# Stream Cipher Encryption



# Stream Cipher Encryption



# Stream Cipher

- Hầu hết các máy đều sử dụng trước chiến tranh thế giới thứ 2
- Máy German Enigma
- A5 – Mã hóa thiết bị cầm tay GSM truyền thông với trạm gốc
- Thuật toán RC-4 (Ron's Code)

# Stream Cipher A5

- A5/1 được dùng trong mạng điện thoại GSM, để bảo mật dữ liệu trong quá trình liên lạc giữa máy điện thoại và trạm thu phát sóng vô tuyến.
- Đơn vị mã hóa của A5/1 là một bít.
- Bộ sinh số mỗi lần sẽ sinh ra hoặc bít 0 hoặc bít 1 để sử dụng trong phép XOR.
- Mã hóa A5/1 có thể được thực hiện dễ dàng bằng các thiết bị phần cứng, tốc độ nhanh.
- Do đó A5/1 đã từng được sử dụng để mã hóa các dữ liệu real-time như các dãy bít audio.
- Ngày nay A5/1 được sử dụng để mã hóa dữ liệu cuộc gọi trong mạng điện thoại GSM.

# Stream Cipher RC4

- RC4 được dùng trong giao thức SSL để bảo mật dữ liệu trong quá trình truyền dữ liệu giữa Web Server và trình duyệt Web.
- Ngoài ra RC4 còn được sử dụng trong mã hóa WEP của mạng Wireless LAN.
- Đơn vị mã hóa của RC4 là một byte 8 bit.
- Khóa K là một dãy gồm N số nguyên 8 bit với N có thể lấy giá trị từ 1 đến 256.
- Bộ sinh số mỗi lần sinh ra một byte để sử dụng trong phép XOR.

# Stream Cipher OTP(One Time Pad)

Với bản rõ  $m$  và khóa  $k$  có cùng độ dài theo bit, One-Time-Pad được xác định như sau:

$$E(m, k) = m \text{ XOR } k = c$$

$$D(c, k) = c \text{ XOR } k = (m \text{ XOR } k) \text{ XOR } k = m$$

Với OTP, khóa  $k$  phải đáp ứng 3 điều kiện sau đây:

- Độ dài của khóa phải bằng kích thước bản rõ.
- Khóa phải được chọn hoàn toàn ngẫu nhiên (truly random)
- Và khóa chỉ được sử dụng một lần
- Nếu thỏa mãn 3 điều kiện trên, hệ mã OTP sẽ là an toàn tuyệt đối theo Shanon
- Ngoài thực tế không thể vì không thể trao đổi khóa bí mật

# BLOCK CIPHER

# Block Cipher

- Mã hóa input 1 block đầu ra là 1 block
- 2 block có độ lớn tương đương
- Block DES có độ dài là 64 bit
- Block AES có độ lớn 128 bit
- Block cipher có nhiều mode khác nhau mã hóa plaintext lớn hơn block

# Block Cipher

- DES, 3-DES
- AES (Rijndael)
- RC-2
- RC-5
- IDEA
- Blowfish, Cast
- Gost

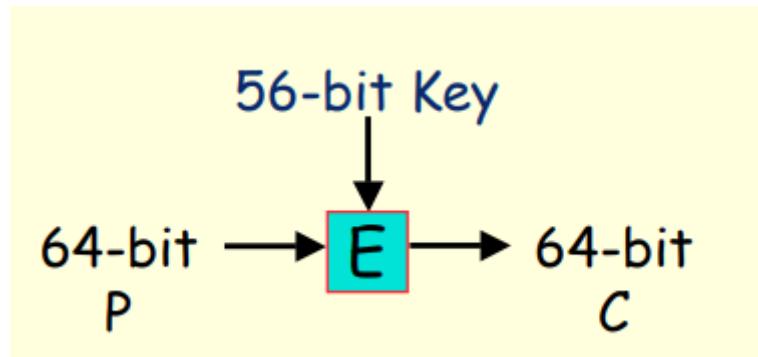
# DES

# ECB Mode Encryption

- ❖ **DES** (viết tắt của **Data Encryption Standard**, hay **Tiêu chuẩn Mã hóa Dữ liệu**) là một phương pháp mã hóa được FIPS(Tiêu chuẩn Xử lý Thông tin Liên bang Hoa Kỳ) chọn làm chuẩn chính thức vào năm 1976.
- ❖ Sau đó chuẩn này được sử dụng rộng rãi trên phạm vi thế giới DES đã được giới nghiên cứu xem xét rất kỹ lưỡng, việc này đã thúc đẩy hiểu biết hiện đại về mật mã khối(*block cipher*) và các phương pháp thám mã tương ứng.
- ❖ Hiện nay DES được xem là không đủ an toàn cho nhiều ứng dụng. Nguyên nhân chủ yếu là độ dài 56 bit của khóa là quá nhỏ.
- ❖ Thuật toán được tin tưởng là an toàn trong thực tiễn có dạng Triple DES (thực hiện DES ba lần), mặc dù trên lý thuyết phương pháp này vẫn có thể bị phá. Gần đây DES đã được thay thế bằng **AES**(*Advanced Encryption Standard*, hay **Tiêu chuẩn Mã hóa Tiên tiến**).

# DES

- ❖ Mỗi lần mã hóa 64 Bits
- ❖ 56 bit làm key
- ❖ Số key có thể là  $2^{56}$
- ❖ DES là một dạng block cipher nhưng có thể có tùy chọn Stream cipher



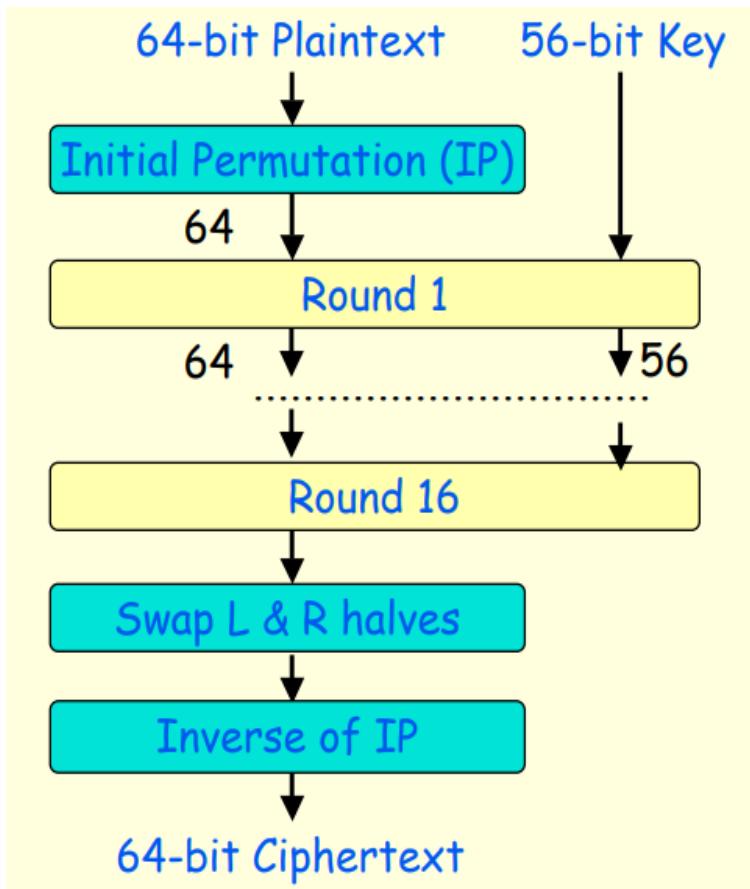
DES

## Tiêu chí thiết kế:

- ❖ Cipher text phải phụ thuộc vào plaintext, key, ...
- ❖ Mỗi bit trên văn bản mã hóa sẽ phụ thuộc tất cả các bit trên văn bản rõ, và tất cả các bit của key
- ❖ Khi thay đổi một bit ở đầu vào thì kéo theo sự thay đổi lớn ở đầu ra. Thay đổi 1 bit key hoặc block sẽ thay đổi 50% bit ở đầu ra



# Cấu Trúc Của DES



## ENC:

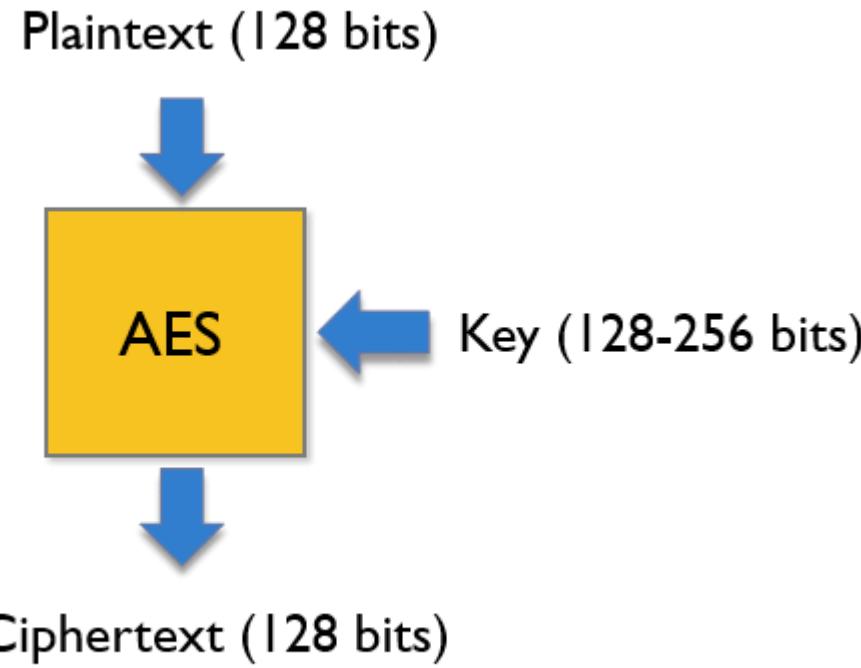
- ❖ Mỗi block qua 16 round thay thế và hoán vị.
- ❖ Mỗi round sử dụng 56 bit của key được gọi là subkey

# AES

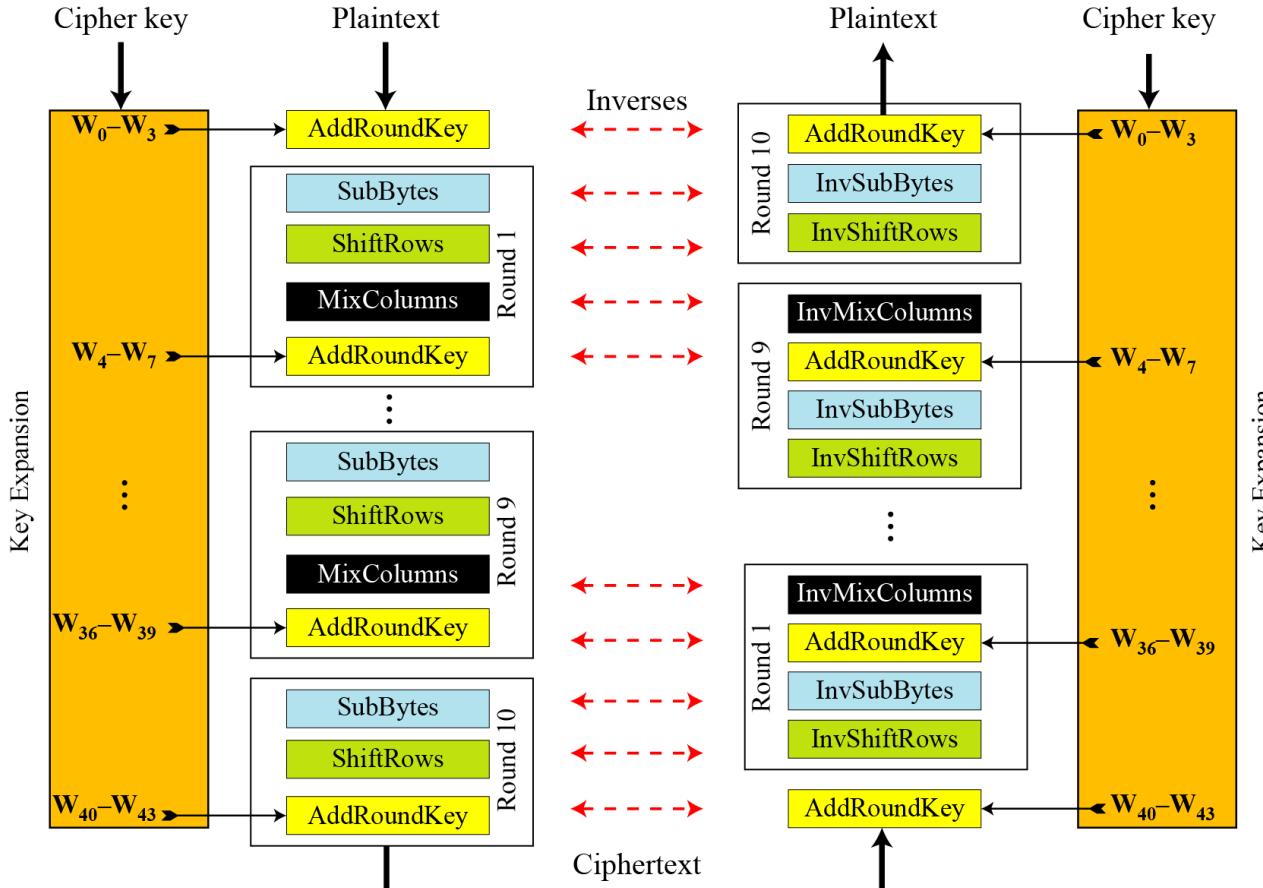
# Giới Thiệu AES

- ❖ Tiêu chuẩn mã hóa nâng cao (AES) được NIST (Viện tiêu chuẩn và công nghệ quốc gia) công bố năm 2001. AES là một mật mã khối đối xứng nhằm thay thế DES làm tiêu chuẩn được phê duyệt cho một loạt các ứng dụng
- ❖ Mật mã AES tạo thành thế hệ mật mã khối mới nhất và bây giờ chúng ta thấy kích thước khối tăng lên đáng kể - từ tiêu chuẩn cũ 64 bit lên đến 128 bit; và các khóa từ 128 đến 256-bit.
- ❖ Mã hóa bí mật đối xứng
- ❖ 128-bit data, 128/192/256-bit keys
- ❖ Stronger & faster than Triple-DES
- ❖ Dễ hiện thực bằng phần cứng và phần mềm

# AES SCHEME

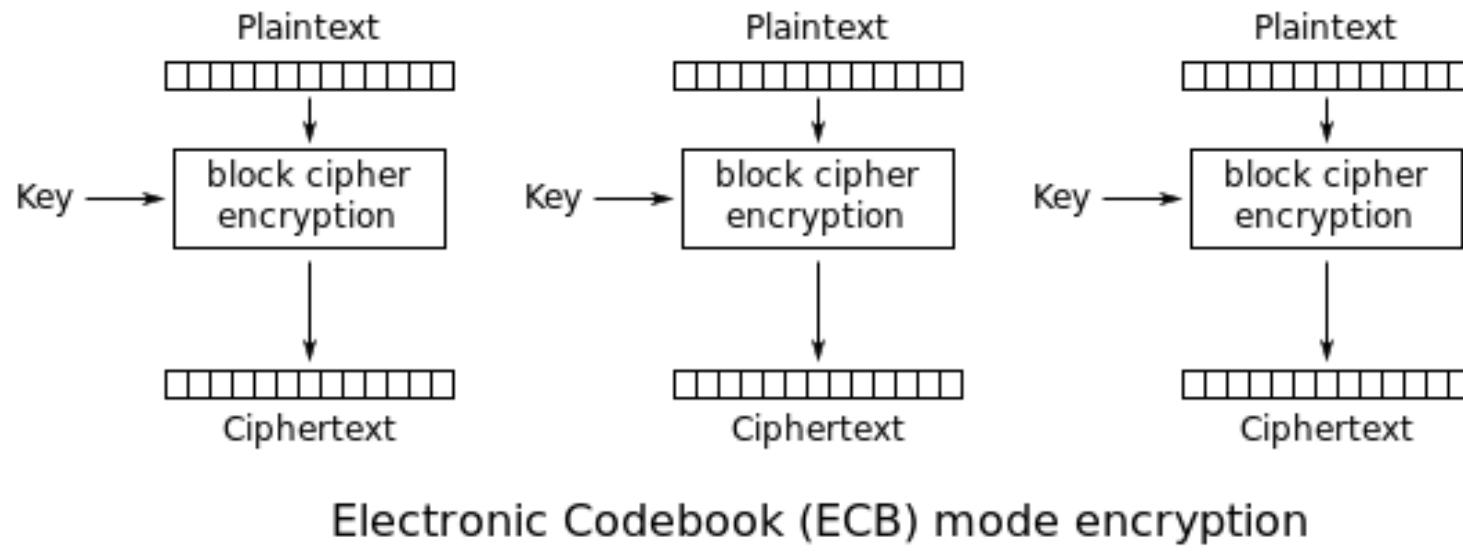


# AES Struct

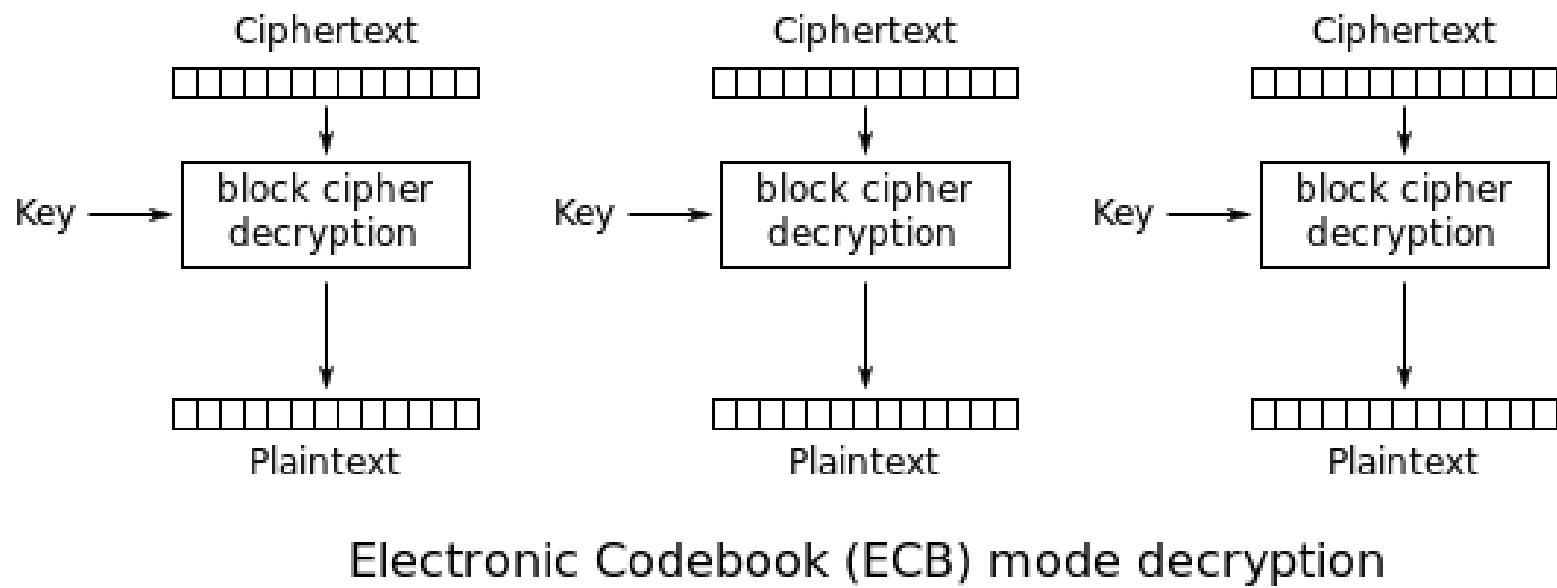


# **ECB MODE ENCRYPTION (ELECTRONIC CODE BOOK)**

# Block Cipher EBC



# Block Cipher EBC

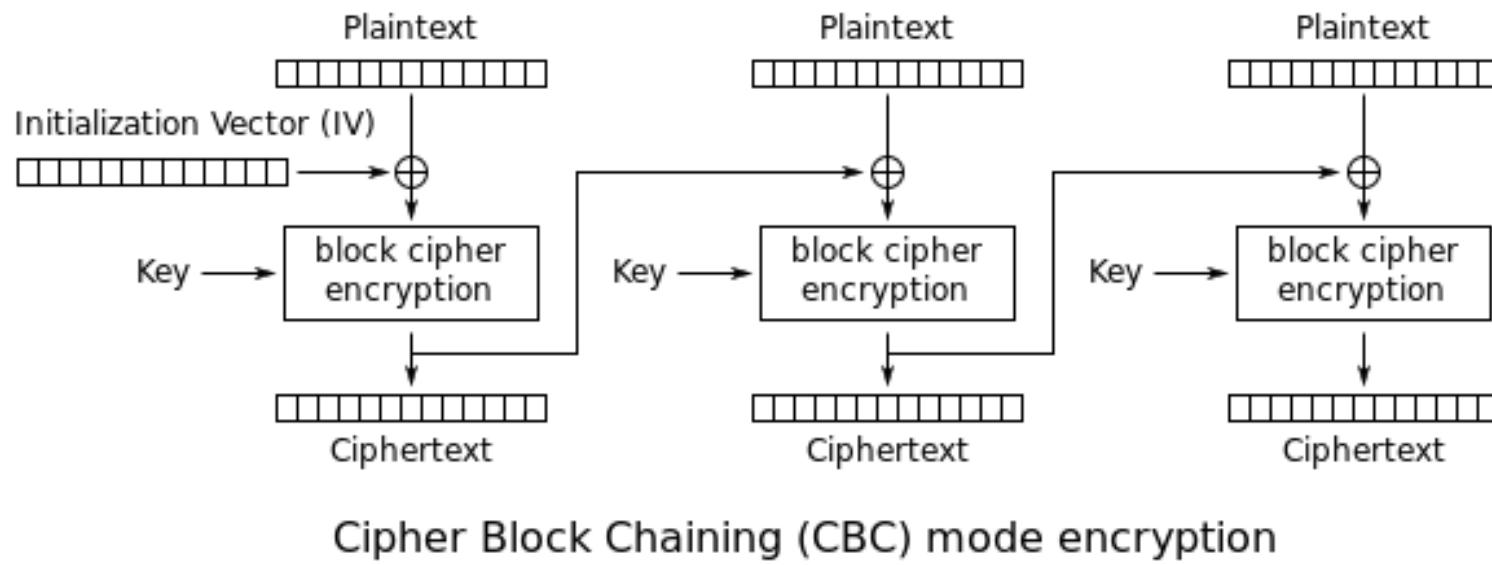


# Block Cipher EBC

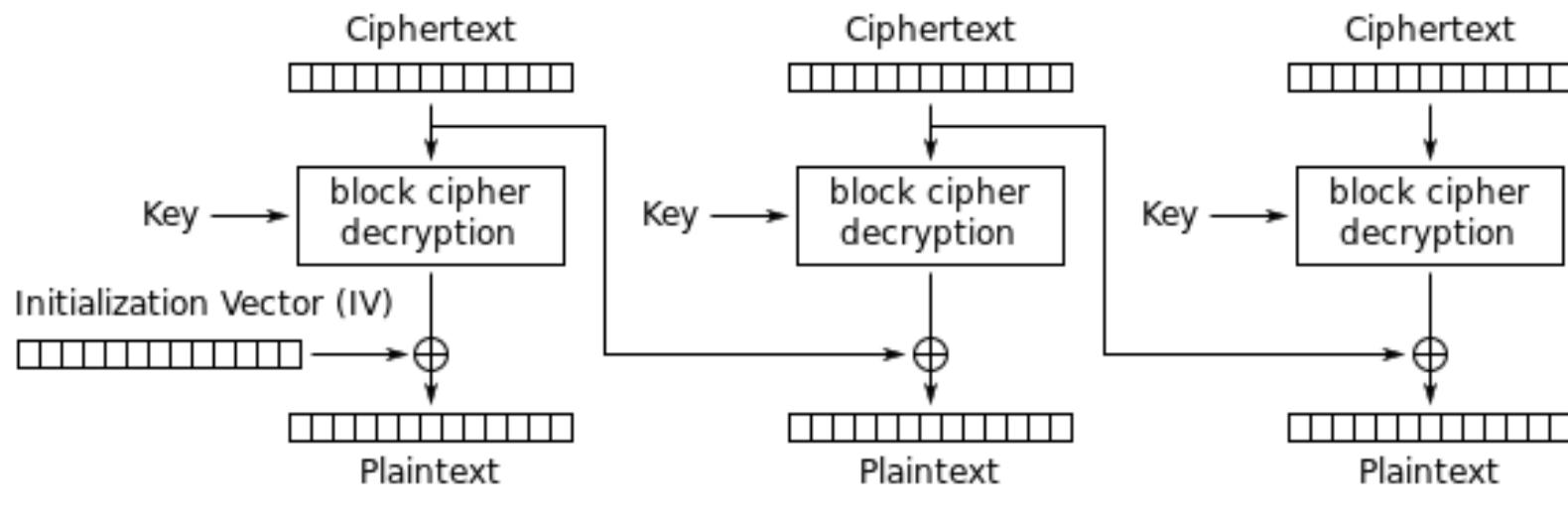
- Đơn giản
- Mã hóa và giải mã nhanh
- Có thể thực hiện song song
- Active attacks có thể xảy ra khi dữ liệu mã hóa nhiều ( trùng lặp lại block, loại trừ block, hay trao đổi block)

# **CBC MODE ENCRYPTION (CIPHER BLOCK CHAINING)**

# Block Cipher CBC



# Block Cipher CBC



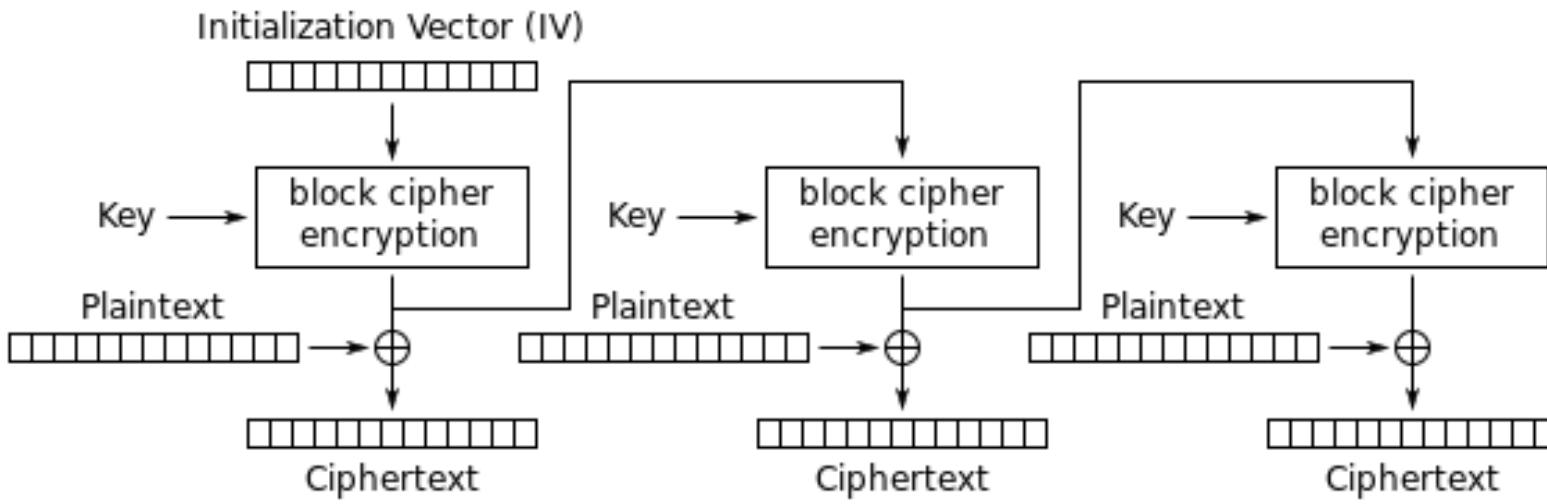
Cipher Block Chaining (CBC) mode decryption

# Block Cipher CBC

- Mật mã không đồng bộ
- Lỗi trong 1 block sẽ làm lan truyền ra các block khác
- Không thể thực hiện song song
- Là chuẩn trong hầu hết hệ thống SLL, IPSEC,..
- Không thể sử dụng để tấn công như mode EBC

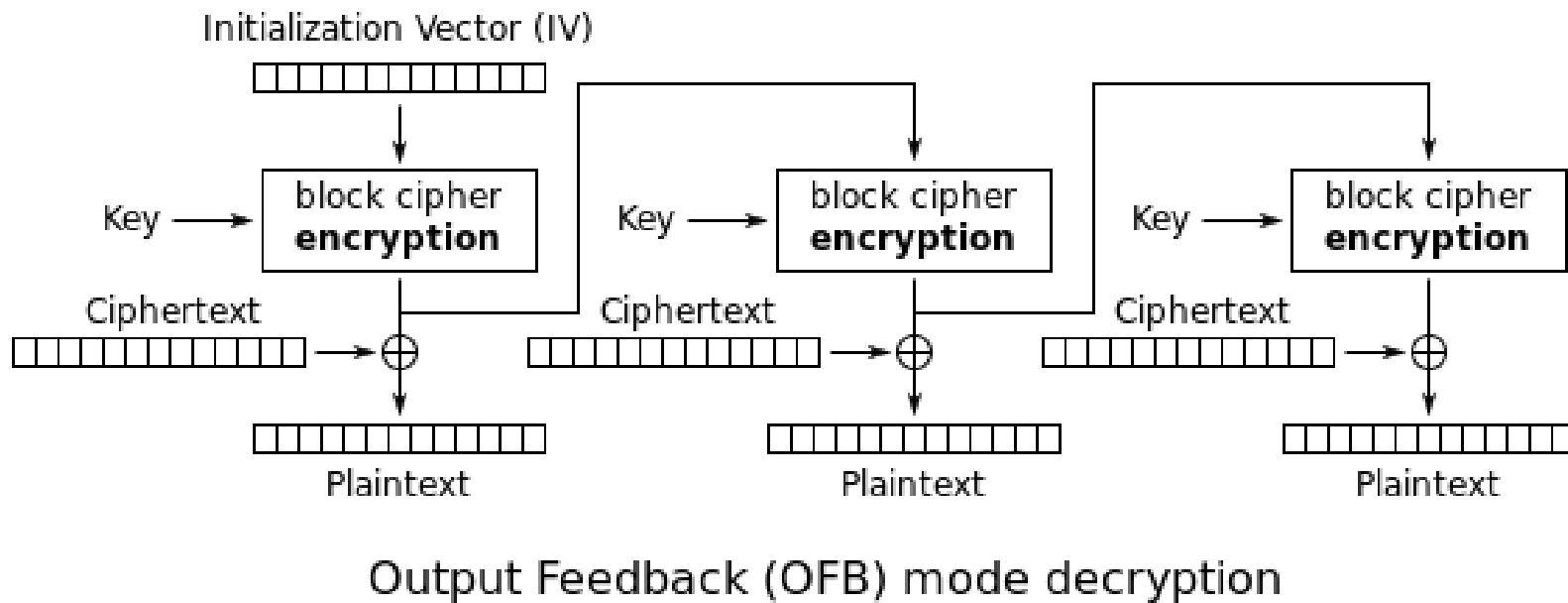
# **OFB MODE (OUTPUT FEEDBACK)**

# Block Cipher OFB Mode



## Output Feedback (OFB) mode encryption

# Block Cipher OFB Mode

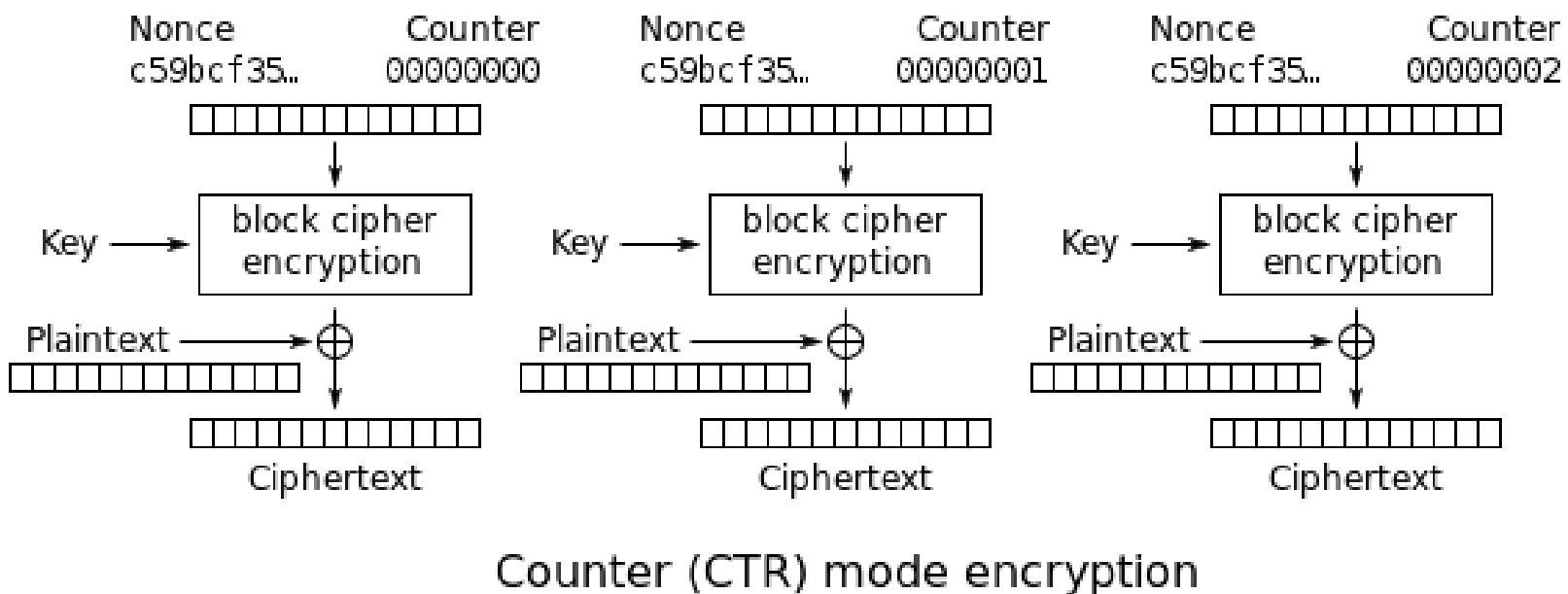


# Block Cipher OFB

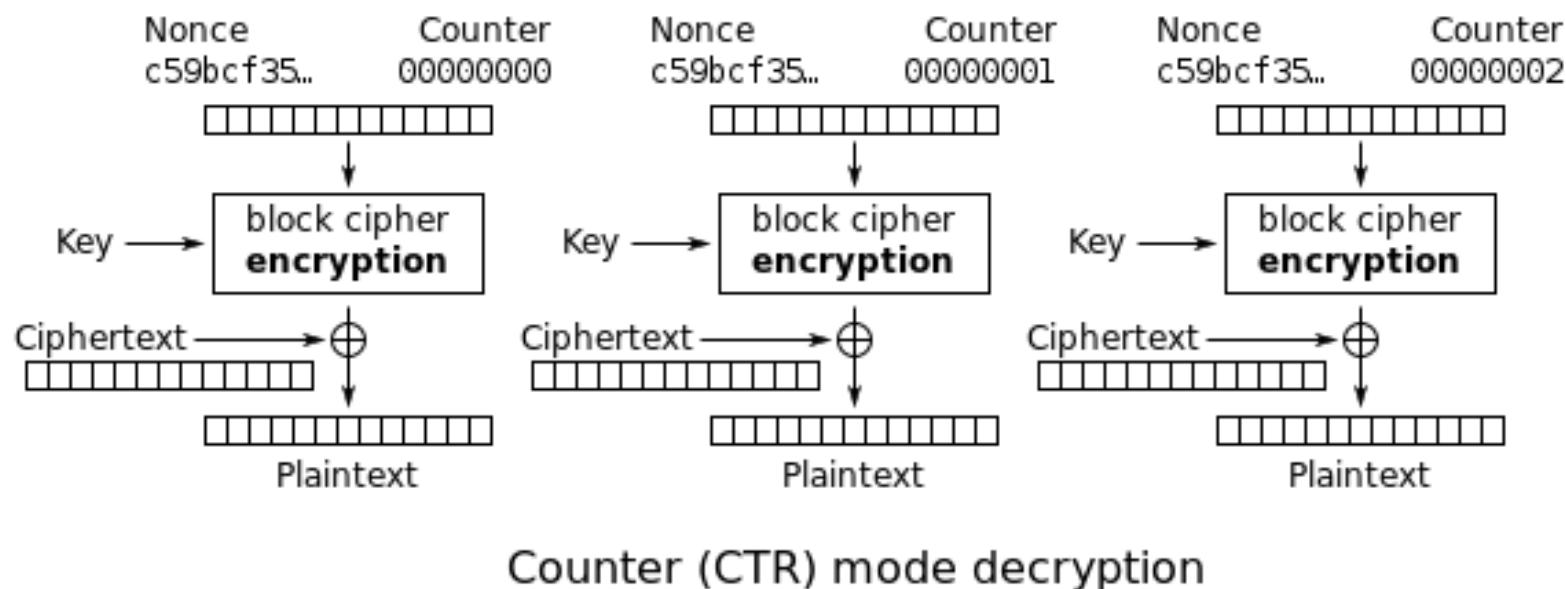
- Mật mã đồng bộ
- Lỗi trong 1 block không lan truyền
- Không thể thực hiện song song
- Che giấu các mẫu văn bản
- Tấn công tích cực bằng cách thao túng văn bản

# CTR (COUNTER) MODE

# Block Cipher CTR Mode



# Block Cipher CTR Mode



# Block Cipher CTR Mode

- Xử lý song song
- Tăng bảo mật
- Đơn giản, hiệu quả

# TĂNG ĐỘ MẠNH CHO MÃ HÓA CÔNG KHAI

# Tăng độ mạnh cho mã hóa công khai

- Thiết kế hệ mã đa dạng key lengths - AES
- Whitening - the DESX ( kết hợp dữ liệu với các phần của key)
- Lắp mã – Triple DES (3-DES), triple IDEA



# **ƯU/ NHƯỢC ĐIỂM**

## Ưu/Nhược Điểm

- Khi sử dụng mã hóa đối xứng với thuật toán bảo mật (AES -256) cực kì an toàn( 1 tỷ năm mới có thể tấn công brutal force)
- Một trong những nhược điểm của các hệ thống mã hóa khóa công khai là chúng cần toán học tương đối phức tạp để hoạt động, khiến chúng rất chuyên sâu về mặt tính toán.
- Mã hóa và giải mã dữ liệu khóa đối xứng(private key) tương đối dễ thực hiện, có hiệu suất cao.

## Ưu/Nhược Điểm

- Bài toán khó nhất của Mã hóa đối xứng chính là bài toán chia sẻ khóa (khó khăn trong chia sẻ khóa bí mật)
- Khi ai đó biết được khóa đối xứng họ có thể giải mã tất cả thông điệp khóa bằng khóa này (2 chiều gửi-nhận). Mã hóa công khai thì chỉ có có thể 1 chiều (gửi – nhận)

# Mã Hóa Ứng dụng-Chương 4



Đoàn Trình Dục/ Đoàn Trình Dục

Mail: duc.duantrinh@stu.edu.vn

2019



# NỘI DUNG CHÍNH



Giới Thiệu Mã Hóa Công Khai



Số Nguyên Tố



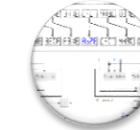
Hệ Mã Công Khai



Giao thức trao đổi khóa



RSA

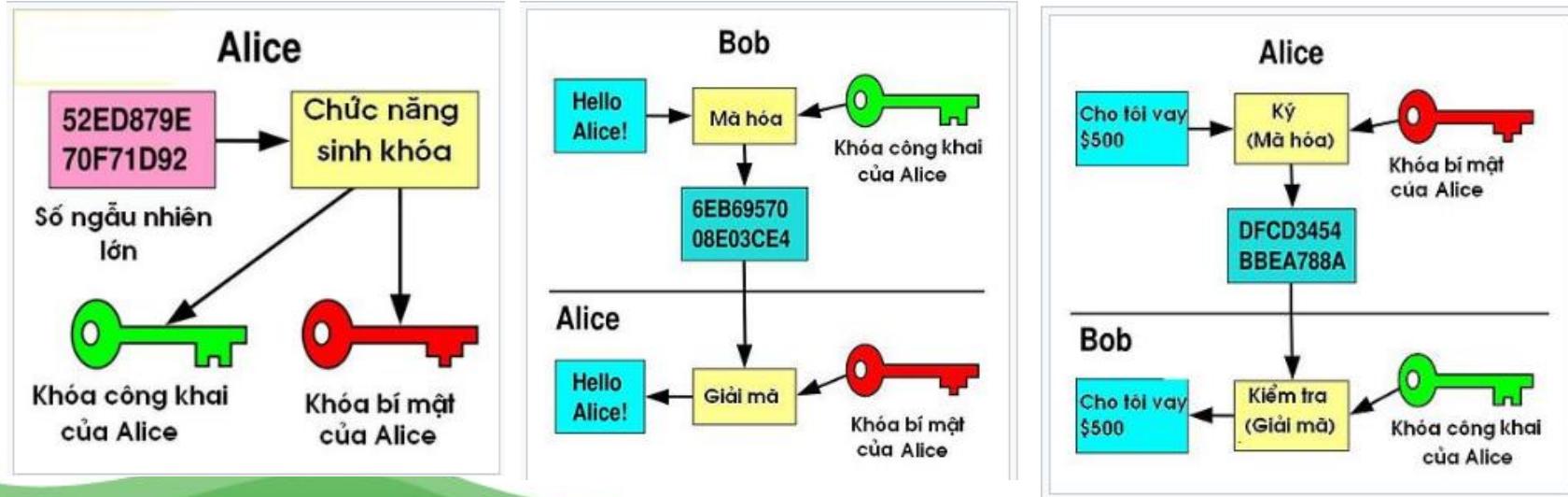


Đường Cong eliptic

# **GIỚI THIỆU MÃ HÓA CÔNG Khai**

# Giới Thiệu

- **Mật mã hóa khóa công khai** là một dạng cho phép người sử dụng trao đổi các mật mã mà không cần phải trao đổi các chung trước đó.
- Điều này được thực hiện bằng cách sử dụng một cặp khóa có quan hệ với nhau là khóa công khai (hay khóa bí mật).
- Thuật ngữ **mật mã hóa khóa bất đối xứng** thường được dùng đồng nghĩa với **mật mã hóa khóa công khai** mặc dù hai khái niệm không hoàn toàn tương đương.



# SỐ NGUYÊN TỐ

# Số nguyên Tố

- Bất kỳ số nguyên  $a > 1$  đều có thể viết dưới dạng:  
 $p_1 p_2 p_3 \dots p_n$  (với  $p_i$  là số nguyên tố)
- Một số nguyên  $p > 1$  là số nguyên tố nếu và chỉ nếu ước duy nhất của nó là  $\pm 1$  và  $\pm p$ .
- Số nguyên tố đóng vai trò quan trọng trong lý thuyết số và trong các kỹ thuật mã hoá khoá
- 200 số nguyên tố đầu tiên

# Thuật toán tìm số nguyên tố

```
L = {2, 3, ..., n};  
i = 1;  
While (L[i]2 <= n) Do {  
    If (L[i] <> 0)  
        k = i2 + 2i;  
        While (k <= n) Do {  
            L[k] = 0;  
            k = k + i;  
        }  
    i++;  
}
```

# 200 số nguyên tố đầu tiên

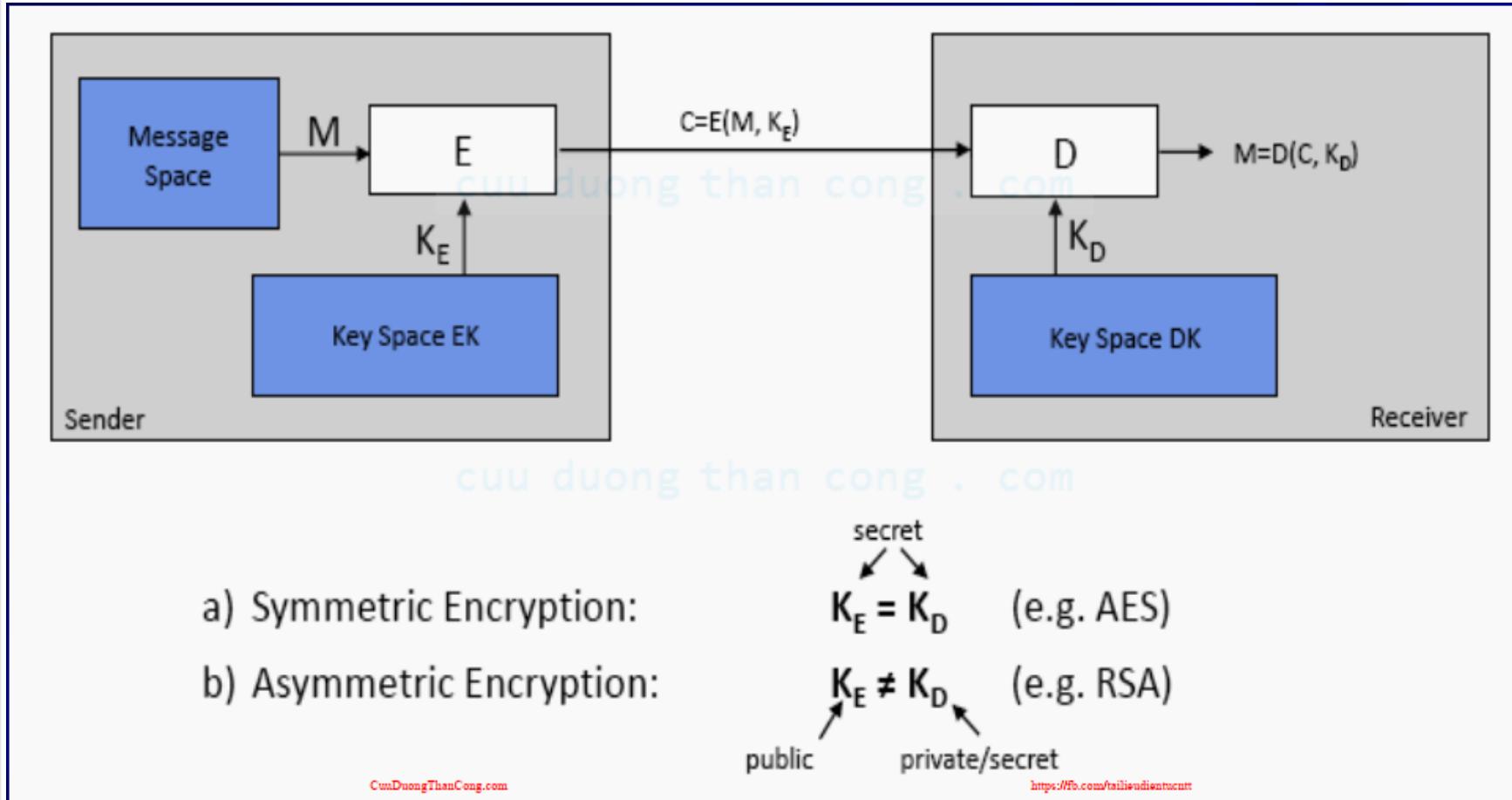
2	101	211	307	401	503	601	701	809	0	1009	1103	1201	1301	1409	1511	1601	1709	1801	1901
3	103	223	311	409	509	607	709	811	911	1013	1109	1213	1303	1423	1523	1607	1721	1811	1907
5	107	227	313	419	521	613	719	821	919	1019	1117	1217	1307	1427	1531	1609	1723	1823	1913
7	109	229	317	421	523	617	727	823	929	1021	1123	1223	1319	1429	1543	1613	1733	1831	1931
11	113	233	331	431	541	619	733	827	937	1031	1129	1229	1321	1433	1549	1619	1741	1847	1933
13	127	239	337	433	547	631	739	829	941	1033	1151	1231	1327	1439	1553	1621	1747	1861	1949
17	131	241	347	439	557	641	743	839	947	1039	1153	1237	1361	1447	1559	1627	1753	1867	1951
19	137	251	349	443	563	643	751	853	953	1049	1163	1249	1367	1451	1567	1637	1759	1871	1973
23	139	257	353	449	569	647	757	857	967	1051	1171	1259	1373	1453	1571	1657	1777	1873	1979
29	149	263	359	457	571	653	761	859	971	1061	1181	1277	1381	1459	1579	1663	1783	1877	1987
31	151	269	367	461	577	659	769	863	977	1063	1187	1279	1399	1471	1583	1667	1787	1879	1999
37	157	271	373	463	587	661	773	877	983	1069	1193	1283		1481	1597	1669	1789	1889	1997
41	163	277	379	467	593	673	787	881	991	1087		1289		1483		1693			1999
43	167	281	383	479	599	677	797	883	997	1091		1291		1487		1697			
47	173	283	389	487		683		887		1093		1297		1489		1699			
53	179	293	397	491		691				1097					1493				
59	181			499										1499					
61	191																		
67	193																		
71	197																		
73	199																		
79																			
83																			
89																			
97																			

# Thuật toán tìm số nguyên tố

- **Thuật toán tìm dãy số nguyên tố nhỏ hơn n (Eratosthenes).**
  - Liệt kê tất cả các số nguyên từ 2 đến n.
  - Số đầu tiên (2) là số nguyên tố.
  - Loại tất cả các bội của 2 ra khỏi bảng.
  - Số nguyên ngay sau số 2 sau khi loại (sàng) là số nguyên tố (số 3).
  - Loại bỏ tất cả các bội của 3.
  - ...
  - Khi tìm được một số nguyên tố lớn hơn căn bậc 2 của n, tất cả các số còn lại không bị loại ra đều là số nguyên tố

# HỆ MÃ HÓA CÔNG KHAI

# Hệ Mã Khóa Công Khai



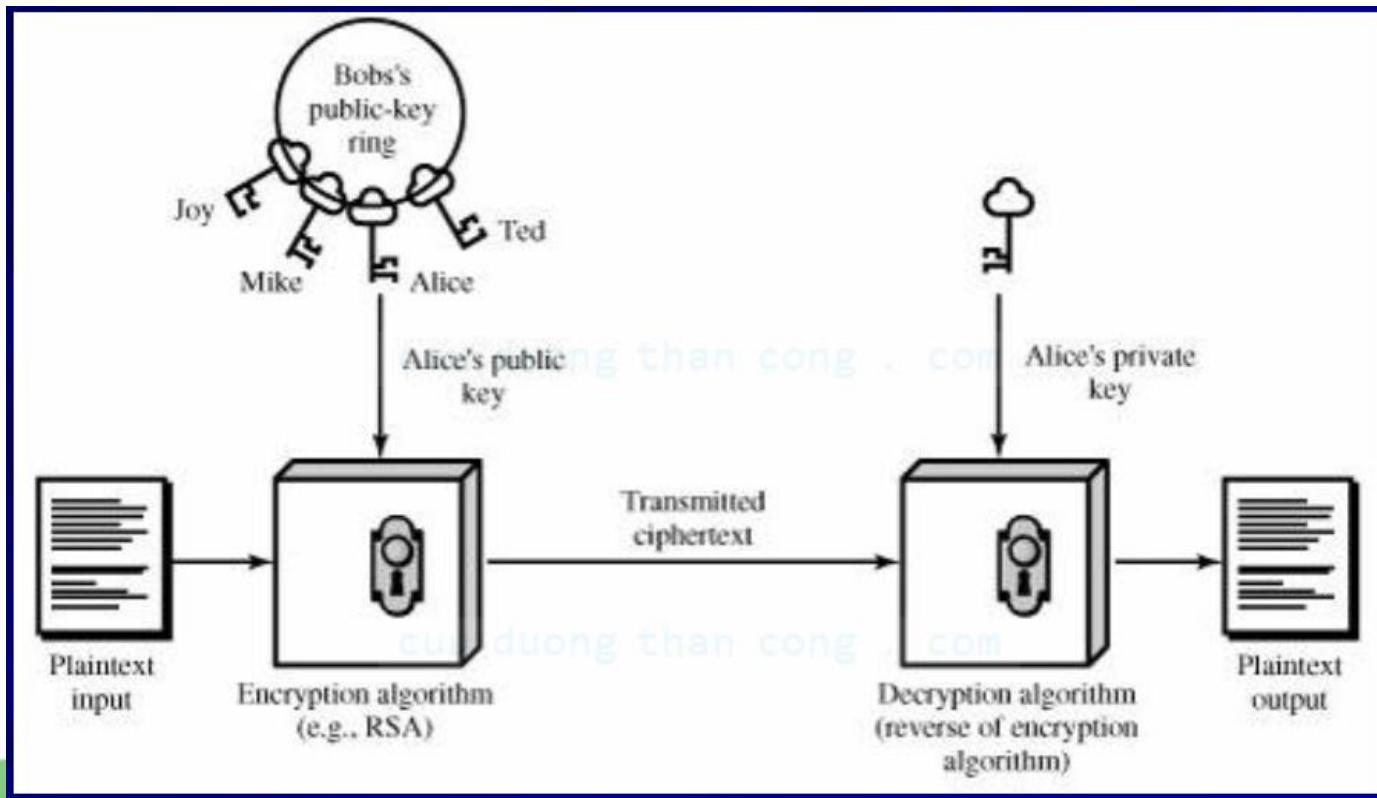
# Hệ Mã Khóa Công Khai

**Các bước chủ yếu khi thực hiện mã hoá khoá công khai:**

1. Mỗi user tạo ra một cặp khoá được sử dụng cho việc mã hoá và giải mã thông điệp.
2. Mỗi user đặt một trong hai khoá trong một đăng ký công cộng. Đây là khoá công khai. Khoá còn lại được giữ kín.
3. Nếu Bob muốn gửi một tin nhắn bí mật cho Alice, Bob mã hoá tin nhắn này bằng cách sử dụng khoá công khai của Alice.
4. Khi Alice nhận được tin nhắn, cô giải mã nó bằng cách sử dụng khoá riêng của mình. Không có ai khác có thể giải mã thông điệp bởi vì chỉ có Alice biết khoá riêng của Alice.

# Ứng Dụng Hệ Mã Hóa Công Khai

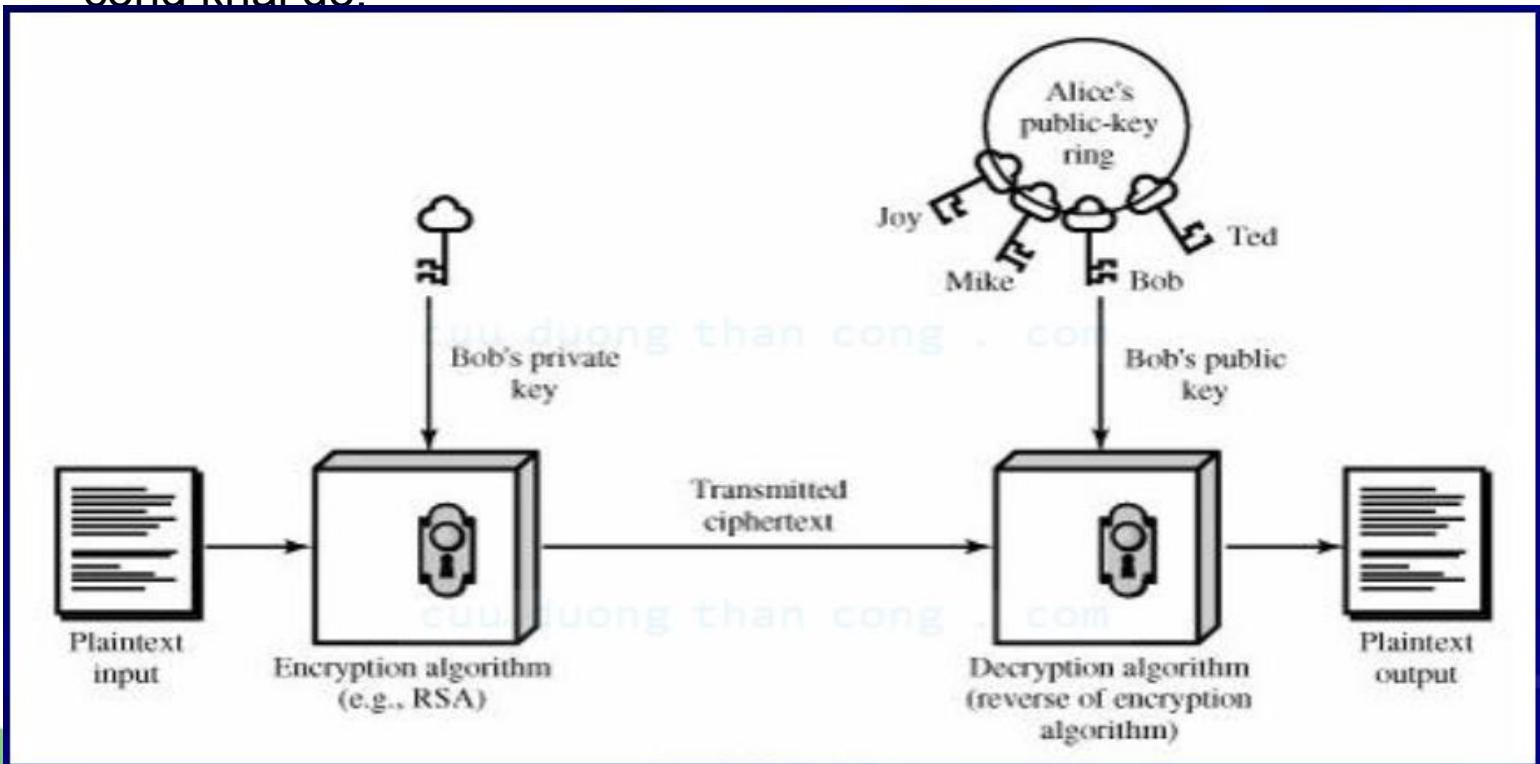
Ứng dụng thông dụng nhất của mật mã hoá công khai là bảo mật (mã hoá/giải mã): một văn bản được mã hoá bằng **khoá công khai** của một người sử dụng thì chỉ có thể giải mã với **khoá bí mật** của người đó.



# Ứng Dụng Hệ Mã Hóa Công Khai

## Xác thực:

- Một người sử dụng có thẻ mã hoá văn bản với khoá bí mật của mình.
- Nếu một người khác có thẻ giải mã với **khoá công khai** của người gửi thì có thể tin rằng văn bản thực sự xuất phát từ người gắn với khoá công khai đó.



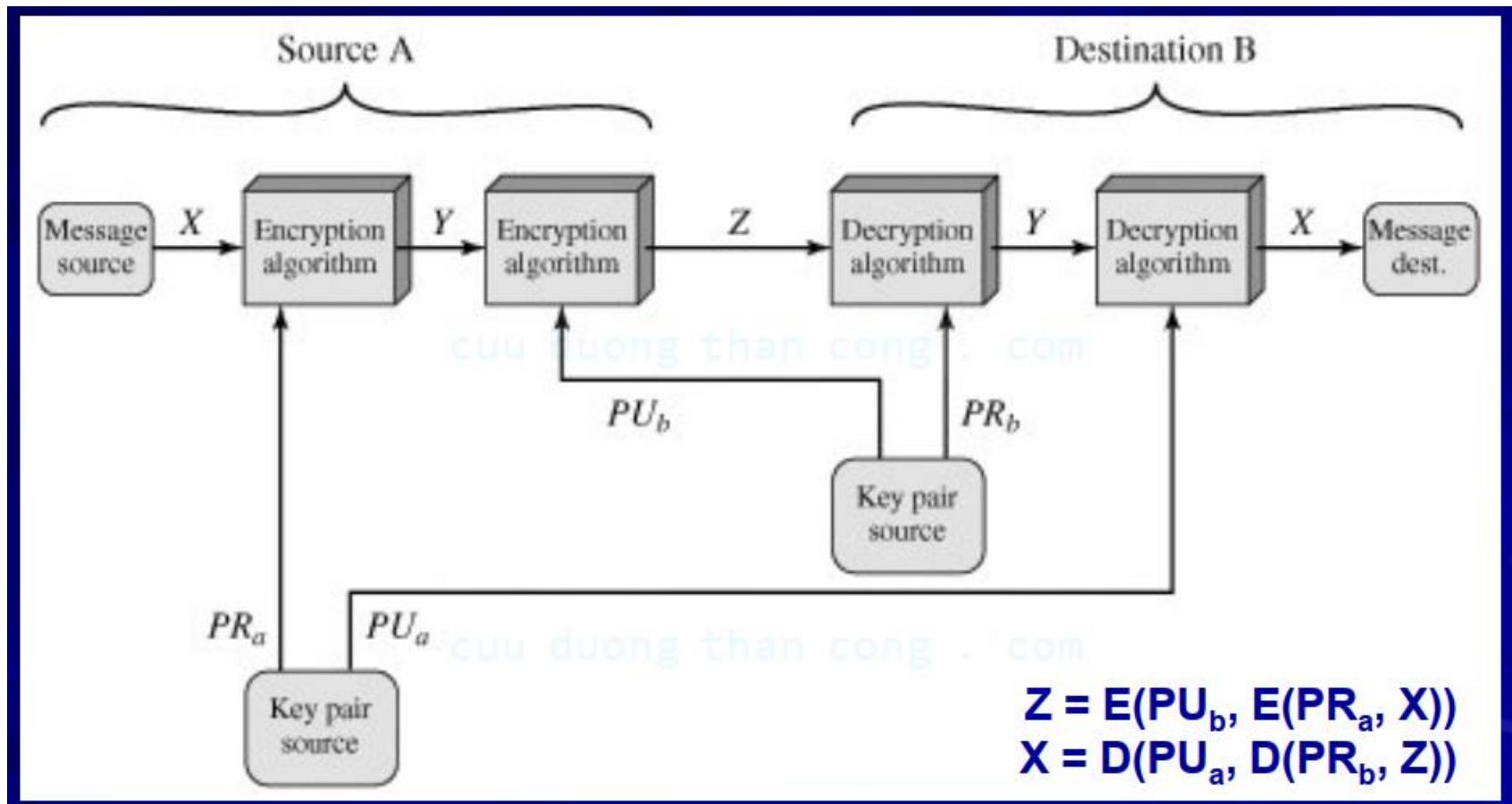
# Ứng Dụng Hệ Mã Hóa Công Khai

## Trao đổi key:

- Hai bên hợp tác để trao đổi session key
- Trước tiên, mã hoá thông điệp X sử dụng khoá secret của người gửi (cung cấp chữ ký số) để được Y.
- Kế đó, mã hoá tiếp Y với khoá public của người nhận.
- Chỉ có người nhận đã xác định trước mới có khoá secret của người nhận và khoá public của người gửi để giải mã hai lần để được X.

# Ứng Dụng Hệ Mã Hóa Công Khai

Trao đổi key:



# Một Số Thuật Toán

Algorithm	Encryption/ Decryption	Digital Signature	Key Exchange
RSA	X	X	X
Elliptic Curve	X	X	X
Diffie-Hellman			X
DSS		X	

# Hệ Mã Hóa Công Khai

**Định nghĩa:** Cho các tập hữu hạn  $S$  và  $T$ . Hàm một chiều  $f: S \rightarrow T$  là hàm khả nghịch thoả:

- $f$  dễ thực hiện; cho  $x \in S$ , dễ dàng tính được  $y = f(x)$
- $f^{-1}$  là hàm ngược của  $f$ , khó thực hiện; cho  $y \in T$ , rất khó tính được  $x = f^{-1}(y)$ .
- $f^{-1}$  chỉ có thể tính được khi biết thêm một số thông tin cần thiết

**Ví dụ:**

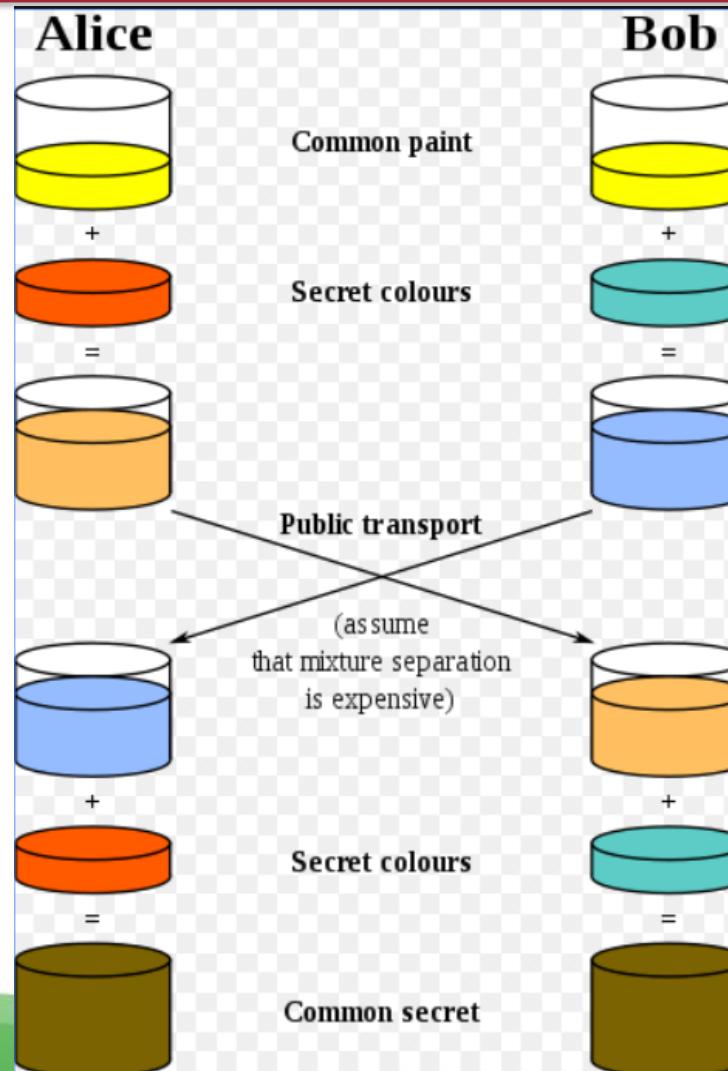
$f(x): p * q \rightarrow n$  là hàm một chiều với  $p$  và  $q$  là các số nguyên tố lớn. Có thể dễ dàng thực hiện phép nhân  $pq$  (độ phức tạp đa thức). Tính  $f^{-1}$  (phân tích ra thừa số nguyên tố - độ phức tạp mũ) là bài toán cực kỳ khó)

# GIAO THỨC TRAO ĐỔI KHÓA DIFFIE-HELLMAN

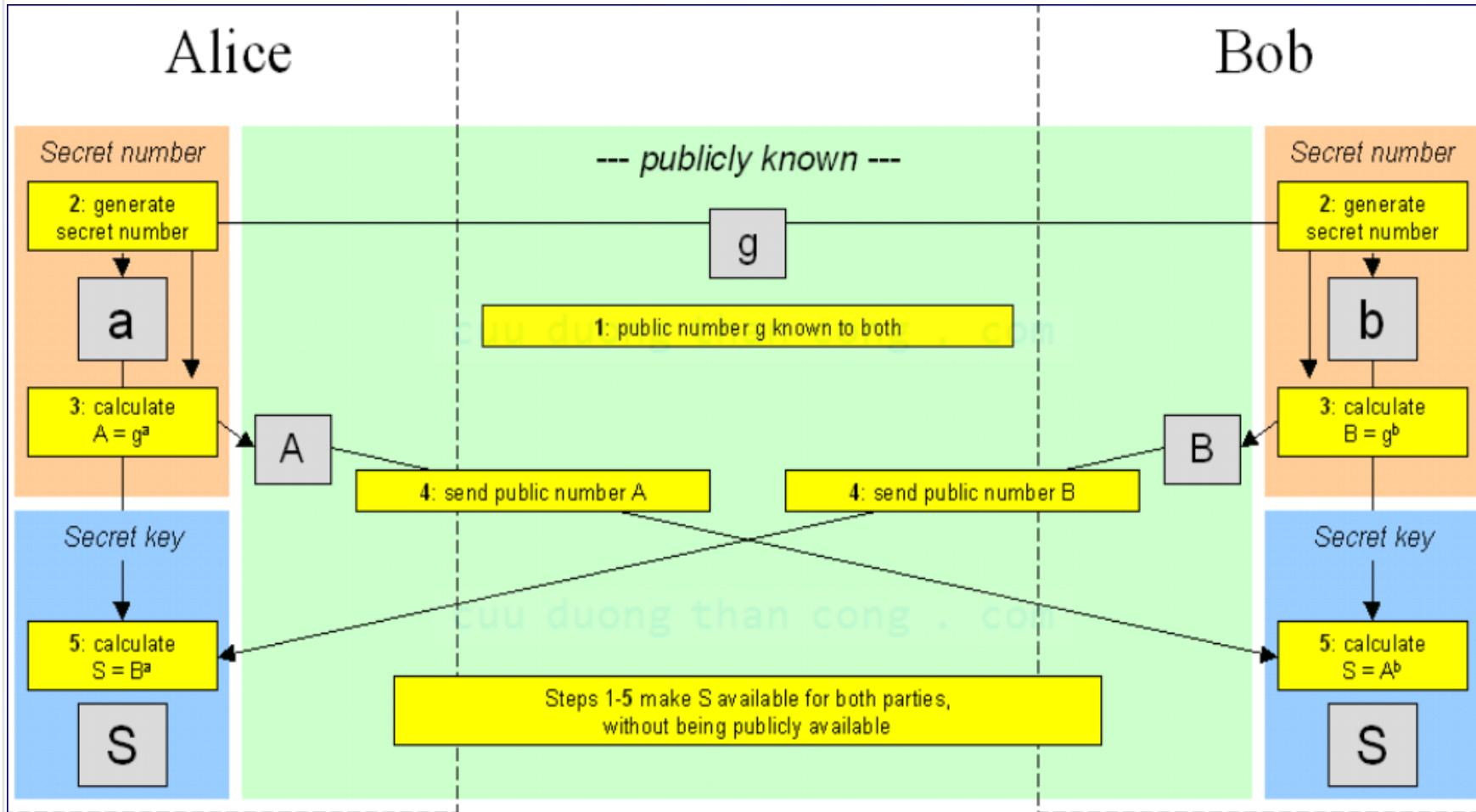
# Giao Thức Trao Đổi Khóa Diffie-Hellman

- Mục đích của thuật toán là cho phép hai người dùng trao đổi khóa bí mật dùng chung trên mạng công cộng, sau đó có thể sử dụng để mã hóa các thông điệp
- Thuật toán tập trung vào giới hạn việc trao đổi các giá trị bí mật, xây dựng dựa trên bài toán khó logarit rời rạc.

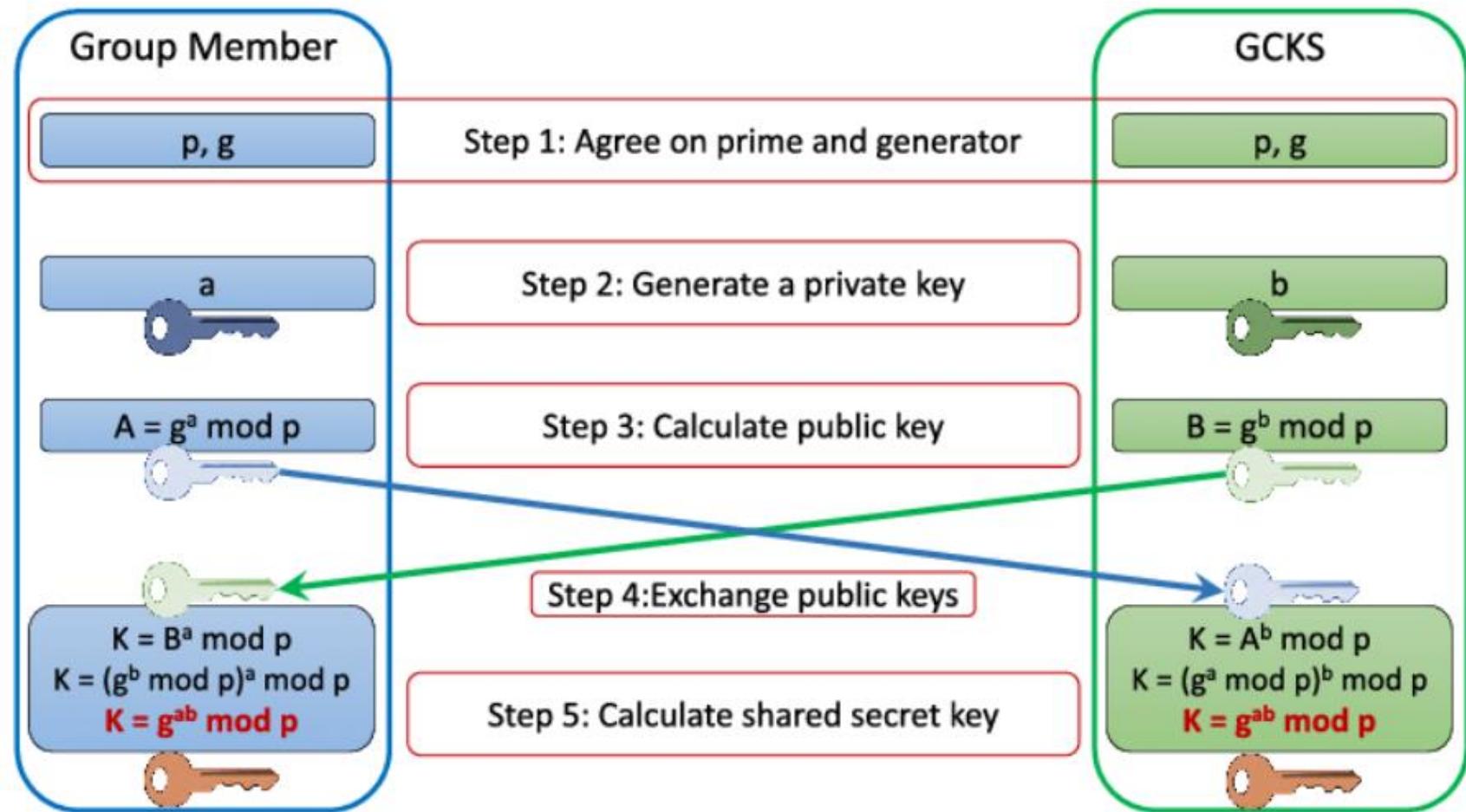
# Giao Thức Trao Đổi Khóa Diffie-Hellman



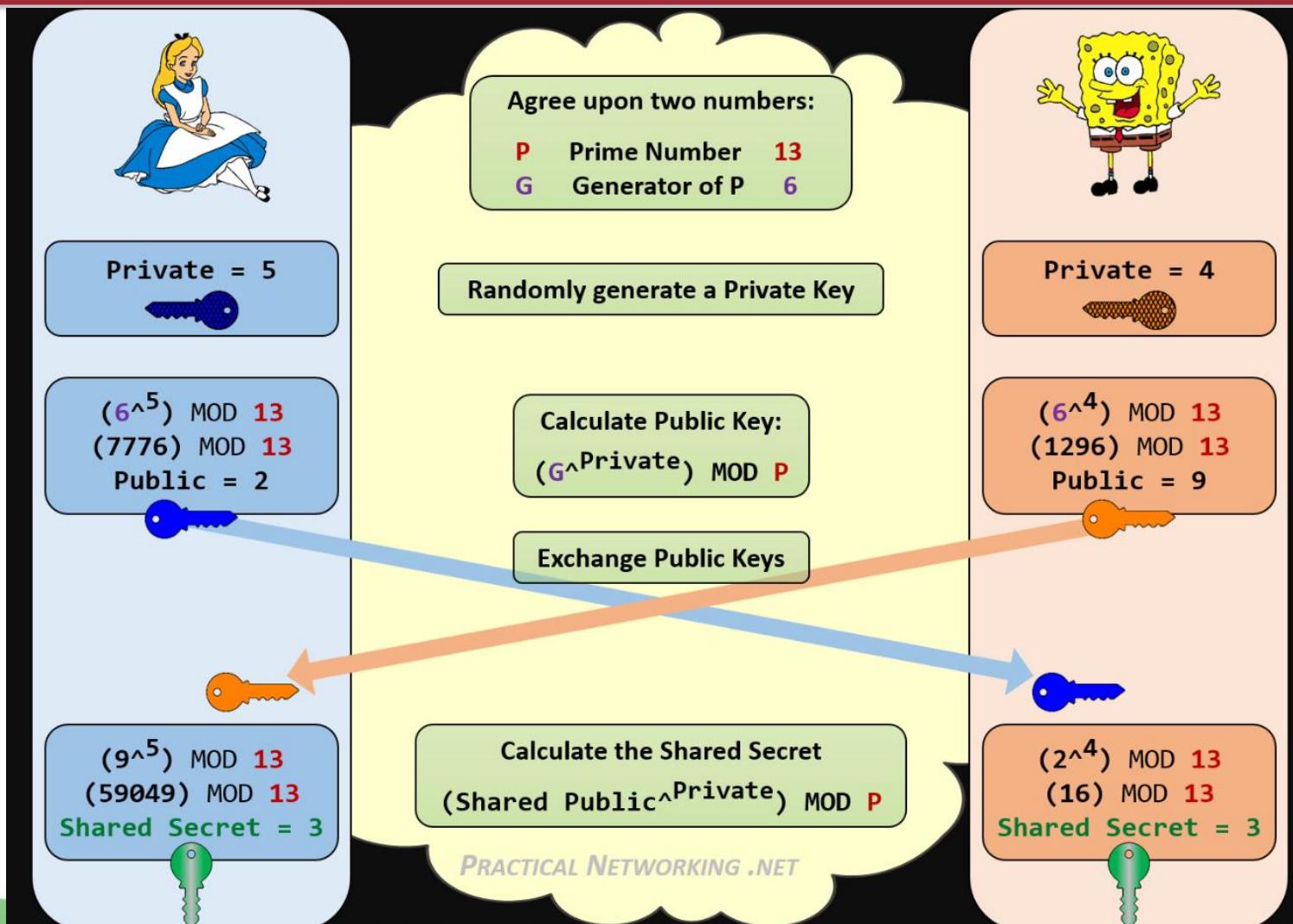
# Giao Thức Trao Đổi Khóa Diffie-Hellman



# Giao Thức Trao Đổi Khóa Diffie-Hellman



# Giao Thức Trao Đổi Khóa Diffie-Hellman



## Giao Thức Trao Đổi Khóa Diffie-Hellman

A và B chọn số nguyên tố chung là  $p=353$  và phần tử sinh  $g$  là 3.

- A chọn  $a=97$  rồi gởi cho B giá trị kết quả của  $3^{97} \text{ mod } 353 = 40$ .
- B chọn  $b=233$  rồi gởi cho A giá trị kết quả của  $3^{233} \text{ mod } 353 = 248$ .
- Cả A và B đều tính được

$$K = 248^{97} \text{ mod } 353 = 160 = 40^{233} \text{ mod } 353$$

# HỆ MÃ RSA

# Hệ Mã RSA

- Giải thuật được phát triển bởi Rivest, Shamir và Adleman, sử dụng một biểu thức với hàm mũ.
- Văn bản rõ được mã hóa ở dạng khối, kích cỡ của khối phải nhỏ hơn hoặc bằng  $\log_2(n)$ .
- Trong thực tế, kích thước khối là  $i$  bit, với  $2^i < n \leq 2^{i+1}$
- Mã hóa và giải mã được thực hiện với một số khối rõ  $M$  (plaintext) và khối mã  $C$  (ciphertext):  
$$C = M^e \text{ mod } n$$
$$M = C^d \text{ mod } n = (M^e)^d \text{ mod } n = M^{e*d} \text{ mod } n$$

# Hệ Mã RSA

## Giải thuật:

- Mã hoá:

Từ khoá công khai  $(n, e)$  và thông điệp là plaintext dưới dạng số nguyên  $M \in [0, n)$ .

Tính cyphertext  $C = M^e \text{ mod } n$

- Giải mã:

$M = C^d \text{ mod } n$ , với  $e * d \text{ mod } n = 1$  là khoá bí mật.

# Hệ Mã RSA

**Các yêu cầu sau đây phải được đáp ứng:**

- Phải có khả năng tìm được giá trị của e, d, n sao cho  $M^{ed} \text{ mod } n = M$ , với  $M < n$ .
- Phải dễ dàng tính toán được:  $M^e \text{ mod } n$  và  $C^d \text{ mod } n$  cho tất cả các giá trị của  $M < n$ .
- Không khả thi để xác định d khi cho e và n.
- Để an toàn, RSA đòi hỏi p và q phải là các số nguyên tố rất lớn để không thể phân tích được  $n=pq$

# Hệ Mã RSA

## Key Generation

Select  $p, q$

$p$  and  $q$  both prime,  $p \neq q$

Calculate  $n = p \times q$

cuu duong than cong . com

Calculate  $\phi(n) = (p - 1)(q - 1)$

Select integer  $e$

$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$

Calculate  $d$

$d \equiv e^{-1} \pmod{\phi(n)}$

Public key

cuu duong than cong . com  
 $PU = \{e, n\}$

Private key

$PR = \{d, n\}$

# Hệ Mã RSA

## Encryption

Plaintext:

$$M < n$$

Ciphertext:

$$C = M^e \bmod n$$

## Decryption

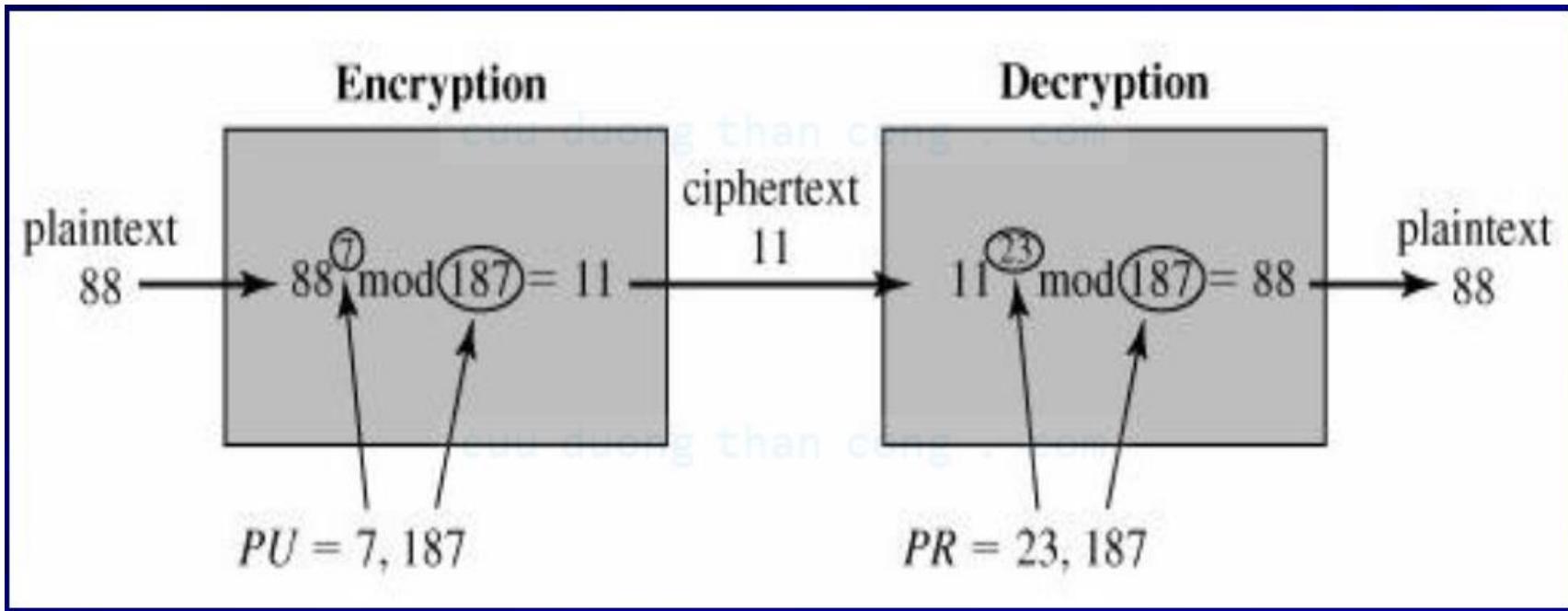
Ciphertext:

$$\text{cuu duong than cong . com}$$

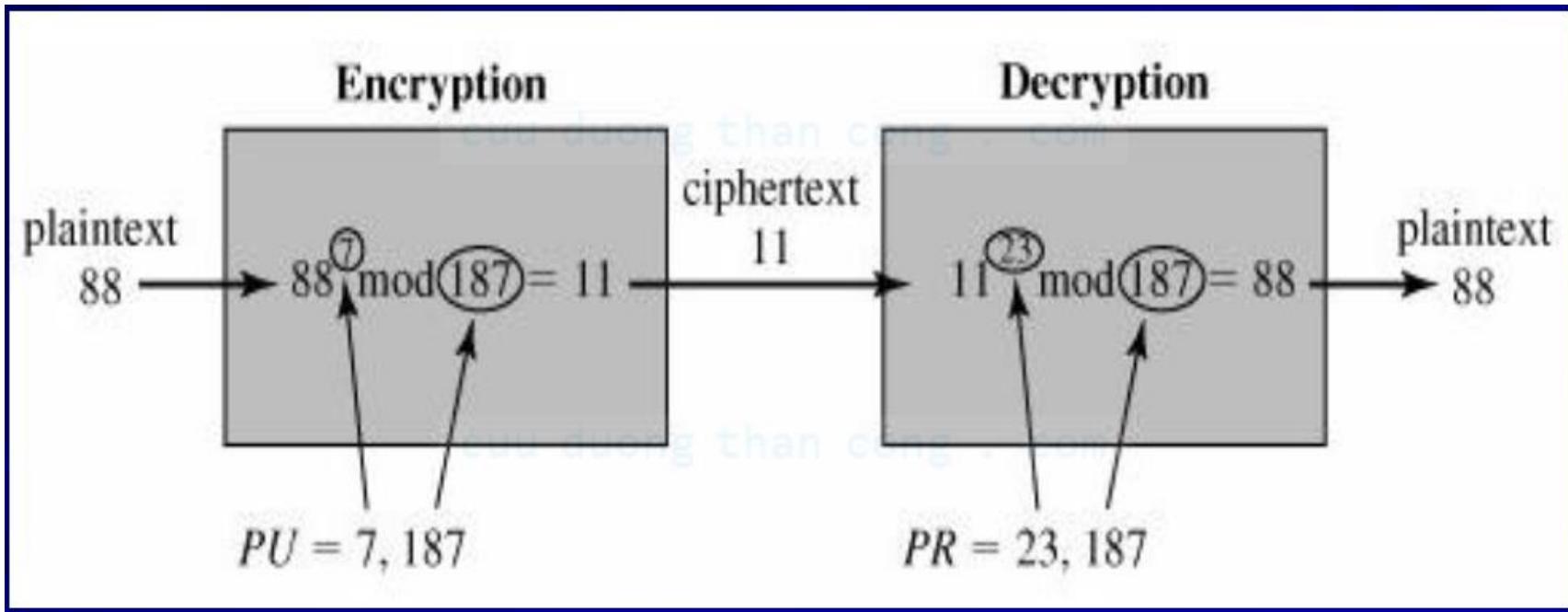
Plaintext:

$$M = C^d \bmod n$$

# Hệ Mã RSA



# Hệ Mã RSA



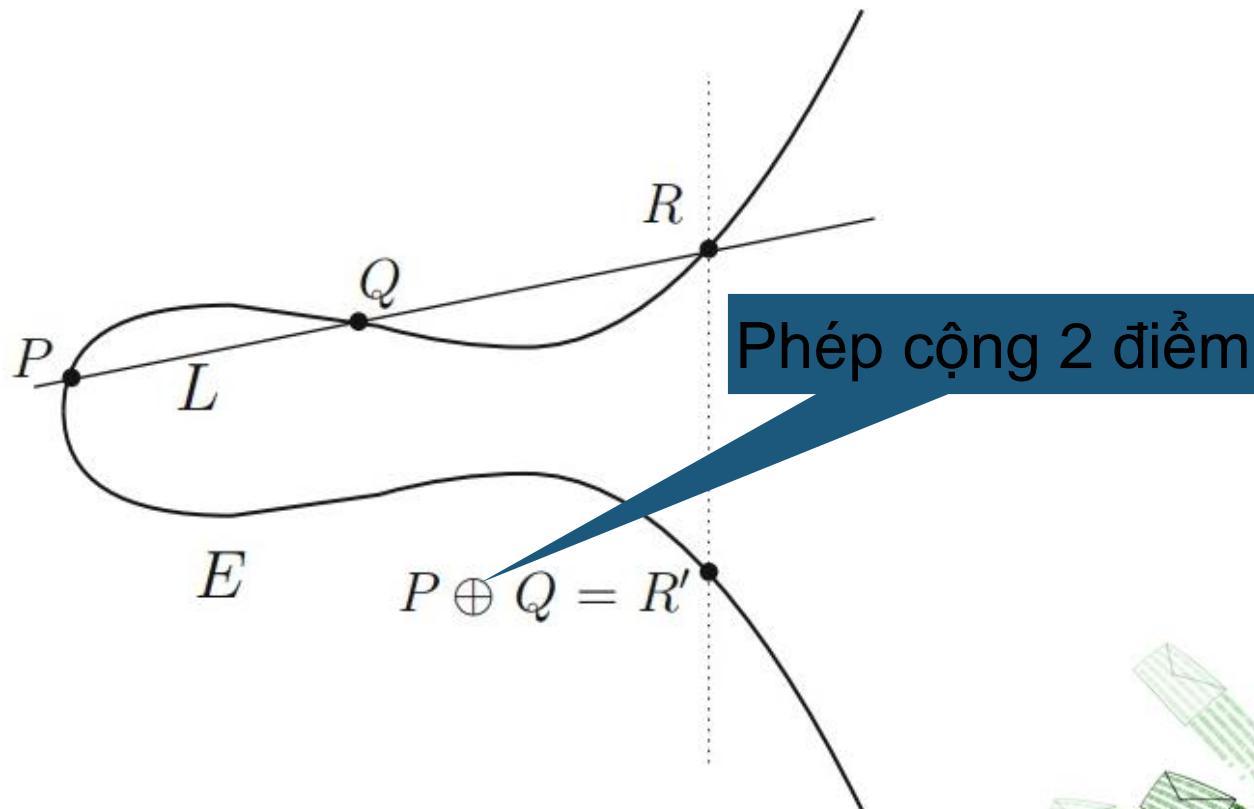
# ĐƯỜNG CONG ELIPTIC

# Đường Cong Ecliptic

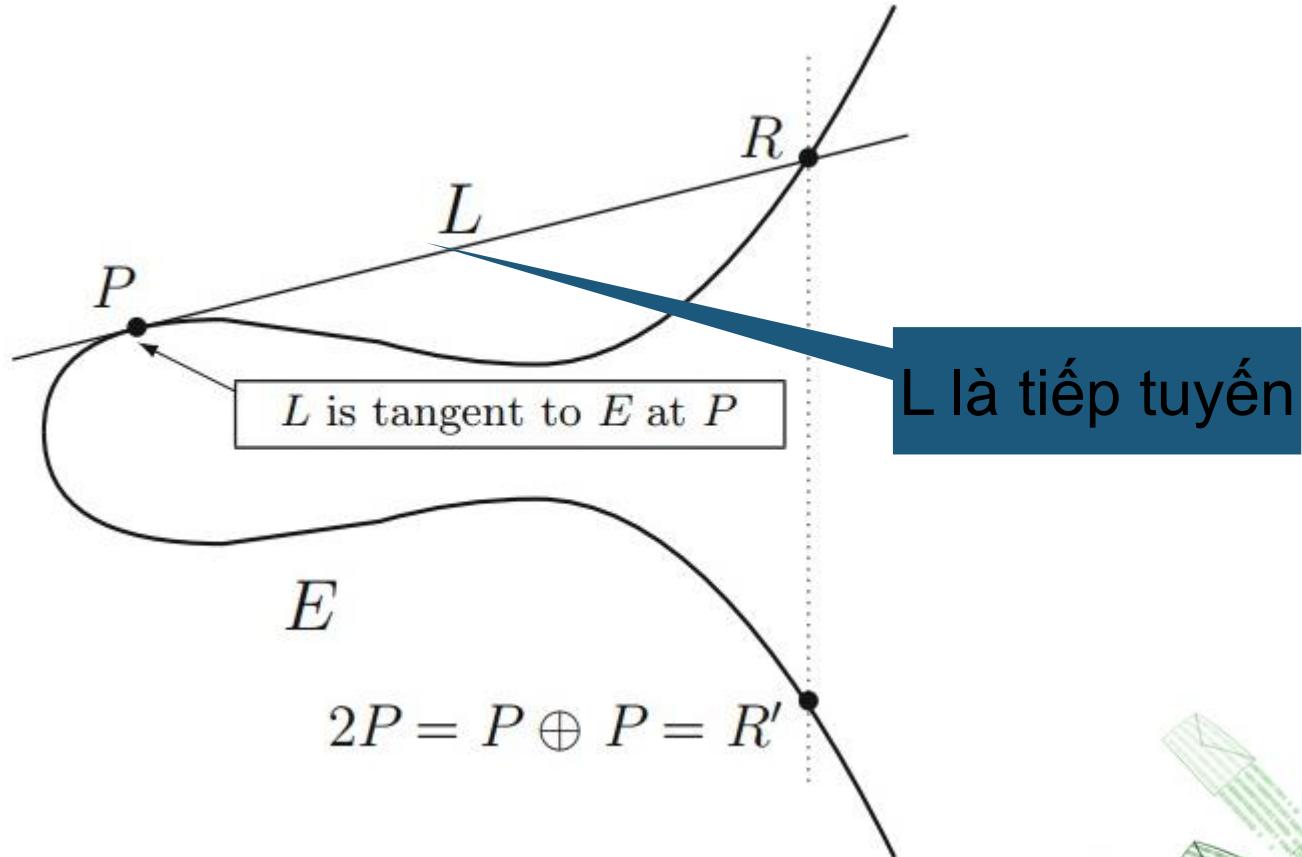
- Năm 1985, thuật toán mã hóa khóa công khai mới được đề xuất dựa trên đường cong Elliptic.
- Một đường cong Elliptic là tập hợp các điểm thỏa mãn một phương trình toán học cụ thể.
- Các phương trình cho một đường cong Elliptic trông giống như sau:

$$y^2 = x^3 + ax + b$$

# Đường Cong Ecliptic



# Đường Cong Ecliptic

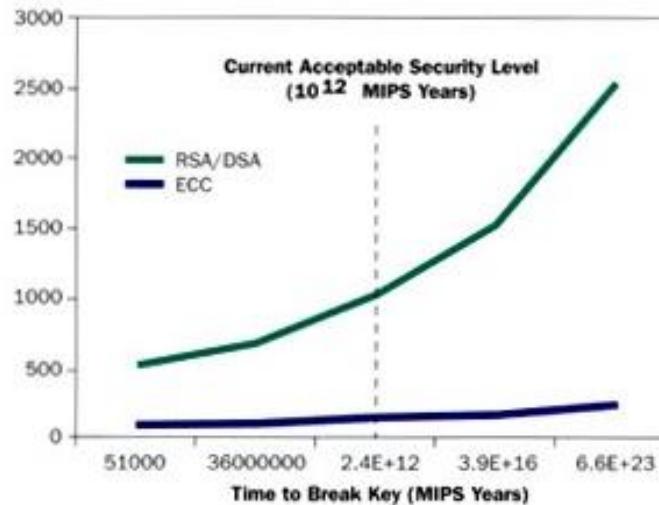


# Đường Cong Ecliptic

- Tính toán cộng 2 điểm bất kì trên đường cong sẽ cho ra một điểm khác thuộc đường cong đó
- Nếu có hai điểm thuộc đường cong với một điểm cộng với nó n lần ra điểm còn lại, thì việc tìm ra n khi mà chỉ biết điểm đầu và điểm cuối là rất khó
- Một hệ thống ECC bao gồm:
  - nguyên tố làm giới hạn
  - một phương trình đường cong và một điểm trên đường cong đó.
  - Một khóa riêng là một số *priv*
  - Một khóa công khai là kết quả của việc cộng điểm ban đầu với chính nó *priv* lần
  - Tính toán các khóa riêng từ khóa công khai trong hệ thống mã hóa này được gọi là logarit rác trên đường cong Elliptic – Elliptic Curve Discrete Logarithm Problem (ECDLP)

# Đường Cong Ecliptic

NIST guidelines for public key sizes for AES			
ECC KEY SIZE (Bits)	RSA KEY SIZE (Bits)	KEY SIZE RATIO	AES KEY SIZE (Bits)
163	1024	1 : 6	
256	3072	1 : 12	128
384	7680	1 : 20	192
512	15 360	1 : 30	256



# Đường Cong Ecliptic

ECC hiện đang được sử dụng trong một loạt các ứng dụng:

- Chính phủ Mỹ sử dụng để bảo vệ thông tin liên lạc nội bộ
- Các dự án Tor sử dụng để giúp đảm bảo ẩn danh
- Bitcoins
- Cung cấp chữ ký số trong dịch vụ iMessage của Apple
- Để mã hóa thông tin DNS với DNSCurve
- SSL/TLS
- ECC đang nhanh chóng trở thành giải pháp thay thế cho RSA.

# Mã Hóa Ứng dụng-Chương 5



Đoàn Trình Dục/ Đoàn Trình Dục

Mail: duc.duantrinh@stu.edu.vn

2019



# NỘI DUNG CHÍNH



Giới Thiệu Hàm Băm



Công Dụng Hàm Băm



Tính Chất Hàm Băm



Cấu Trúc Hàm Băm

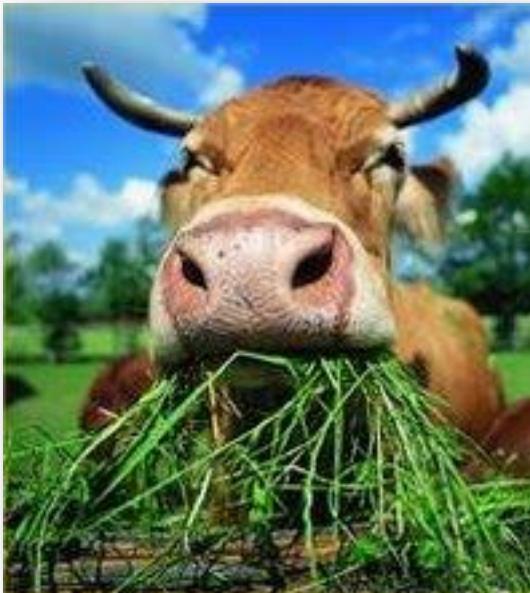


Tấn Công Hàm Băm

# GIỚI THIỆU HÀM BĂM

# Giới Thiệu

- Là giải thuật nhằm sinh ra các **giá trị băm** tương ứng với mỗi **khối dữ liệu**.
- **Giá trị băm** đóng vai gần như một **khóa** để phân biệt các **khối dữ liệu**, tuy nhiên, người ta chấp **hiện tượng trùng khóa**
- Hàm Băm là chức năng hàm một chiều:



# Hashing và Encryption

Hello, world.  
A sample sentence to  
show encryption.

k

E

NhbXBsZSBzZW50ZW5jZS  
B0byBzaG93IEVuY3J5cHR  
pb24KsZSBzZ

Hello, world.  
A sample sentence to  
show encryption.

D

k

NhbXBsZSBzZW50ZW5jZS  
B0byBzaG93IEVuY3J5cHR  
pb24KsZSBzZ

Encryption là hàm 2 chiều

This is a clear text that  
can easily read without  
using the key. The  
sentence is longer than  
the text above.

h

52f21cf7c7034a20  
17a21e17e061a863

Hashing là hàm một chiều( one way)

# Động Lực Hashing

- Hạn chế của hàm kiểm tra CheckSum
- Thuật toán kiểm tra sự toàn vẹn dữ liệu
- Xây dựng một thuật toán mà giá trị gốc ban đầu không thể suy ra từ mã kiểm tra
- Một sự thay đổi bất kỳ của giá trị đầu vào sẽ sinh ra giá trị đầu ra khác

# ỨNG DỤNG CỦA HASHING

# Fingerprint

- Kiểm tra tính toàn vẹn của file
- Kiểm tra tính toàn vẹn của public key
- Lưu trữ password
- Chữ ký số

Windows 7 Home Premium with Service Pack 1 (x86) - DVD (English)

ISO | English | Release Date: 12/5/2011 | Details

This media refresh includes the installation hotfix described in [KB Article 2534111](#). No other product.

File Name: en\_windows\_7\_home\_premium\_with\_sp1\_x86\_dvd\_u\_676701.iso

Languages: English

SHA1: 6071B4553FCF0EA53D589A846B5AE76743DD68FC

Permalinks: [File](#) [Download](#)

Command Line SFTP Client

The authenticity of host '192.168.100.101 (192.168.100.101)' can't be established.

RSA key fingerprint is da:47:93:b4:3a:90:5b:50:1f:20:a8:f9:b7:a1:d0:e1.

Are you sure you want to continue connecting (yes/no)?

GUI-based SFTP Client

Verify Host key

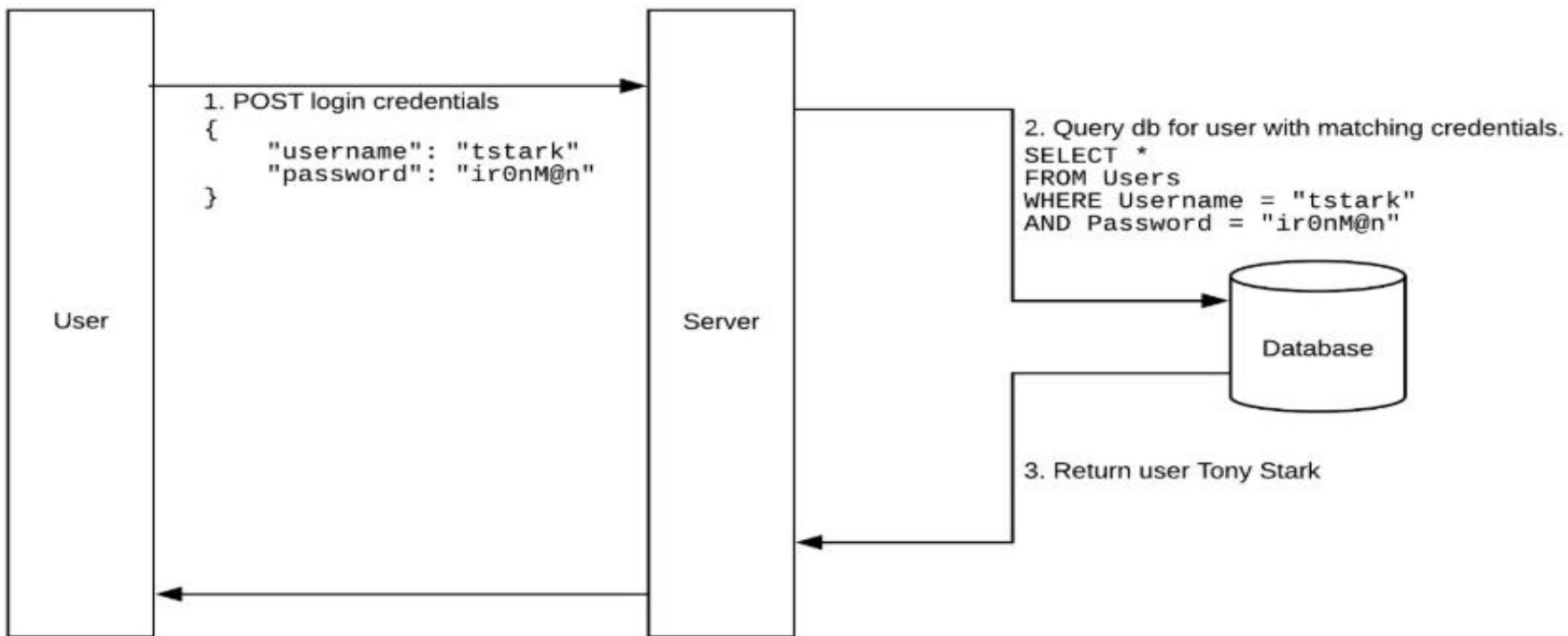
New host key

Connecting to 127.0.0.1

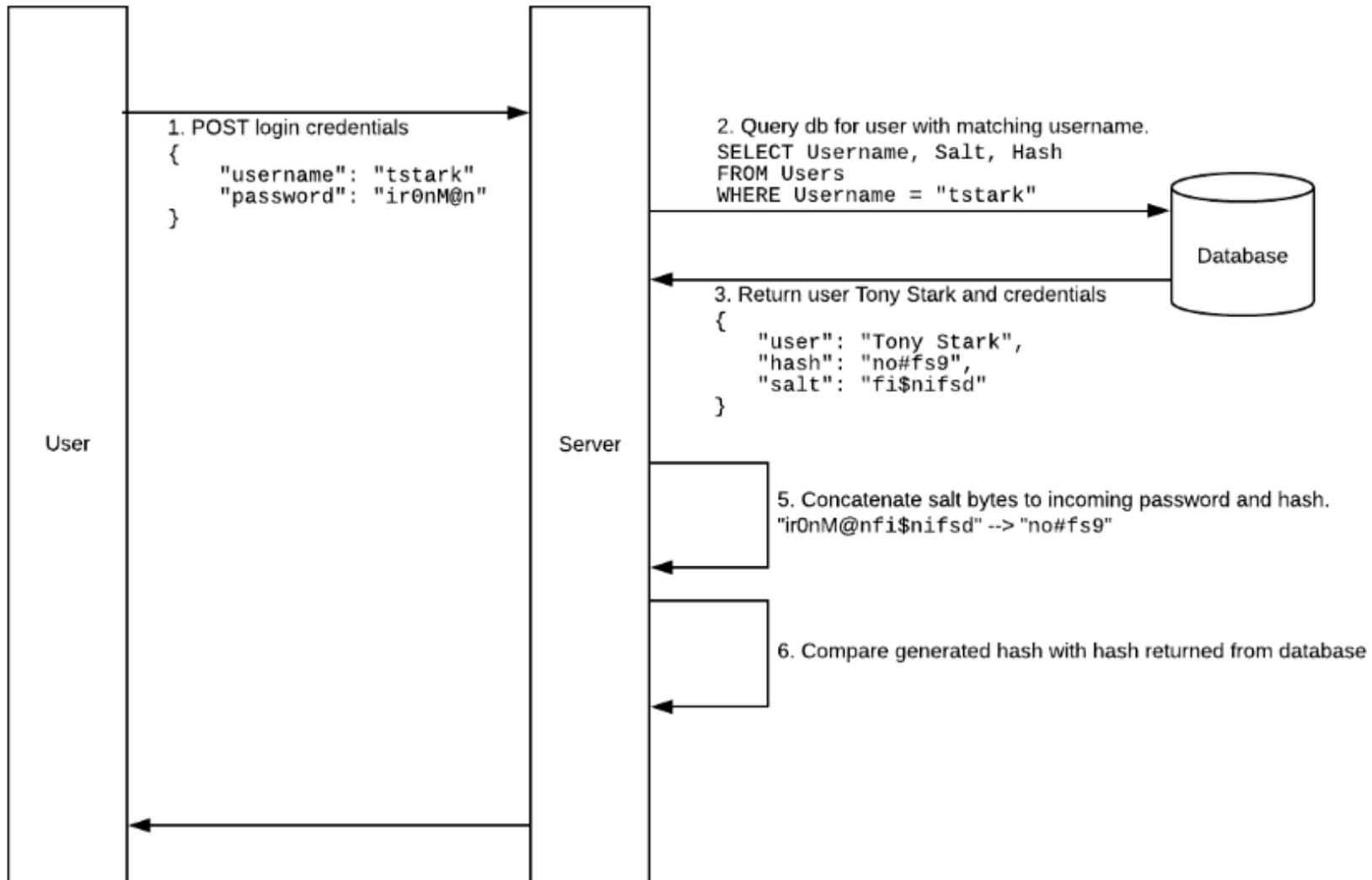
Fingerprint: da:47:93:b4:3a:90:5b:50:1f:20:a8:f9:b7:a1:d0:e1

Accept and Save Accept

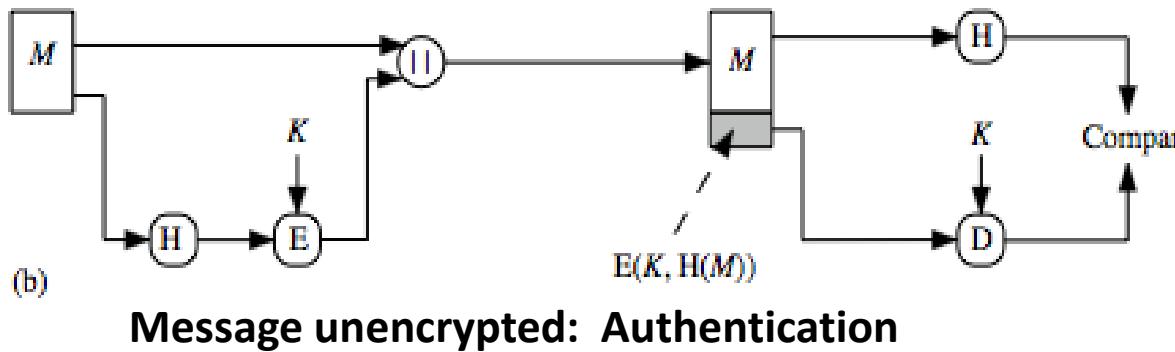
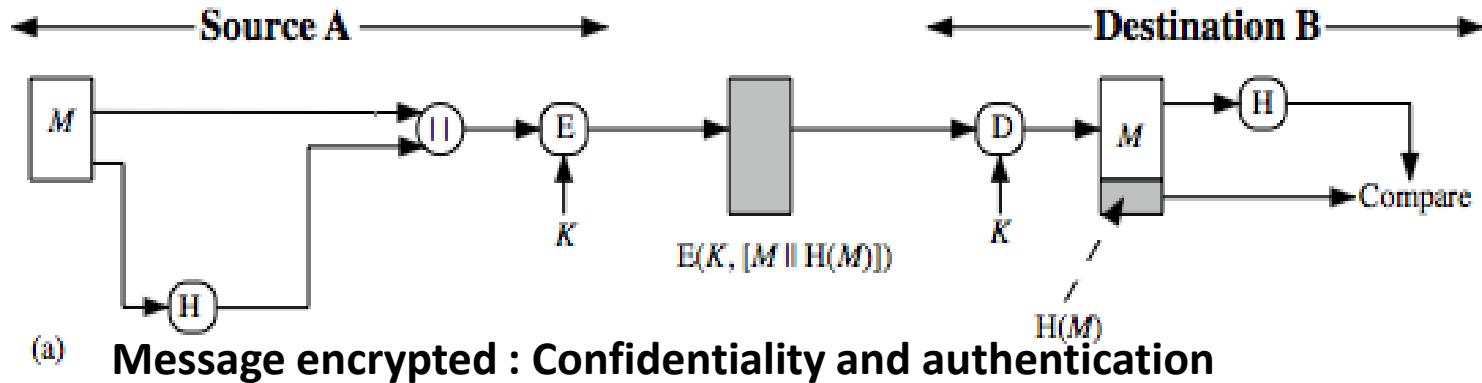
# Password storage (one-way encryption)



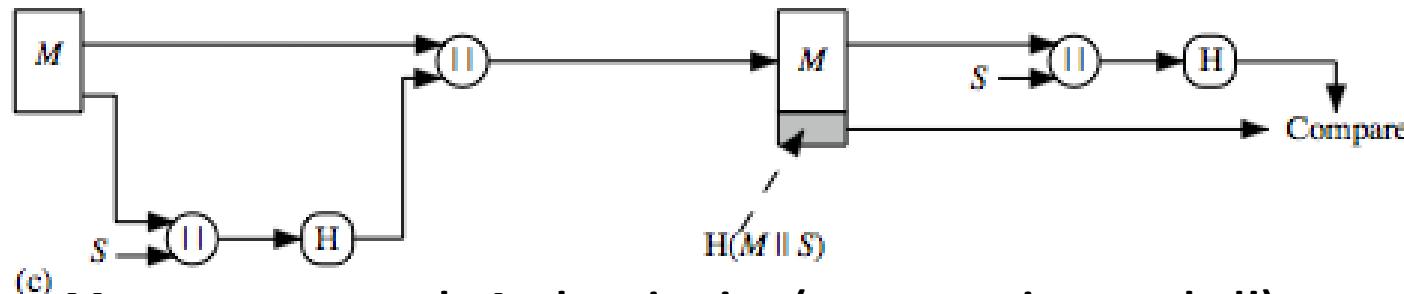
# Password storage (one-way encryption)



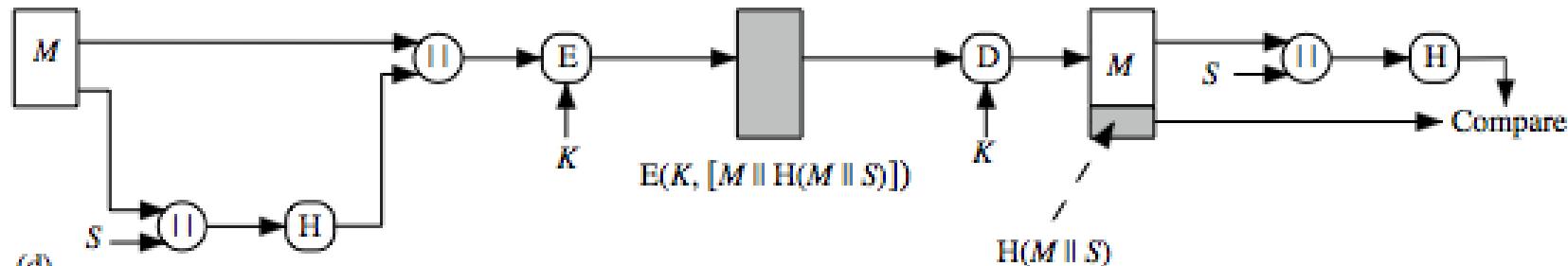
## Hash Function Usages (I)



## Hash Function Usages (I)

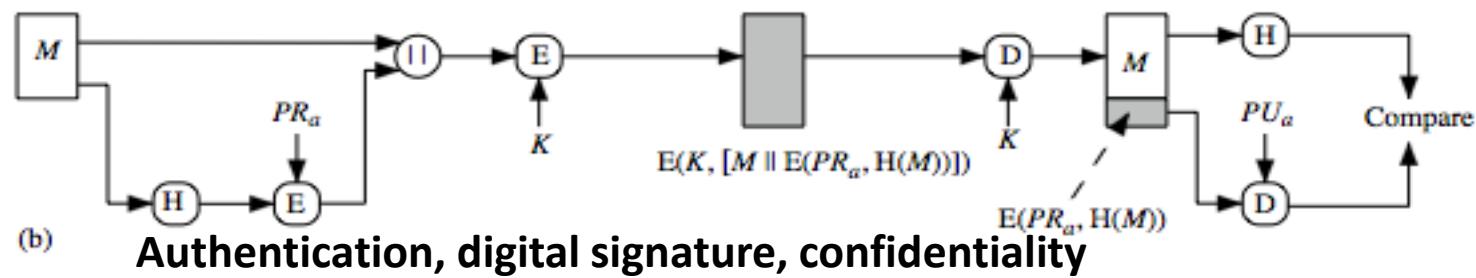
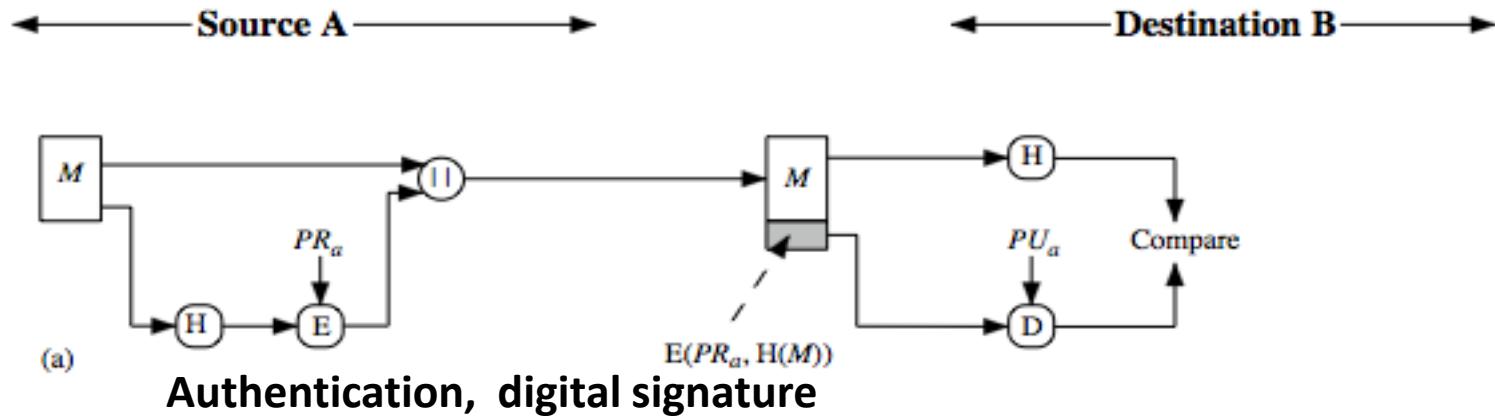


**Message encrypted : Authentication (no encryption needed!)**



**Message unencrypted: Authentication, confidentiality**

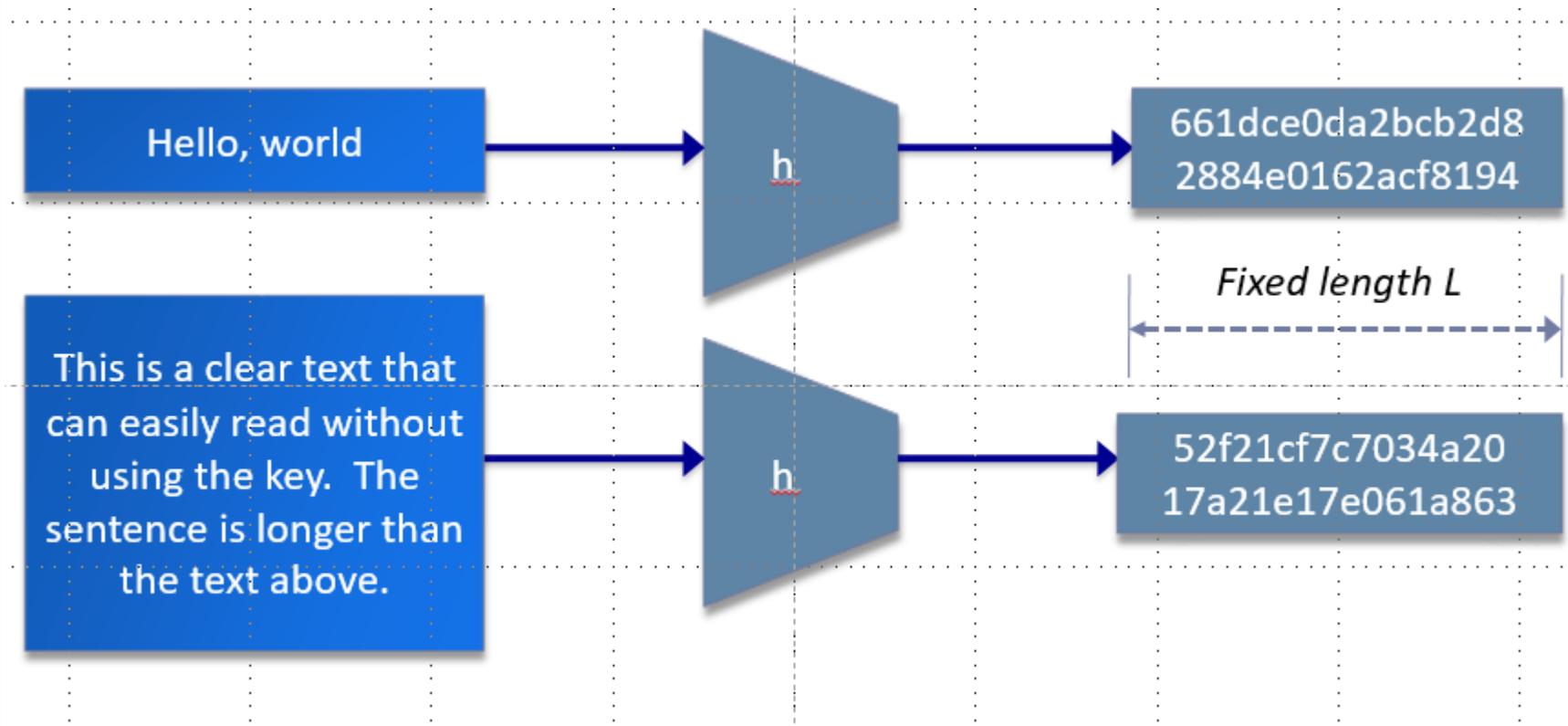
# Hash Function Usages (I)



# TÍNH CHẤT HASHING

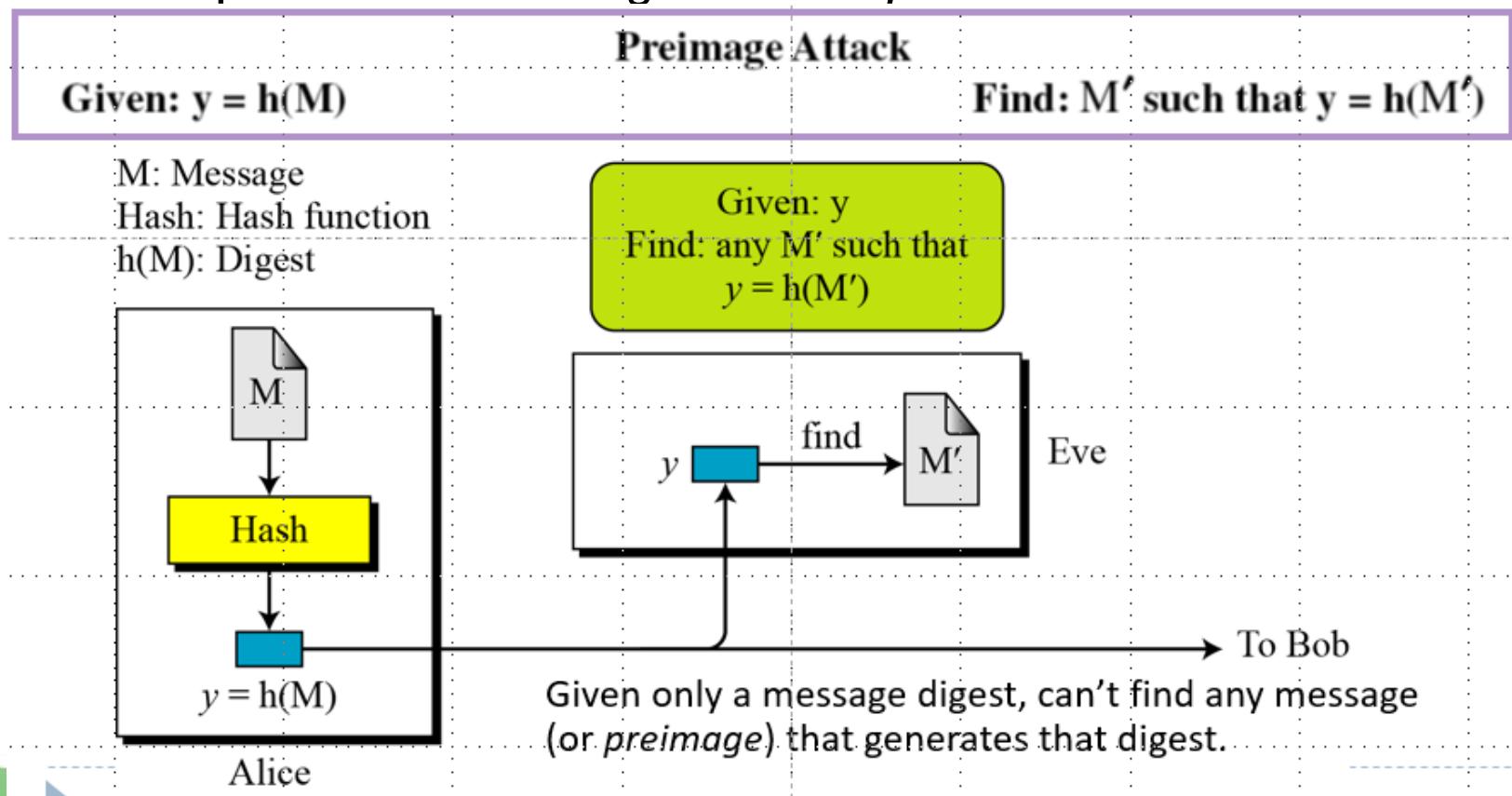
## Properties : Fixed length

Hash chuyển text có độ dài tùy ý thành text có độ dài cố định

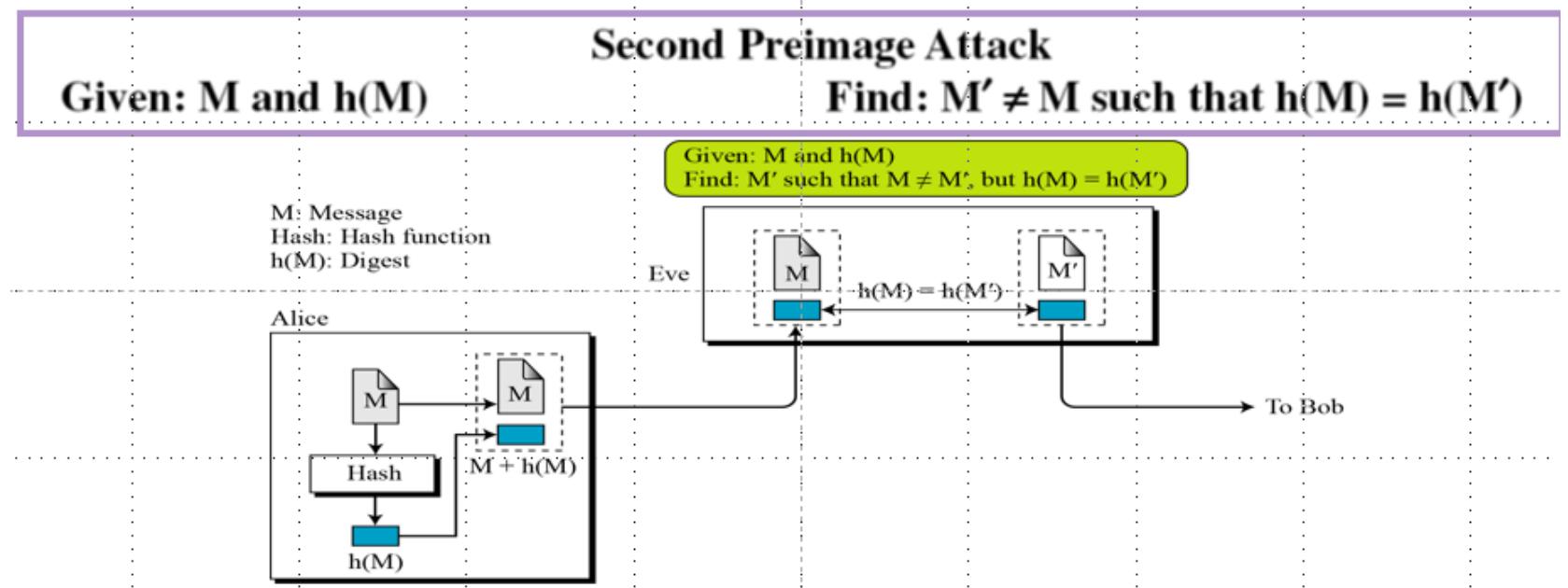


## Properties : Preimage resistant

- Hàm hash phải đảm bảo rất khó để xác định giá trị ban đầu khi biết mã hash
- Nói một cách bình thường là *Hàm một chiều*



# Properties : Second Preimage resistant



# Properties : Collision resistant

Given: none

Collision Attack

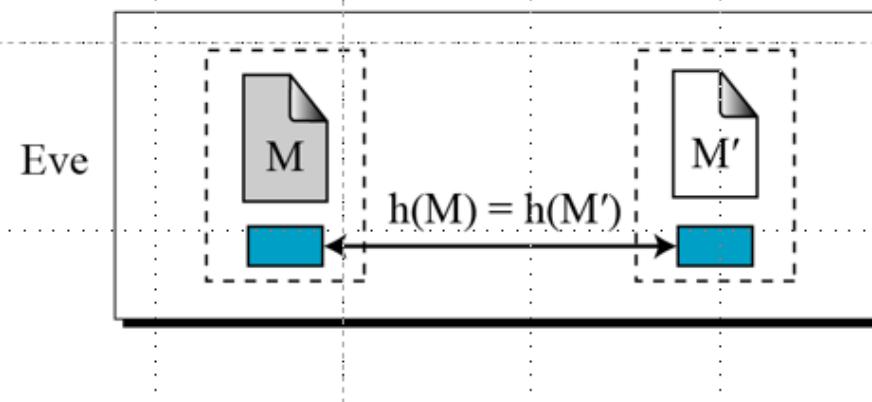
Find:  $M' \neq M$  such that  $h(M) = h(M')$

M: Message

Hash: Hash function

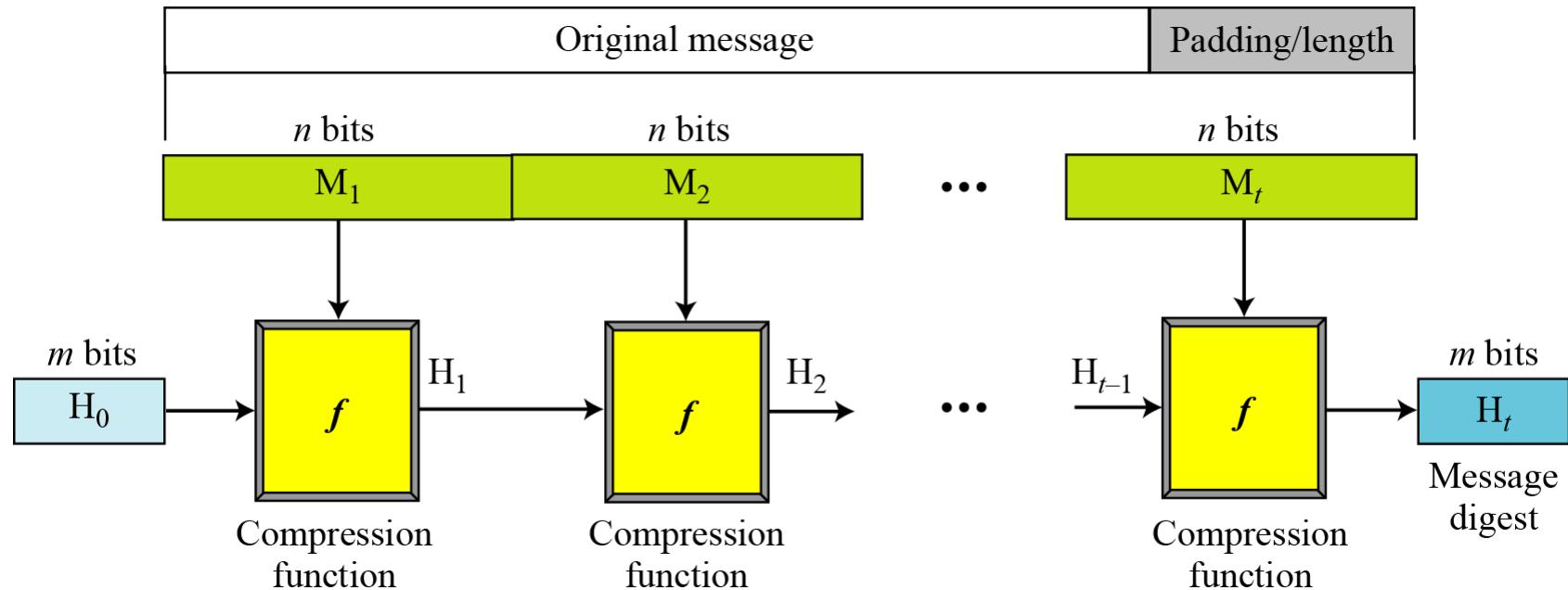
$h(M)$ : Digest

Find:  $M$  and  $M'$  such that  $M \neq M'$ , but  $h(M) = h(M')$



# HASH FUNCTION STRUCTURE

# Merkle-Damgard Scheme



# Hash Functions Family

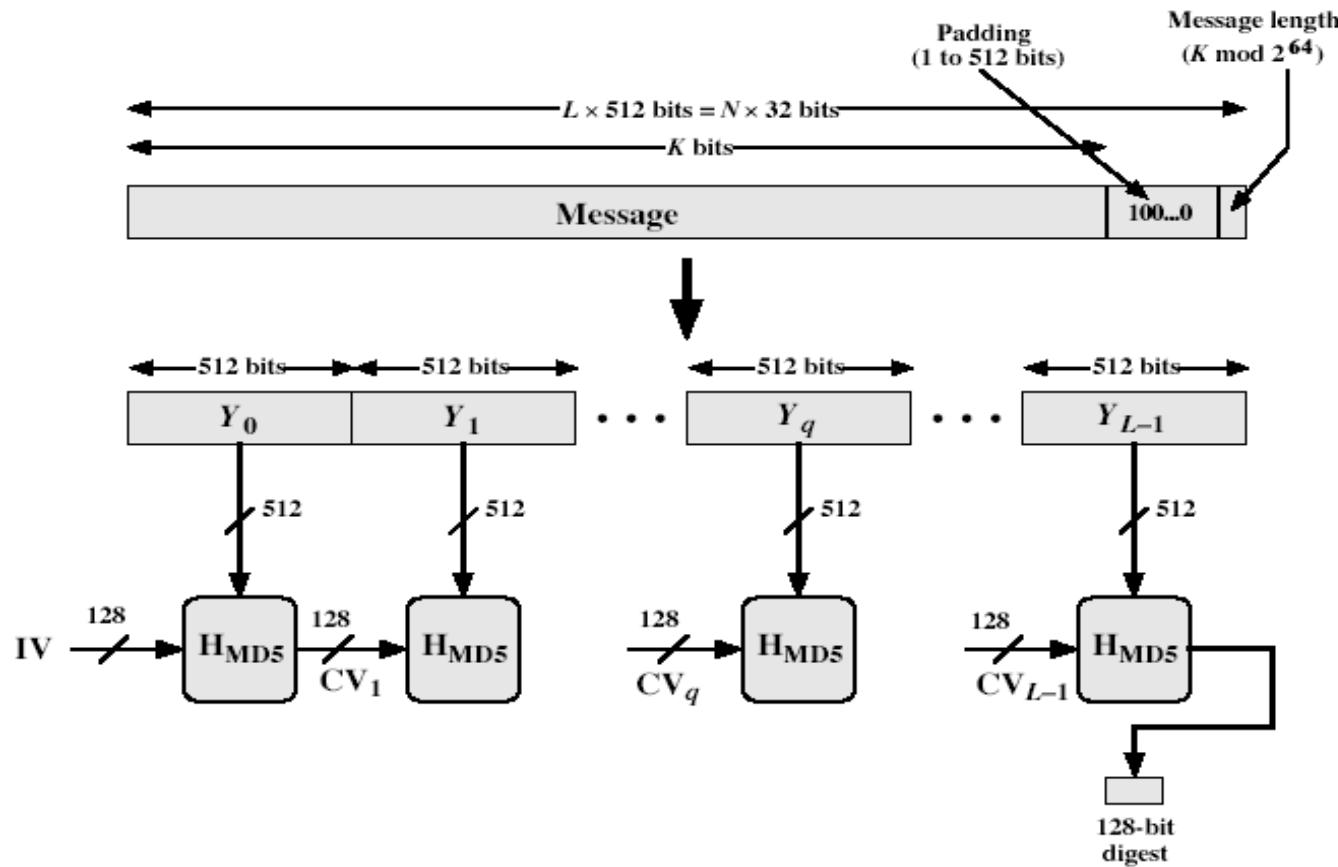
- **MD (Message Digest)**
  - Designed by Ron Rivest
  - Family: MD2, MD4, MD5
- **SHA (Secure Hash Algorithm)**
  - Designed by NIST
  - Family: SHA-0, SHA-1, and SHA-2
    - SHA-2: SHA-224, SHA-256, SHA-384, SHA-512
    - SHA-3: New standard in competition
- **RIPEMD (Race Integrity Primitive Evaluation Message Digest)**
  - Developed by Katholieke University Leuven Team
  - Family : RIPEMD-128, RIPEMD-160, RIPEMD-256, RIPEMD-320

# Hash Functions Family

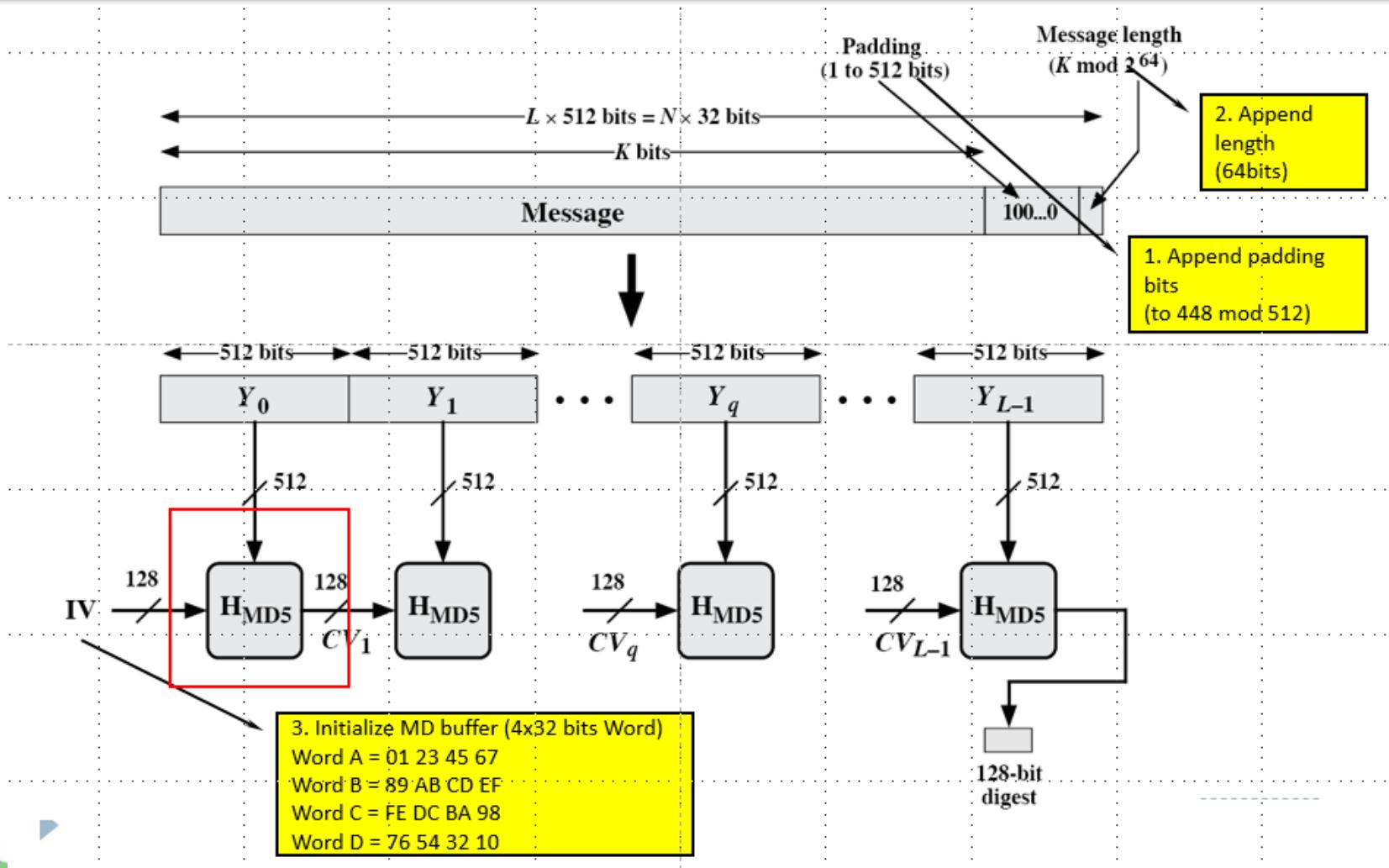
	MD5	SHA-1	RIPEMD-160
Digest length	128 bits	160 bits	160 bits
Basic unit of processing	512 bits	512 bits	512 bits
Number of steps	64 (4 rounds of 16)	80 (4 rounds of 20)	160 (5 paired rounds of 16)
Maximum message size	$\infty$	$2^{64} - 1$ bits	$2^{64} - 1$ bits
Primitive logical functions	4	4	5
Additive constants used	64	4	9
Endianness	Little-endian	Big-endian	Little-endian

# MD5

# MD5



## MD5



## MD5 Security

- ❖ Một hàm băm được coi là bảo mật khi chống được các cuộc tấn công
  - Preimage resistant
  - Second Preimage resistant
  - Collision resistant
- ❖ Vào năm 2011 RFC 615 đã khuyến cáo về cuộc tấn công Collision resistant vào md5 trong 10 s đối với máy tính 2.66 GHz Pentium 4 system
- ❖ Md5 không bảo mật

# SHA

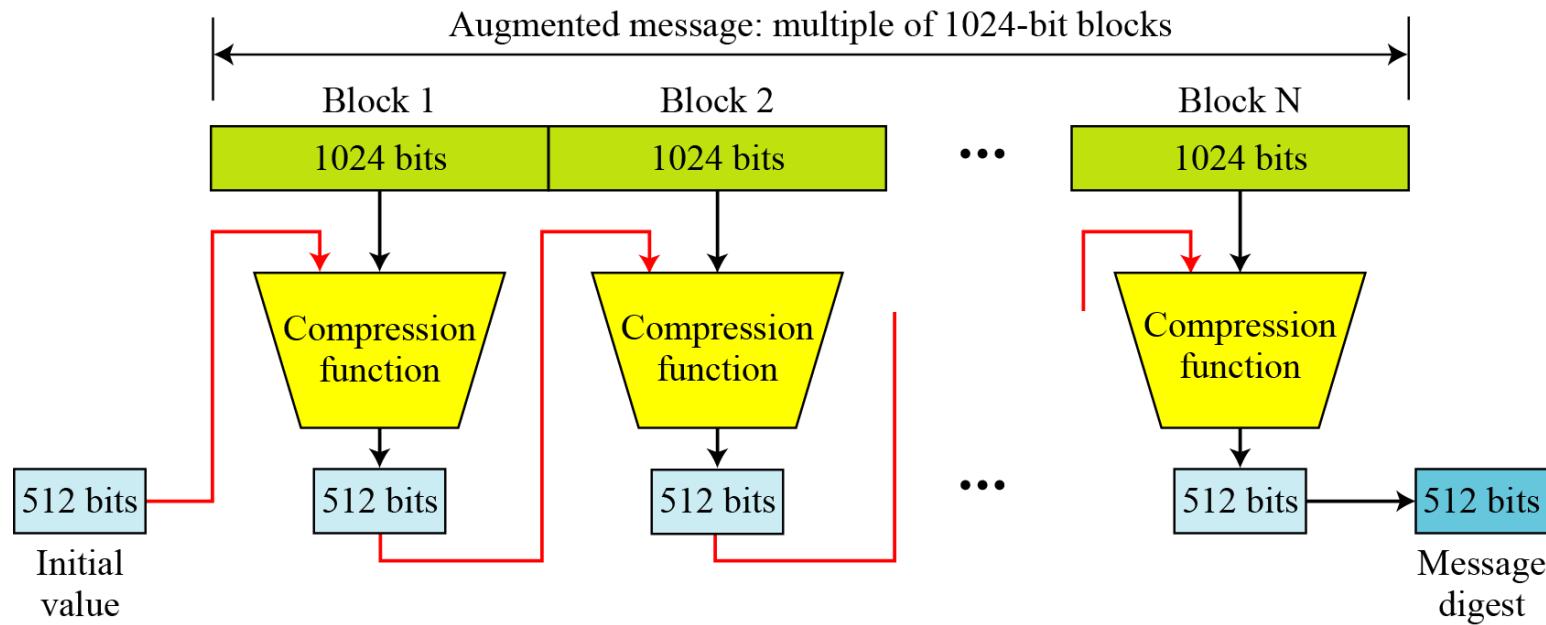
## SHA

- SHA ban đầu được thiết kế bởi NIST & NSA vào năm 1993 và được sửa đổi vào năm 1995 với tên SHA-1
- Sử dụng trong sơ đồ chữ ký DSA
- Tiêu chuẩn RFC3174
- Dựa trên thiết kế của MD4 với sự khác biệt chính tạo ra các giá trị băm 160 bit
- SHA -1 đã bị phá vỡ vào năm 2005

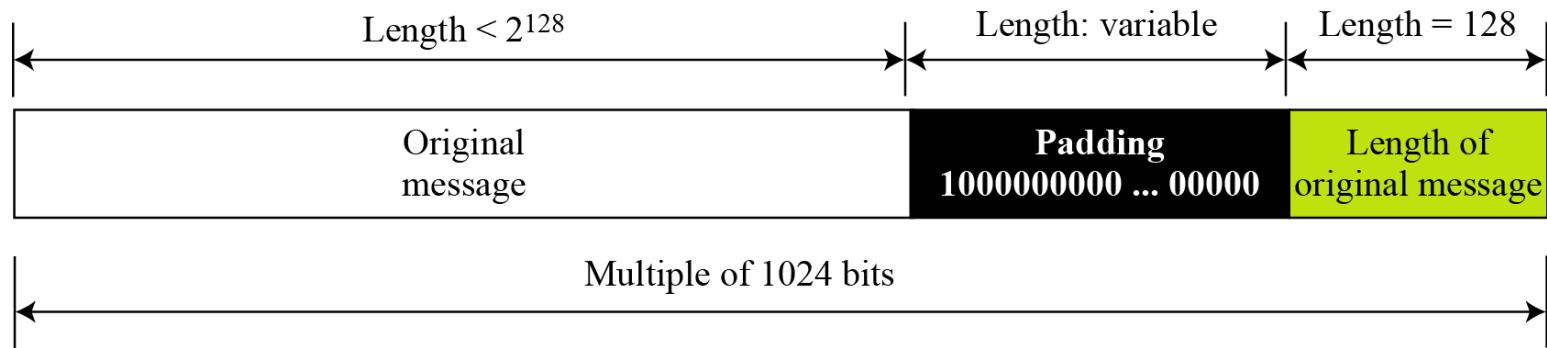
# SHA

	MD5	SHA-0	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Digest size	128	160	160	224	256	384	512
Message size	$2^{64}-1$	$2^{64}-1$	$2^{64}-1$	$2^{64}-1$	$2^{64}-1$	$2^{128}-1$	$2^{128}-1$
Block size	512	512	512	512	512	1024	1024
Word size	32	32	32	32	32	64	64
# of steps	64	64	80	64	64	80	80

# SHA



# SHA Padding Length

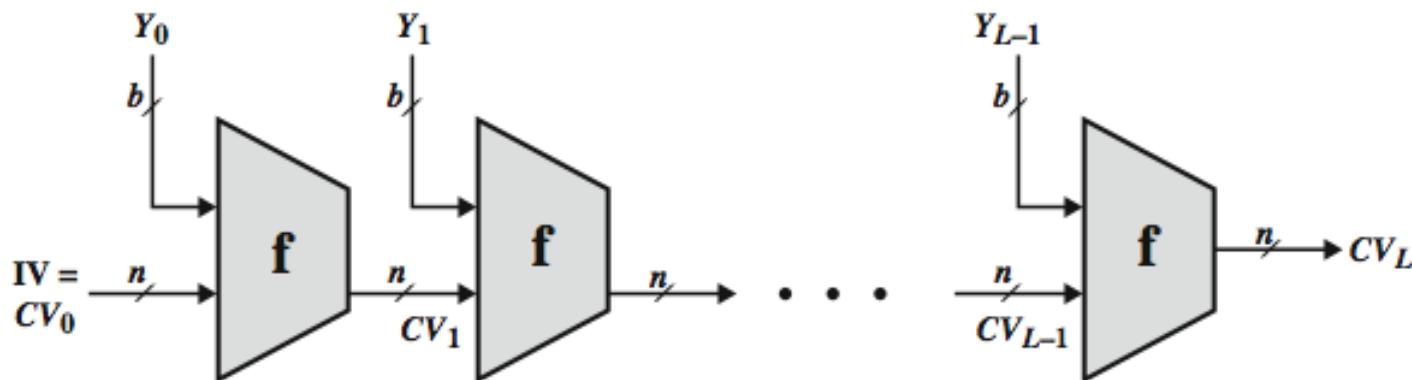


## SHA Version

- ❖ Năm thuật giải SHA bao gồm:
  - SHA-1 (trả lại kết quả dài 160 bit)
  - SHA-224 (trả lại kết quả dài 224 bit)
  - SHA-256 (trả lại kết quả dài 256 bit)
  - SHA-384 (trả lại kết quả dài 384 bit)
  - SHA-512 (trả lại kết quả dài 512 bit)
- ❖ SHA-1 được sử dụng rộng rãi trong nhiều ứng dụng và giao thức an ninh khác nhau, bao gồm TLS&SSL, PGF, SSH và IPSec
- ❖ SHA-1 không còn được coi là an toàn
- ❖ SHA-2 vẫn chưa bị phá vỡ
- ❖ SHA-3 đang phát triển

## Hash Function Cryptanalysis

- Tấn công dựa vào thuộc tính thuật toán tốt hơn là tìm kiếm toàn diện
- Hàm băm lặp lại trong các block message
- Tấn công thường nhắm vào sự đụng độ của hàm  $f$



# Mã Hóa Ứng dụng-Chương 6



Đoàn Trình Dục/ Đoàn Trình Dục

Mail: duc.duantrinh@stu.edu.vn

2019



# NỘI DUNG CHÍNH



**Giới thiệu Chứng Thực Dữ Liệu**



**Chứng thực dữ liệu sử dụng mã hóa**



**Chứng thực MAC**



**Chứng thực hàm Băm**



**Chữ kí số**

# GIỚI THIỆU CHỨNG THỰC

## Giới Thiệu

Chứng thực (authentication) hay còn gọi xác thực nhằm giải quyết:

- Xác nhận nguồn gốc của dữ liệu.
- Thuyết phục người sử dụng là dữ liệu này chưa bị sửa đổi hoặc giả mạo.
- Chứng thực dữ liệu là cơ chế quan trọng để duy trì tính toàn vẹn và không thể từ chối của dữ liệu.

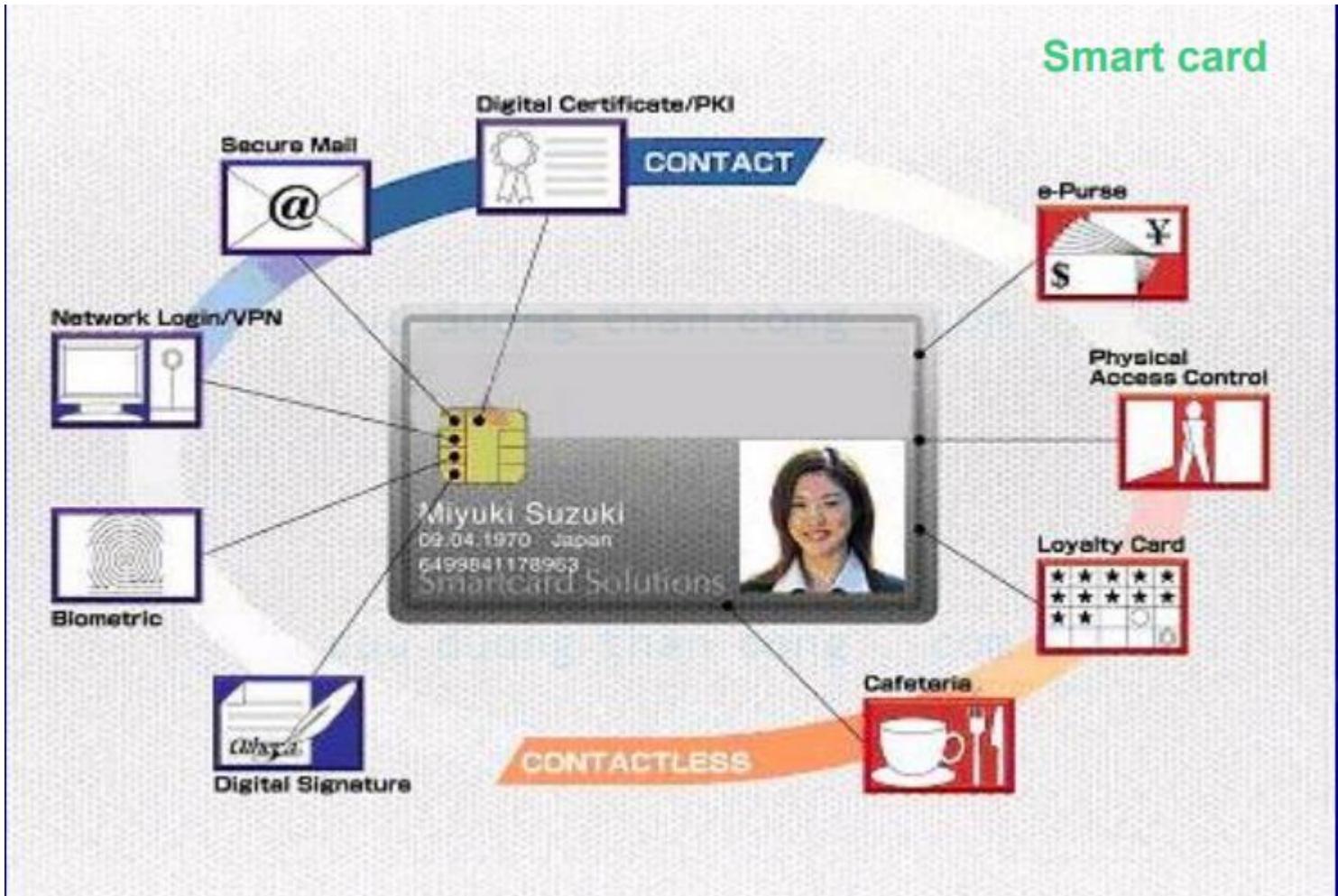
# Chứng Thực Qua Nhận Dạng

Chứng thực thông qua nhận dạng Việc nhận dạng dựa trên một hoặc nhiều yếu tố:

- Password
- PIN
- Smart card
- Biometric: vân tay, võng mạc...
- Chữ ký...



# Chứng Thực Qua Nhận Dạng



# Phương Pháp Chứng Thực

Có 3 phương pháp chứng thực chính sử dụng:

- Mã hoá thông điệp: sử dụng mật mã hoá khoá bí mật hoặc mật mã hoá khoá công khai để mã hoá thông điệp rõ thành mật mã.
- Mã chứng thực thông điệp (MAC – Message Authentication Code): một hàm và một khoá bí mật tạo ra một giá trị có chiều dài cố định sử dụng để chứng thực.
- Hàm băm (Hash Function): một hàm ánh xạ một thông điệp có chiều dài bất kỳ vào một giá trị băm có chiều dài cố định sử dụng để chứng thực.

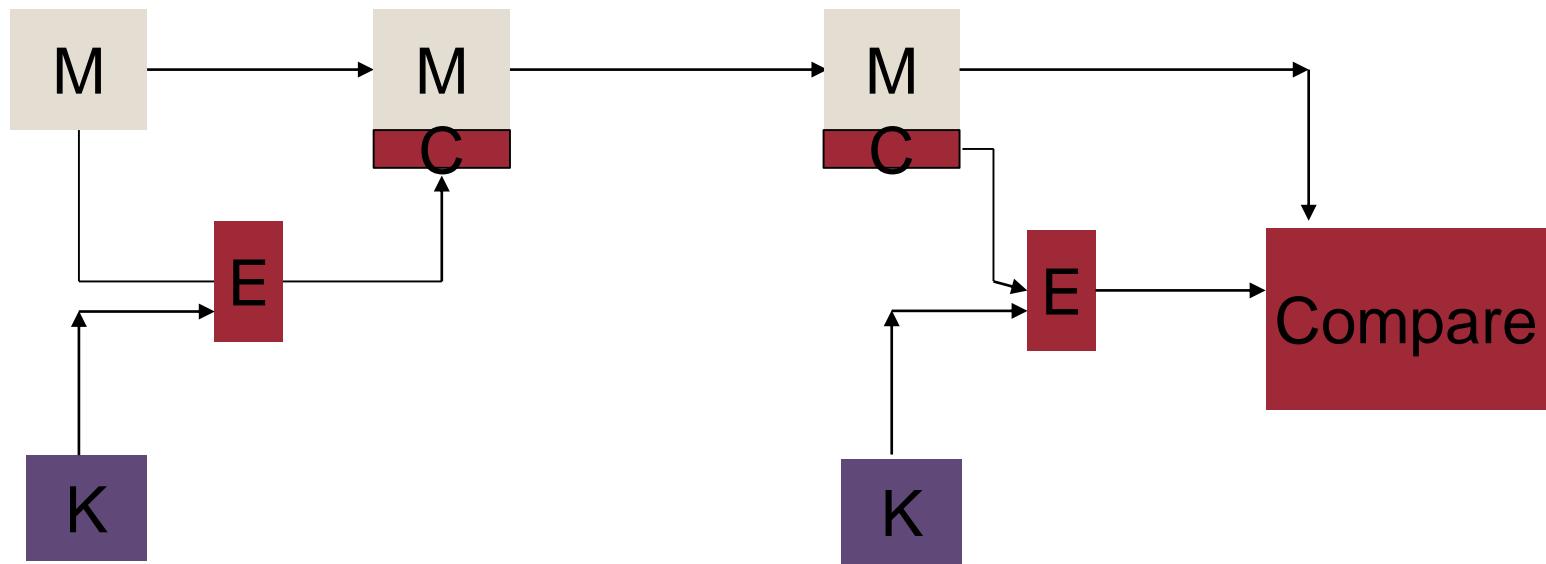
# CHỨNG THỰC SỬ DỤNG MÃ HÓA

# Chứng Thực sử dụng mã hóa

Giả sử Alice và Bob chia sẻ một **khoá bí mật** chung K. Alice muốn gửi một chuỗi dữ liệu M cho Bob và thuyết phục Bob rằng M thực sự đến từ Alice và không bị sửa trong quá trình truyền. Điều này có thể thực hiện như sau:

- Alice gửi M cùng với C cho Bob, với  $C = EK(M)$  và E là một giải thuật mã hoá thông thường đã quy ước trước giữa Alice và Bob.
- Do chỉ có Alice và Bob biết K, Bob có thể sử dụng K để giải mã C thu được  $M'$ .
- Bob sẽ được thuyết phục rằng M thực sự đến từ Alice và M không bị thay đổi trong quá trình truyền nếu và chỉ nếu  $M' = M$ .

# Chứng Thực sử dụng mã hóa



# Chứng Thực sử dụng mã hóa

Tuy nhiên, phương pháp này cho phép Alice có thể từ chối Charlie rằng M xuất phát từ Alice vì M có khả năng xuất phát từ Bob do cùng chia sẻ khoá bí mật K.

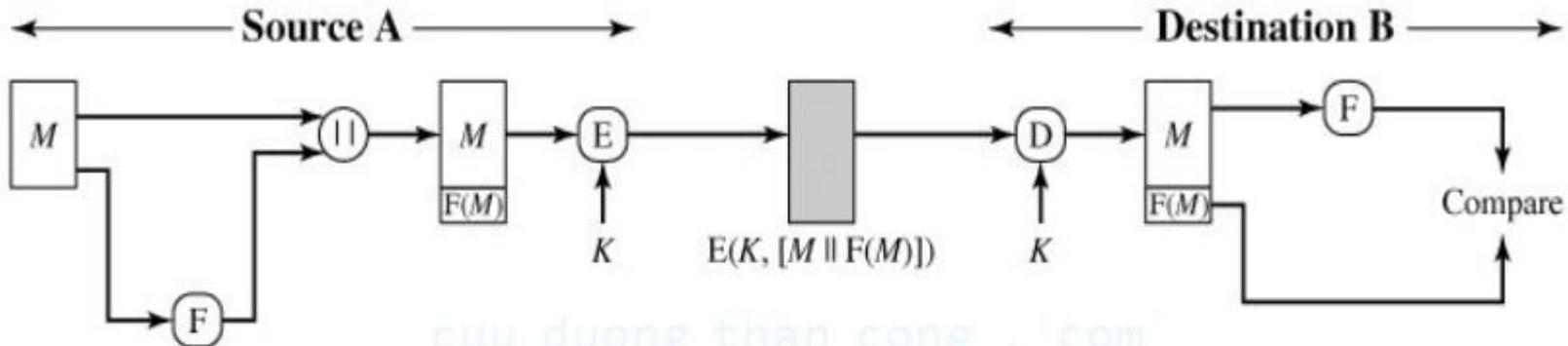
- Nhược điểm này được giải quyết bằng mật mã hoá khoá công khai.
- Nếu chuỗi M ngắn, có thể mã hóa M trực tiếp để xác nhận nó.
- Nếu chuỗi M dài, chỉ cần tính toán một h ngắn đại diện cho M và mã hóa h.

# Chứng Thực sử dụng mã hóa

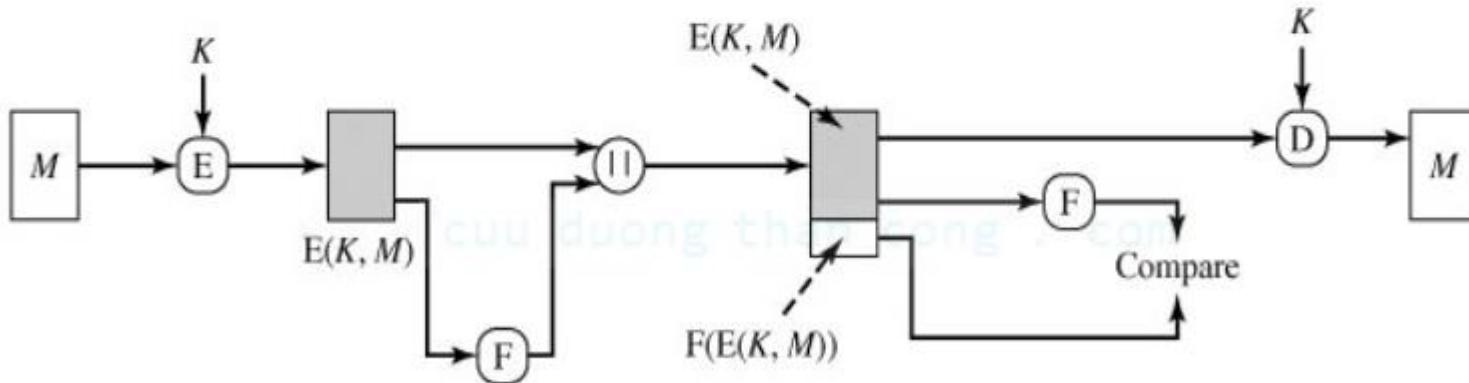
Gọi H là thông điệp đi kèm để kiểm tra tính toàn vẹn dữ liệu ta có những định nghĩa sau:

- H được tạo ra mà không sử dụng khoá bí mật được gọi là digital digest hoặc digital fingerprint (dấu vân tay kỹ thuật số), có thể thu được từ một hàm băm (Hash Function).
- H được tạo ra bằng cách sử dụng một khoá bí mật được gọi là một mã xác thực thông điệp (MAC –Message Authentication Code).
- H cũng có thể thu được bằng cách sử dụng giải thuật checksum. Kết hợp một hàm băm và giải thuật checksum để tạo ra một mã xác thực tin nhắn keyed-hash (HMAC, Keyed-Hash Message Authentication Code).

# Quá trình kiểm lỗi



(a) Internal error control

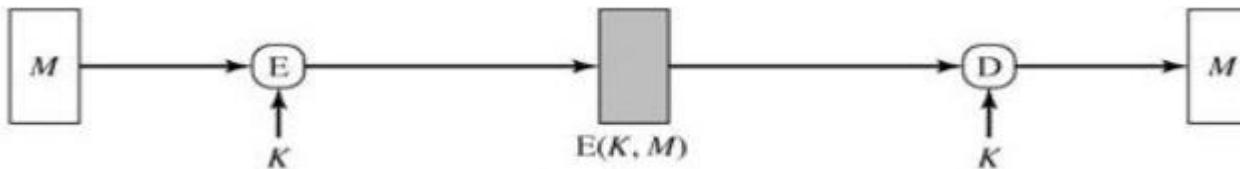


(b) External error control

# Công dụng của mã hóa

**Mã hóa khoá đối xứng (khoá bí mật):  $A \rightarrow B: E(K, M)$**

- Bảo mật: chỉ A và B chia sẻ K
- Chứng thực:
  - Có thể đến chỉ từ A
  - Không thay đổi trong quá trình truyền
  - Yêu cầu một số định dạng và dự phòng
- Không cung cấp chữ ký
  - Người nhận có thể giả mạo thông điệp
  - Người gửi có thể phủ nhận đã gửi thông điệp



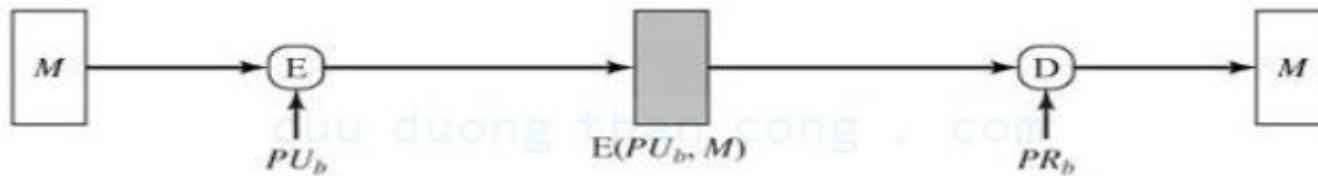
(a) Symmetric encryption: confidentiality and authentication

# Công dụng của mã hóa

## Mã hóa khoá bất đối xứng (khoá công khai)

$A \rightarrow B: E(PUb, M)$

- Bảo mật
  - Chỉ B có PR<sub>b</sub> giải mã
- Không cung cấp chứng thực
  - Bất cứ ai cũng có thể sử dụng PUb để mã hóa thông điệp và tự xưng là A.



(b) Public-key encryption: confidentiality

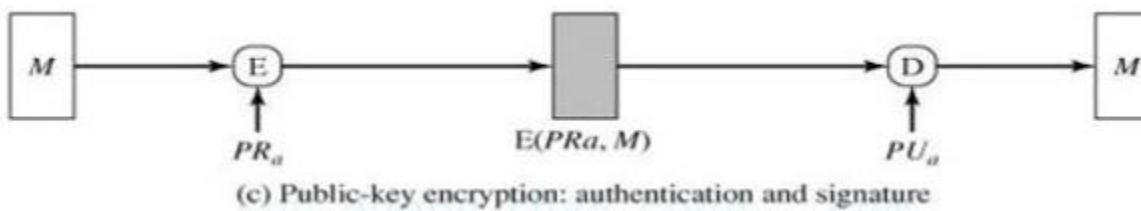
# Công dụng của mã hóa

**Mã hóa khoá công khai: chứng thực và chữ ký số**

**A → B: E(PR<sub>a</sub>, M)**

Cung cấp chứng thực và chữ ký số

- Chỉ A có PR<sub>b</sub> để mã hóa
- Không bị thay đổi trong quá trình truyền
- Yêu cầu một số định dạng và dự phòng
- Bất kỳ ai cũng có thể sử dụng PUA để xác minh chữ ký

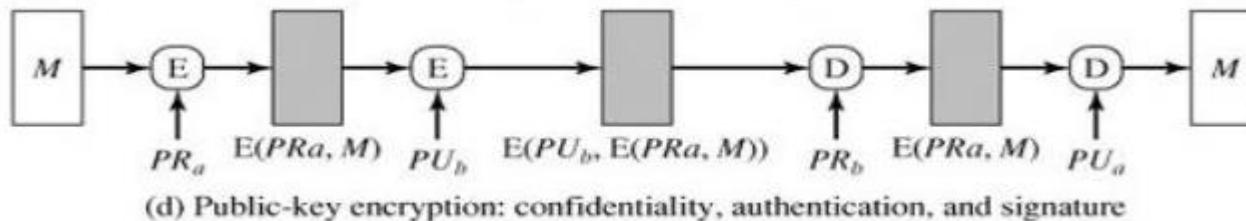


# Công dụng của mã hóa

Mã hóa khoá công khai: bảo mật, chứng thực, và chữ ký số

A → B:  $E(PUb, E(PR_a, M))$

- Cung cấp bảo mật nhờ PUb.
- Cung cấp chứng thực và chữ ký số nhờ PRa.

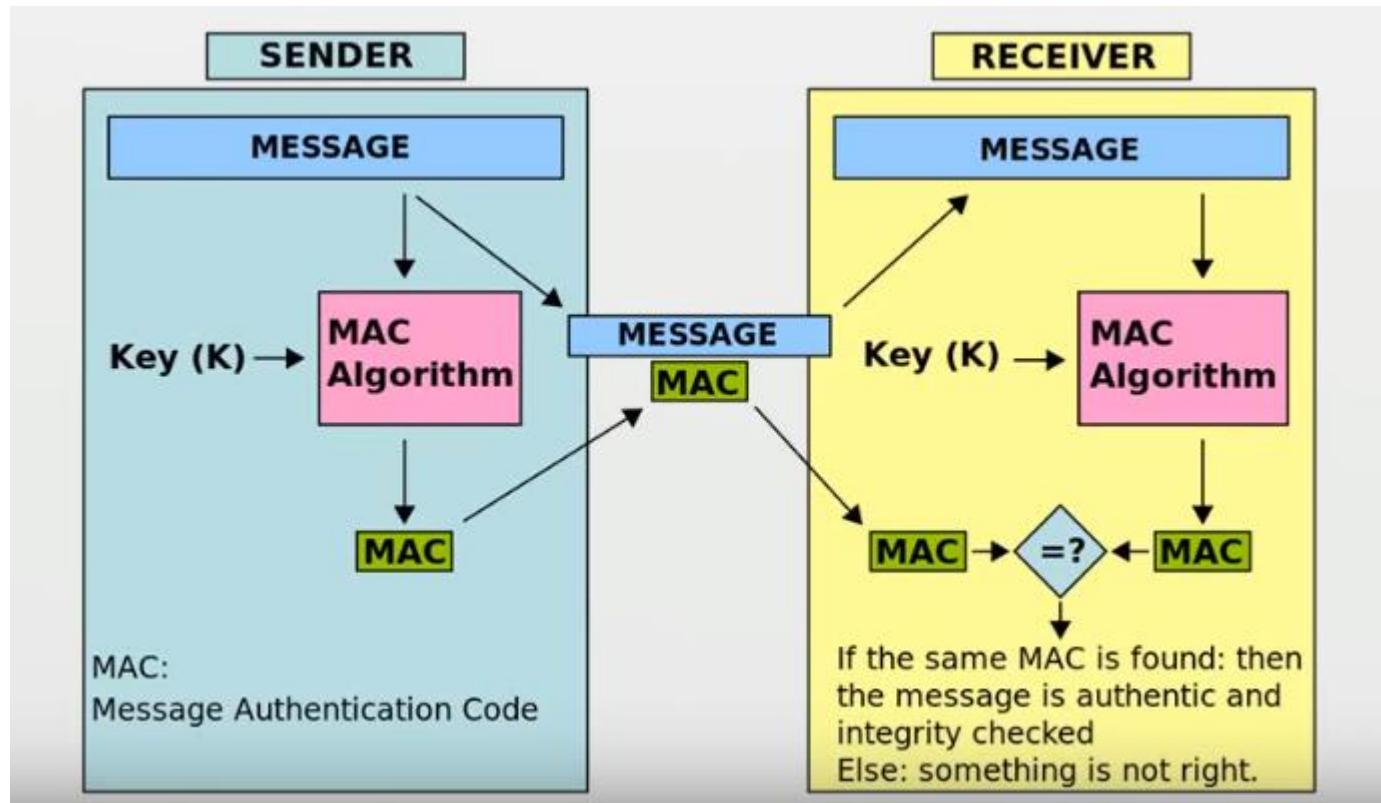


# **CHỨNG THỰC THÔNG ĐIỆP MAC**

# Mã chứng thực thông điệp

- Là một kỹ thuật chứng thực liên quan đến việc sử dụng một khoá bí mật để tạo ra một khối dữ liệu có kích thước nhỏ cố định (checksum hoặc MAC) và được thêm vào thông điệp.
- Kỹ thuật này giả sử rằng 2 phía tham gia truyền thông là A và B chia sẻ một khoá bí mật K. Khi A có một thông điệp gửi đến B, A sẽ tính toán MAC như là một hàm của thông điệp và khoá:  
 $MAC = C(K, M)$ , với
  - M: thông điệp đầu vào có kích thước biến đổi
  - C: hàm MAC
  - K: khoá bí mật chia sẻ giữa người gửi và người nhận
  - MAC: mã chứng thực thông điệp có chiều dài cố định

# Mã chứng thực thông điệp



# Mã chứng thực thông điệp

- Thông điệp cộng với MAC được truyền tới người nhận. Người nhận thực hiện các tính toán tương tự trên các thông điệp đã nhận sử dụng cùng một khóa bí mật, để tạo ra một MAC mới.
- MAC vừa tạo sẽ được so với MAC nhận. Giả sử chỉ người nhận và người gửi biết khóa bí mật:
  - Nếu MAC nhận phù hợp với MAC vừa tính thì thông điệp không bị thay đổi trong quá trình truyền và chắc chắn được gởi tới từ người gởi đã biết.
  - Nếu MAC nhận khác với MAC vừa tính thì thông điệp đã bị thay đổi hoặc bị giả mạo và được gởi từ attacker.

# Mã chứng thực thông điệp

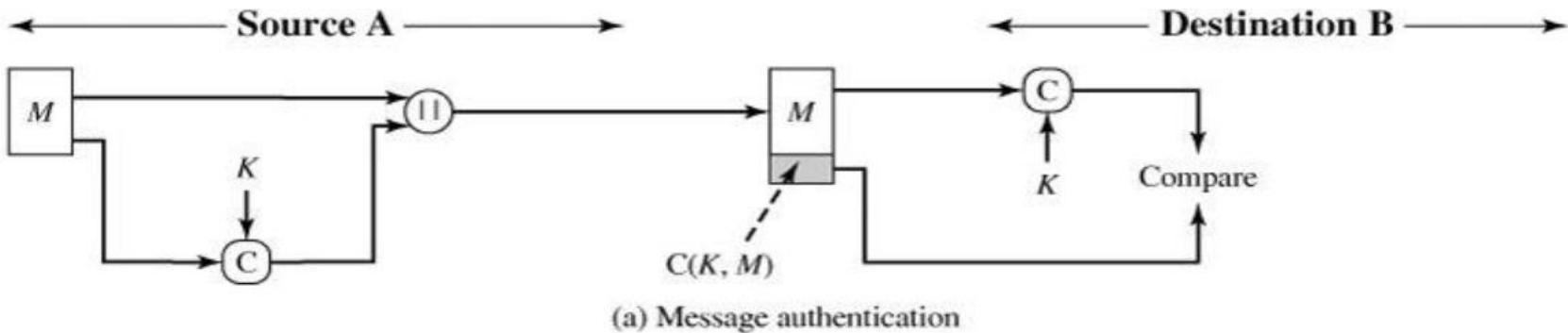
- Chiều dài thông thường của MAC: 32..96 bit → để tấn công cần thực hiện  $2^n$  lần thử với n là chiều dài của MAC (bit).
- Chiều dài thông thường của khoá K: 56..160 bit → để tấn công cần thực hiện  $2^k$  lần thử với k là chiều dài của khoá K (bit).
- Ứng dụng trong:
  - Banking: sử dụng MAC kết hợp triple-DES
  - Internet: sử dụng HMAC và MAC kết hợp AES

# Công dụng của MAC

## 1. ? Chứng thực

$A \quad B: M || C(K, M)$

- Chỉ A và B biết khóa bí mật K nên chỉ có nếu B giải mã và so sánh thành công thì chứng tỏ A là người gửi

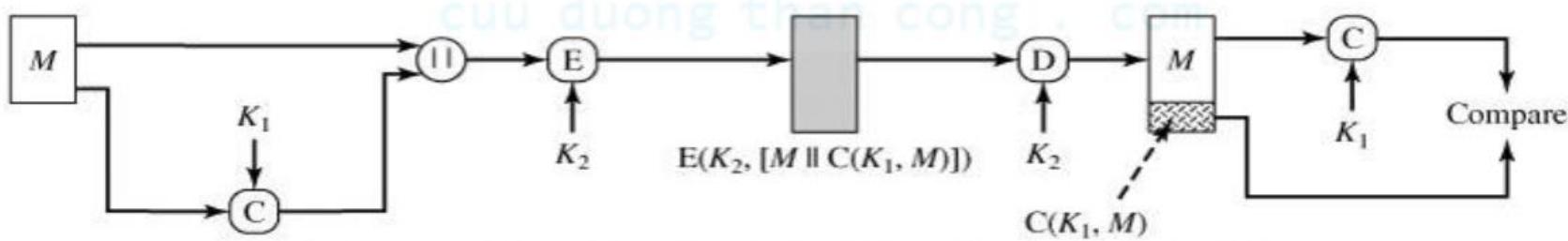


# Công dụng của MAC

## 2. Chứng thực và bảo mật

A → B: E( $K_2$ , [M || C( $K_1$ , M)])

- Chứng thực: chỉ A và B chia sẻ  $K_1$
- Bảo mật: chỉ A và B chia sẻ  $K_2$



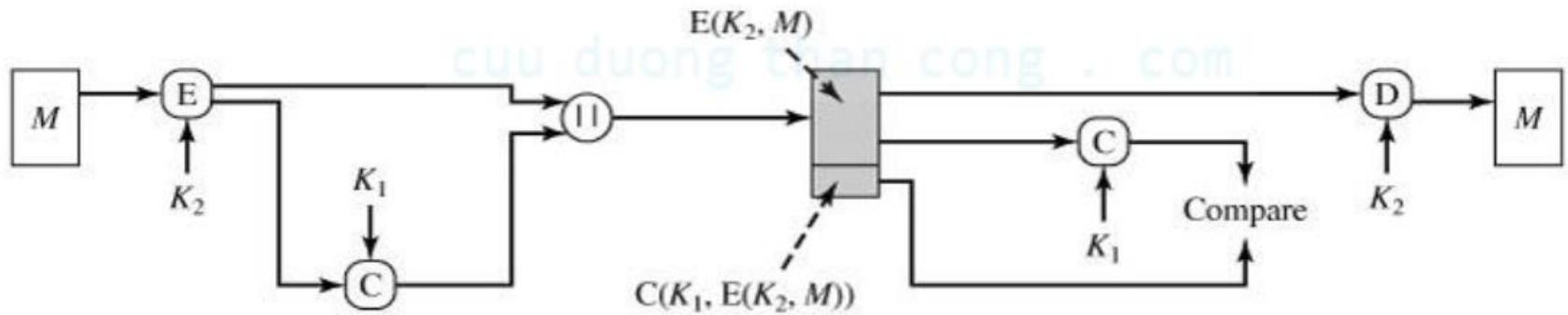
(b) Message authentication and confidentiality; authentication tied to plaintext

# Công dụng của MAC

## 3. Chứng thực và bảo mật

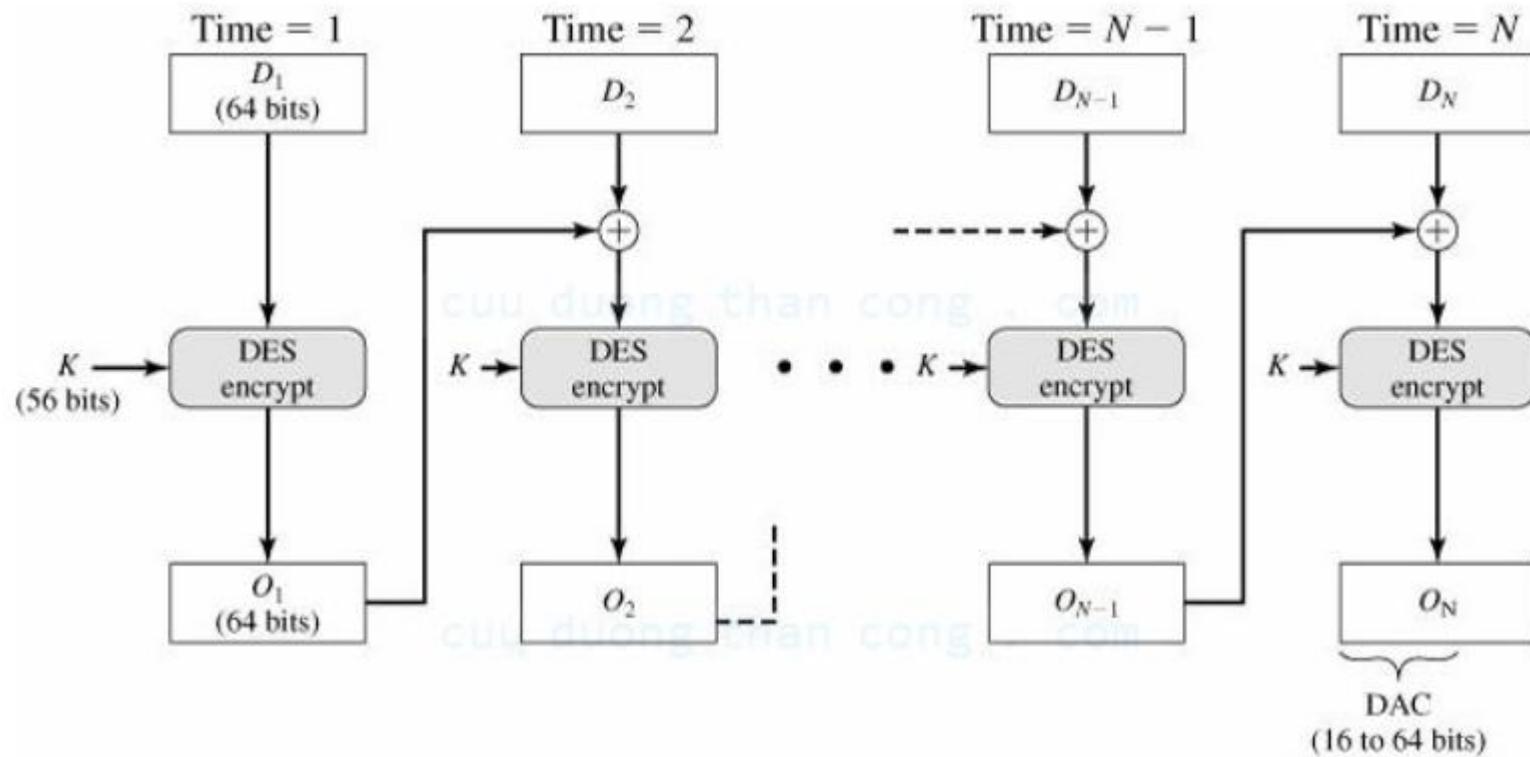
A → B:  $E(K_2, M) \parallel C(K_1, E(K_2, M))$

- Chứng thực: chỉ A và B chia sẻ  $K_1$
- Bảo mật: chỉ A và B chia sẻ  $K_2$



(c) Message authentication and confidentiality; authentication tied to ciphertext

# Tính toán MAC dựa vào DES



# CHỨNG THỰC SỬ DỤNG HÀM BĂM

# Chứng thực dựa theo hàm băm

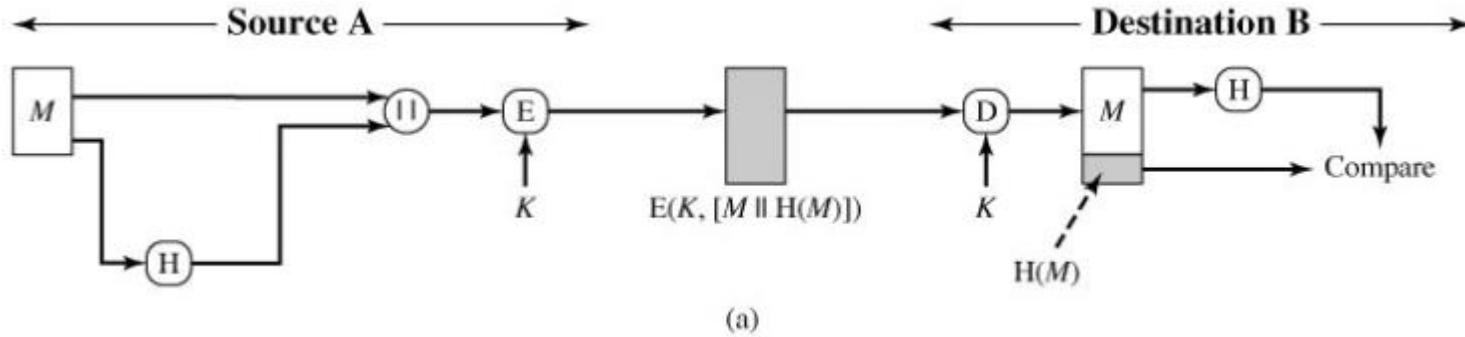
- Một hàm băm nhận một chuỗi dài ở đầu vào, ngắt nó thành nhiều mảnh, trộn lẫn chúng và tạo ra một chuỗi mới với chiều dài ngắn.
- Không phải mọi hàm băm đều thích hợp cho việc tạo ra một dấu vân tay kỹ thuật số.
- Để sinh ra một dấu vân tay kỹ thuật số tốt, hàm băm H cần phải có:
  - Thuộc tính một chiều (one-way property)
  - Thuộc tính duy nhất.
- Hàm băm này được gọi là một hàm băm mật mã (Cryptographic Hash Function – CHF)

# Công dụng cơ bản của hàm băm

## 1. Mã hoá thông điệp cộng với mã băm

$A \rightarrow B: E(K, [M || H(M)])$

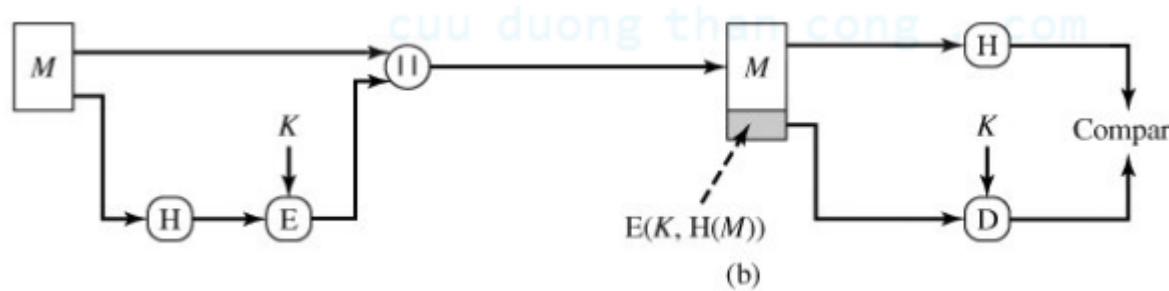
- Bảo mật: chỉ A và B chia sẻ K
- Chứng thực:  $H(M)$  được bảo vệ bằng mật mã



## Công dụng cơ bản của hàm băm

### 2. Mã hoá mã băm chia sẻ với khoá bí mật

- $A \rightarrow B: M \parallel E(K, H(M))$
- Chứng thực:  $H(M)$  được bảo vệ bằng mật mã

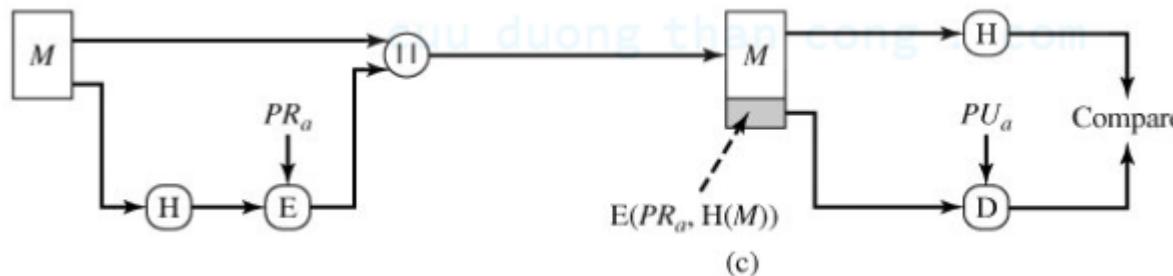


## Công dụng cơ bản của hàm băm

### 3. Mã hoá khoá bí mật với mã băm của người gửi

$A \rightarrow B: M \parallel E(PR_A, H(M))$

- Chứng thực và chữ ký số:
- $H(M)$  được bảo vệ bằng mật mã
- Chỉ A có thể tạo  $E(PR_A, H(M))$

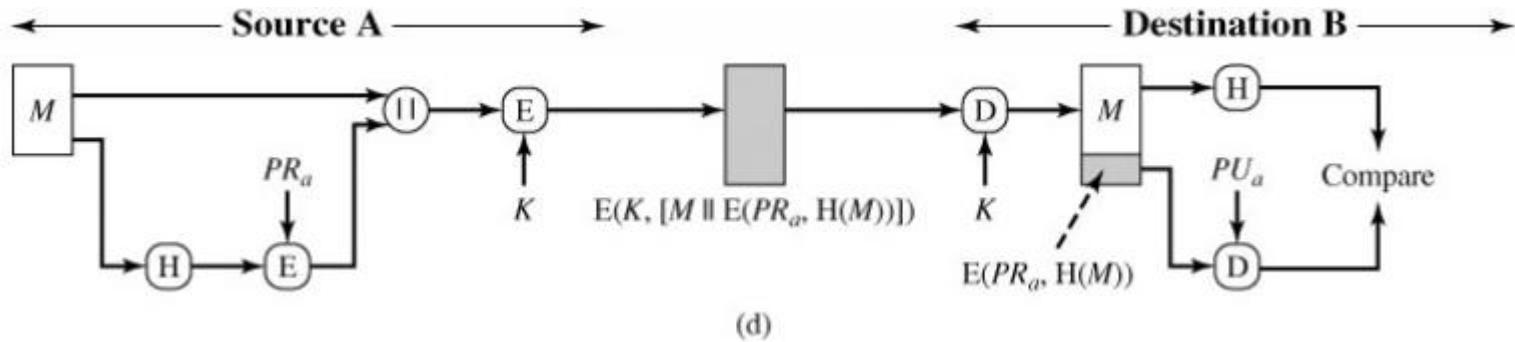


## Công dụng cơ bản của hàm băm

### 4. Mã hoá kết quả của (c) với khoá bí mật chia sẻ

$A \rightarrow B: E(K, [M || E(PRA, H(M))])$

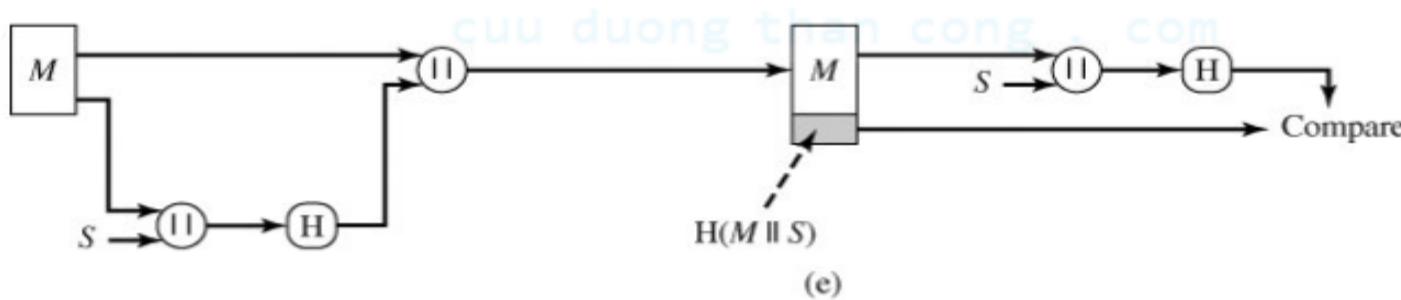
- Bảo mật: chỉ A và B chia sẻ K
- Chứng thực và chữ ký số



## Công dụng cơ bản của hàm băm

### 5. Tính mã băm của thông điệp công với trị bí mật

- $A \rightarrow B: M \parallel H(M \parallel S)$
- Chứng thực: chỉ A và B chia sẻ S

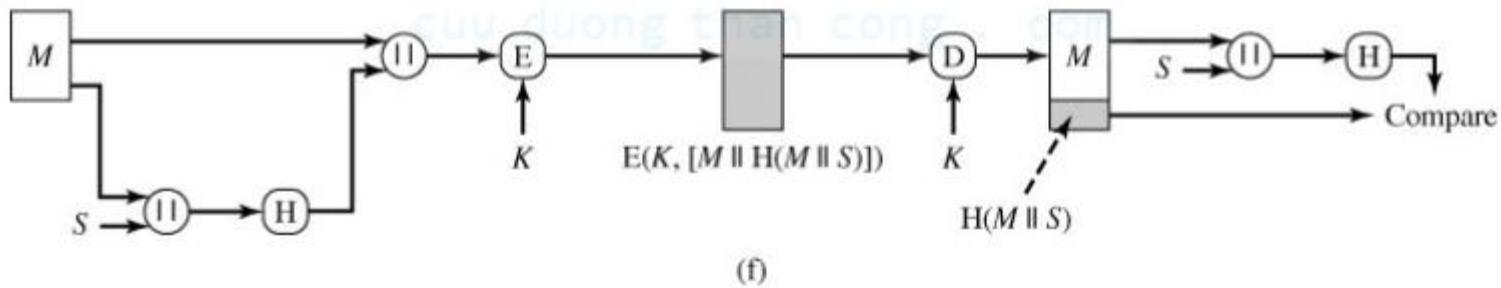


# Công dụng cơ bản của hàm băm

## 6. Mã hoá kết quả của (e)

$A \rightarrow B: E(K, [M \parallel H(M \parallel S)])$

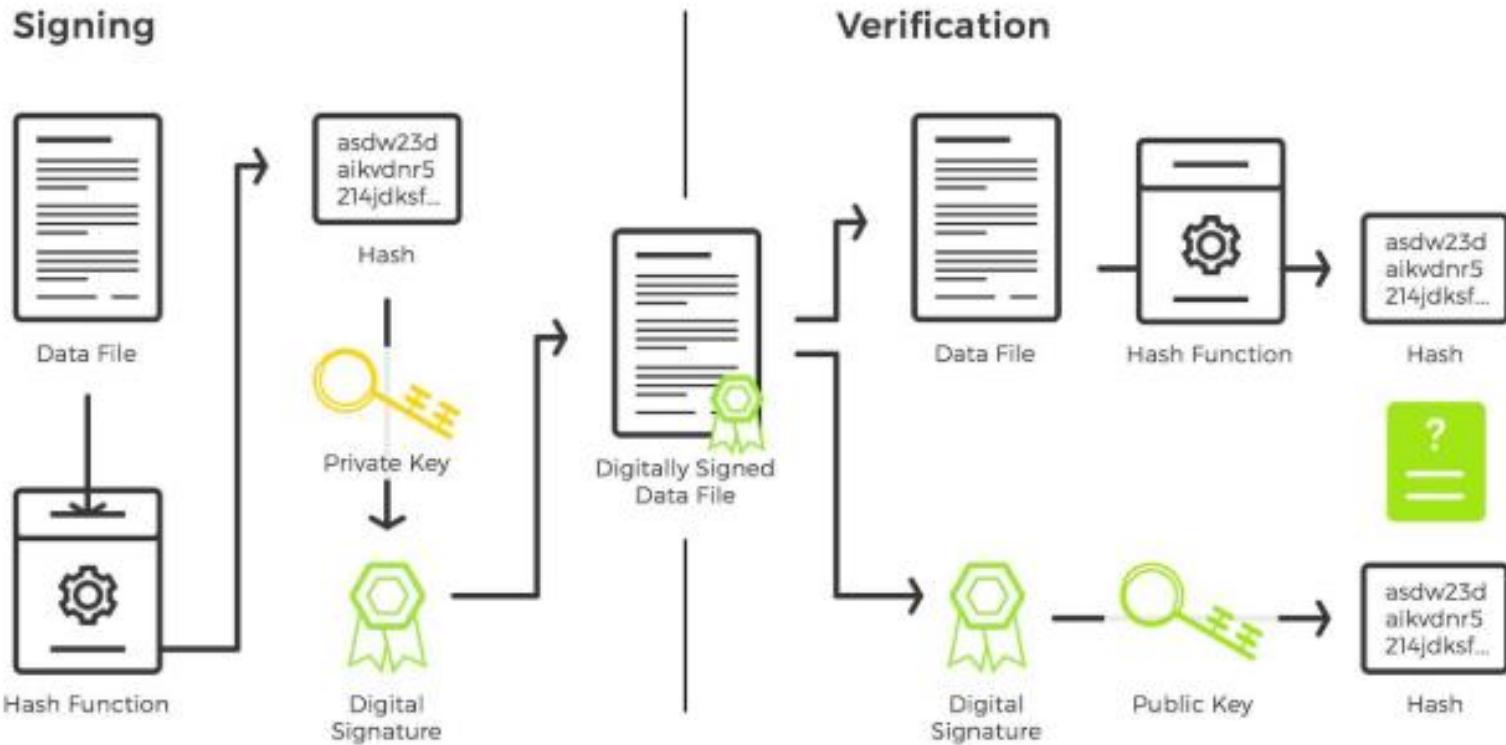
- Chứng thực: chỉ A và B chia sẻ S
- Bảo mật: chỉ A và B chia sẻ K



(f)

# CHỮ KÝ SỐ

# Chữ Ký số



# Chữ Ký số

**Sử dụng khoá công khai để tạo chữ ký số:**

- Giả sử A cần gởi cho B một thông điệp mật kèm chữ ký điện tử, A sẽ sử dụng khoá công khai của B để mã hoá thông điệp rồi dùng khoá cá nhân của mình để mã hoá chữ ký, sau đó gởi cả thông điệp lẫn chữ ký cho B. B sẽ dùng khoá công khai của A để giải mã chữ ký, rồi dùng khoá cá nhân của mình để giải mã thông điệp của A.
- Việc tạo chữ ký và kiểm chứng chữ ký thường được thực hiện nhờ hàm băm.

# Mã Hóa Ứng dụng-Chương 7



Đoàn Trình Dục/ Đoàn Trình Dục

Mail: duc.duantrinh@stu.edu.vn

2019



# NỘI DUNG CHÍNH



Cipher System



Công Dụng Hàm Băm



Tính Chất Hàm Băm



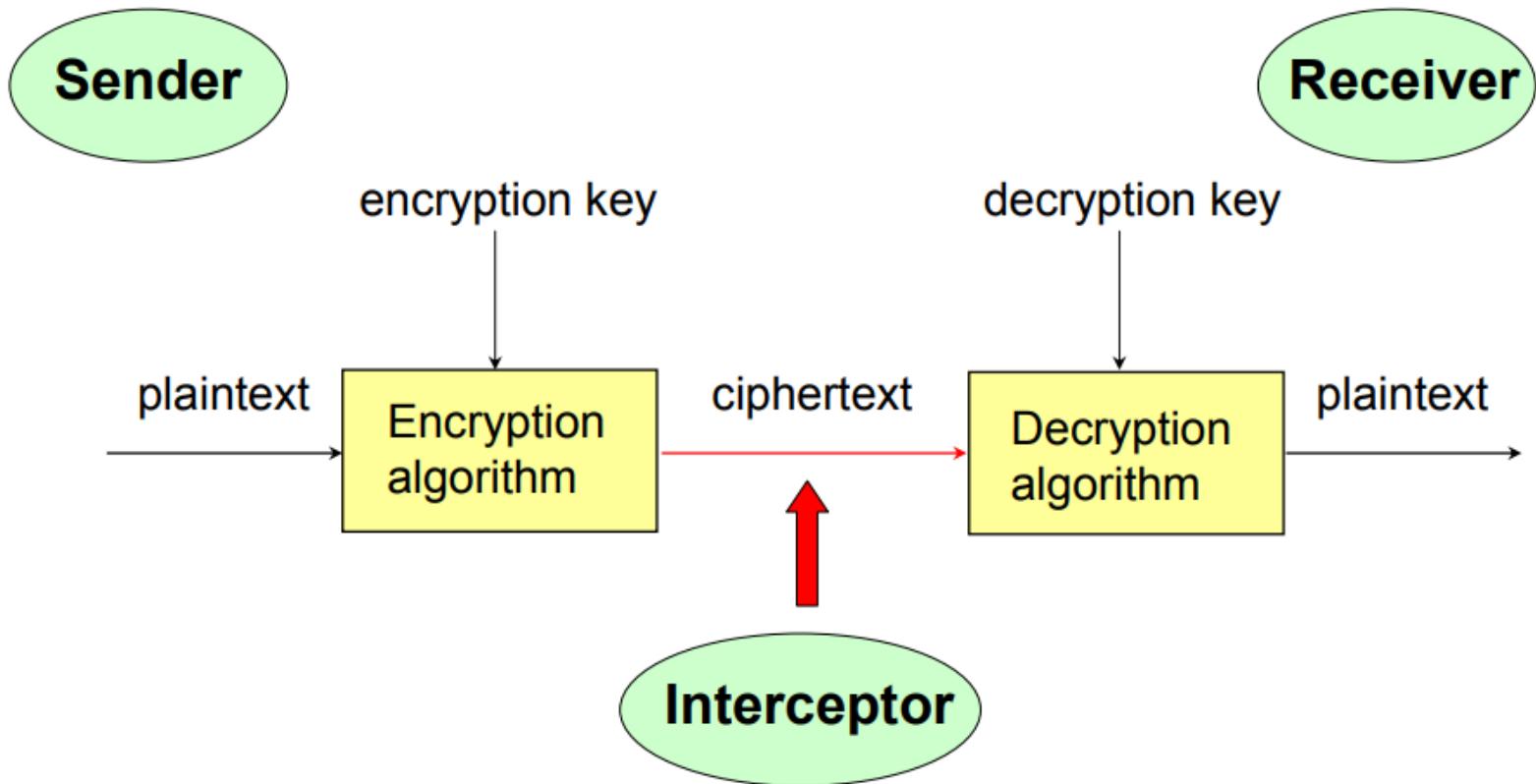
Cấu Trúc Hàm Băm



Tấn Công Hàm Băm

# CIPHER SYSTEM

# Cipher System



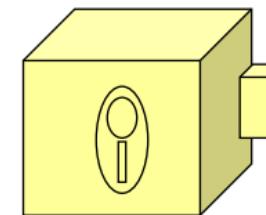
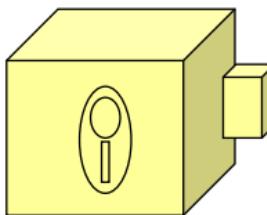
# Cipher System-Symmetric systems

- Trong các hệ thống mật mã đối xứng, khóa giải mã dễ dàng thu được từ khóa mã hóa.
- Đối với hệ thống này khóa mã hóa và khóa giải mã là giống nhau
- Tất cả các hệ thống mật mã thực tế trước 1980 là hệ thống mật mã đối xứng. Thực sự hệ thống đối xứng ngày nay vẫn còn được sử dụng nhiều và không có dấu hiệu nào cho thấy dừng lại

Locking

=

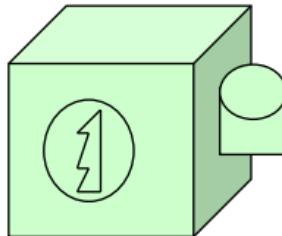
Unlocking



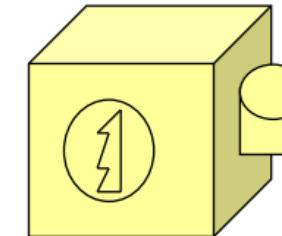
# Cipher System-Public key systems

- Trong các hệ thống mật mã công khai được tính toán để không thể tính được private key từ public key
- Đối với hệ thống này khóa mã hóa và khóa giải mã là Khác nhau
- Hệ thống public key cipher systems còn có tên gọi khác là asymmetric cipher systems

**Anyone can lock**

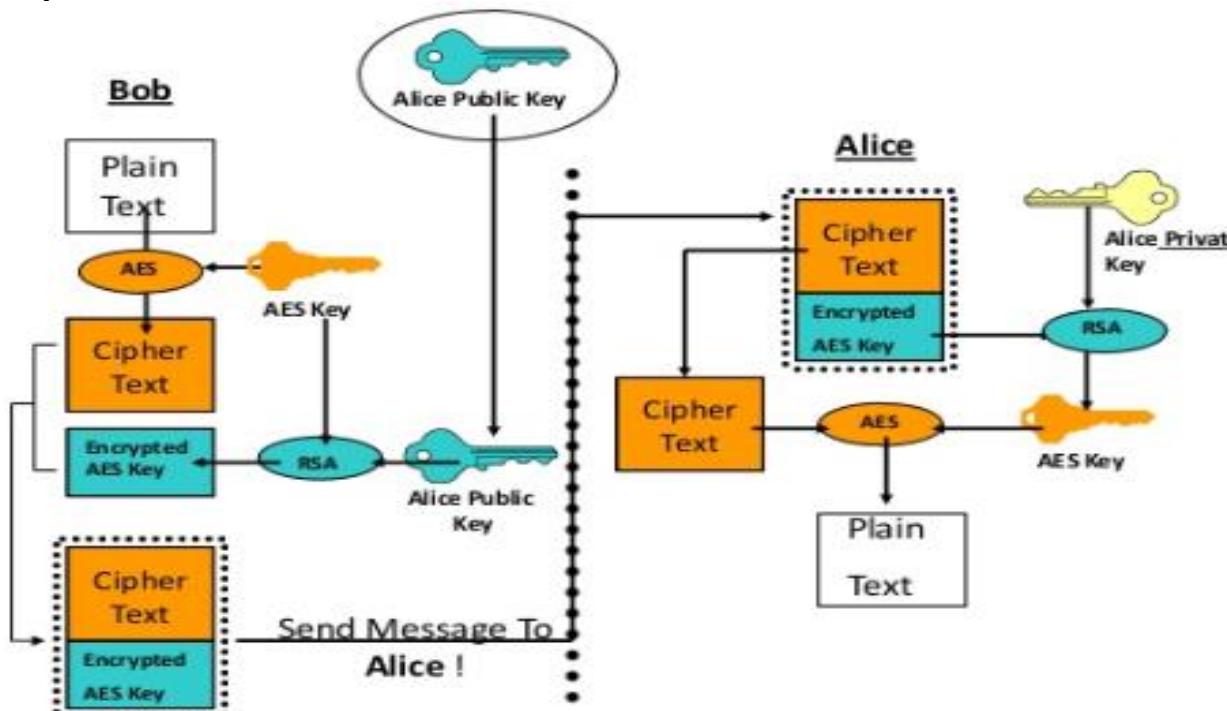


**Only a key holder  
can unlock**



# Cipher System

- Với khả năng đặt khóa công khai các Public key systems có rất nhiều ứng dụng trong thực tế
- Tuy nhiên các hệ thống Public key systems cũng có những vấn đề riêng của nó
- Trong thực tế người ta thường sử dụng cả 2 kỹ thuật này vào trong hệ thống thật

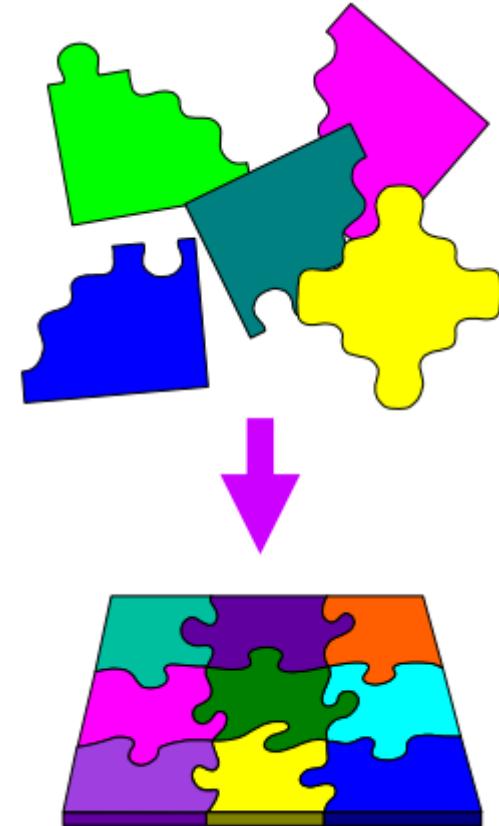


# Cipher System-Service

- Entity authentication:
- Data integrity.
- Data origin authentication
- Non-repudiation

# Security Systems

- Physical Security
- Access Control
- Auditability
- Accountability
- Network Security
- Security Management
- Policies, Standards and Procedures
- **Cryptography**
- Disaster Recovery



# Security Systems

Cryptographic security system muốn được đảm bảo hoạt động ổn định thì chi phí tốn nhất nằm ở chi phí management bao gồm:

- Procedures and Standards
- Audit Trail Management
- User Management
- Token Management (e.g. smart cards)
- **Key Management**
- Access Control
- Security Violations Investigation
- Contingency Planning

# KEY MANAGEMENT

# Key Management

- ANSI X9.17 (Financial Institutions Key Management – Wholesale, 1985)
- Tiêu chuẩn này thiết lập các phương thức (generation, exchange, use, storage and destruction của secret keys).
- Thường là sự kết hợp của:
  - Network topology (e.g. point-to-point, many-to-many)
  - Cryptographic services (e.g. confidentiality, nonrepudiation)
  - Cryptographic mechanisms (e.g. encryption, digital signature)
- Có rất nhiều chuẩn của key management như:
  - ANSI X9.17 / ISO 8732
  - ANSI X9.24
  - ETEBACS (France)
  - AS2805.6.xx (Australia)
  - APACS 40 & APACS 70 (UK)
  - ISO 11166
  - ISO 11568

# Key Management

Key generation

Key destruction

Key Establishment

Key storage

Key change

Key usage

# Key Generation

1. Symmetric keys
  - Random hoặc pseudo-random
  - Loại trừ weak key hoặc semi-weak
  - Sử dụng hàm tạo password hoặc PIN theo chuẩn (VD:PKCS#5)
2. Asymmetric keys
  - Tuân theo một số lý thuyết number-theory
  - Thường là phải search( tốn thời gian nhất định)
3. Pseudorandom phải có thuộc tính sau:
  - Uncorrelated sequences
  - Long period
  - Uniformity
  - Efficiency
4. Key length phải có độ dài tương ứng với thời gian tồn tại và độ quan trọng của dữ liệu nó mã hóa

# Key Storage

Khóa bí mật cần được lưu trữ an toàn:

- Bên trong tamper-resistant hardware security module(phần cứng chống giả mạo)
- Trên smart card hoặc token
- Được mã hóa bằng khóa khác và được lưu trữ trên cơ sở dữ liệu
- Lưu trữ các khóa trong phần mềm được coi có độ bảo mật kém hơn so với các phần cứng chuyên dụng
- Khóa bí mật có thể được lưu trữ trong thời gian dài

# Key Change-Key destruction

Tất cả các cryptographic systems cần có cơ chế thay đổi key:

- Định kỳ(theo kế hoạch ví dụ 2 tháng 1 lần)
- Không theo kế hoạch

Key khi không được sử dụng hoặc lâu không sử dụng phải được xóa một cách an toàn

Theo [ANSI X9.17 \(Section 3.6.1\)](#):

*“Paper-based keying materials shall be destroyed by crosscut, shredding, burning or pulping. Keying material stored on other media shall be destroyed so that it is impossible to recover by physical or electronic means.”*

## Key usage

- Key chỉ sử dụng khi có mục đích thật sự
- Key nên được sử dụng riêng biệt
- Sự riêng biệt nên được thực hiện bằng module phần cứng

# KEY ESTABLISHMENT

# Diffie-Hellman key agreement

- Basic

1. One-time Setup



$(p = 23, \alpha = 5)$



3. (a)(b) random  $x$  and  $y$   
and sends  $\alpha^x$  and  $\alpha^y$

$x = 6$



$$A = 5^6 \bmod 23$$



$y = 15$



$$B = 5^{15} \bmod 23$$

2. Protocol Message  
 $A \rightarrow B : \alpha^x$

$$A = 8$$

$\alpha^x$

$$B = 19$$

3. (c)(d) receives  $\alpha^y$  and  $\alpha^x$   
and shares  $(\alpha^y)^x = (\alpha^x)^y$

$$K = 19^6 \bmod 23$$

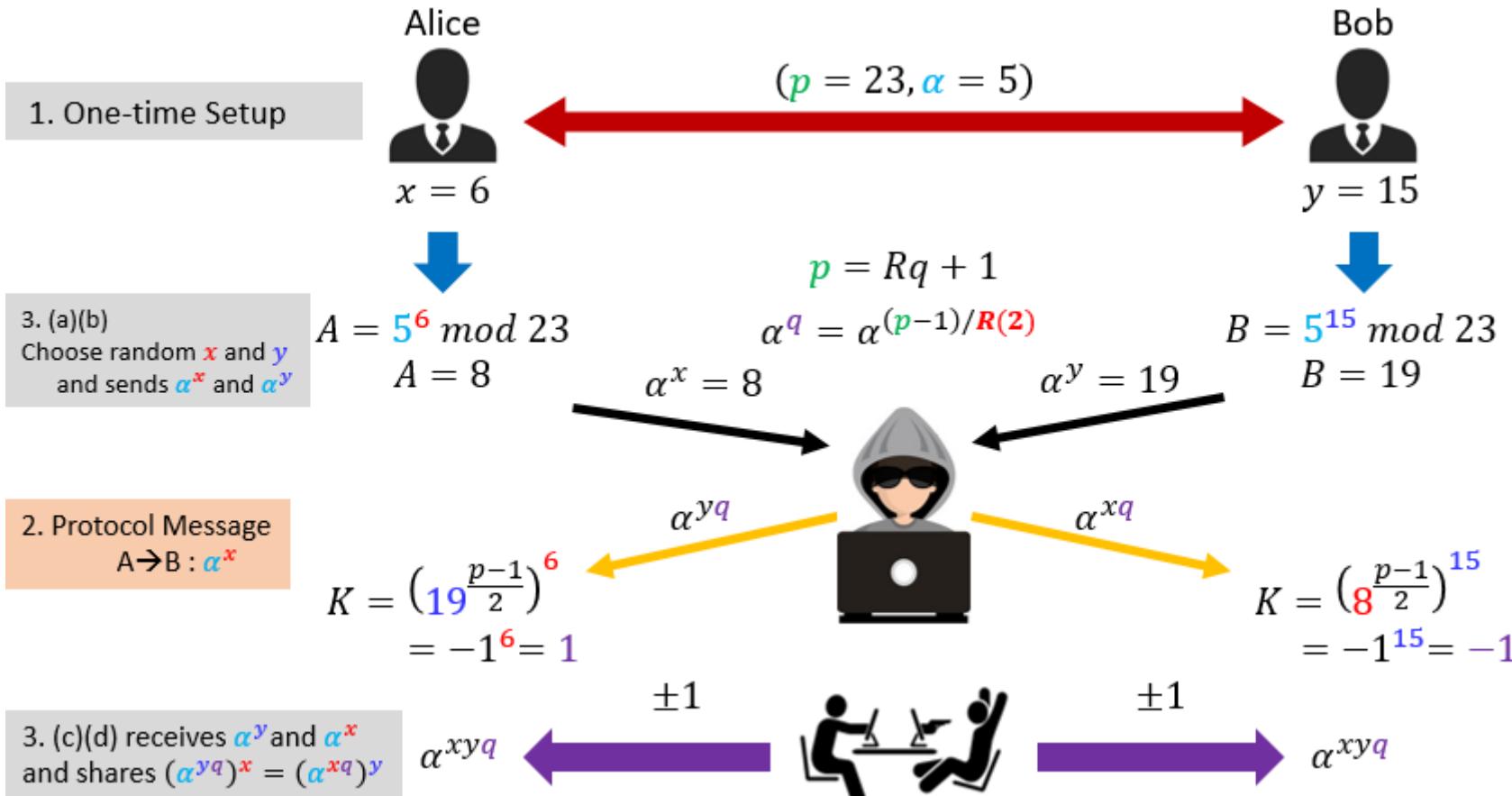
$$K = 2$$

$$K = 8^{15} \bmod 23$$

$$K = 2$$

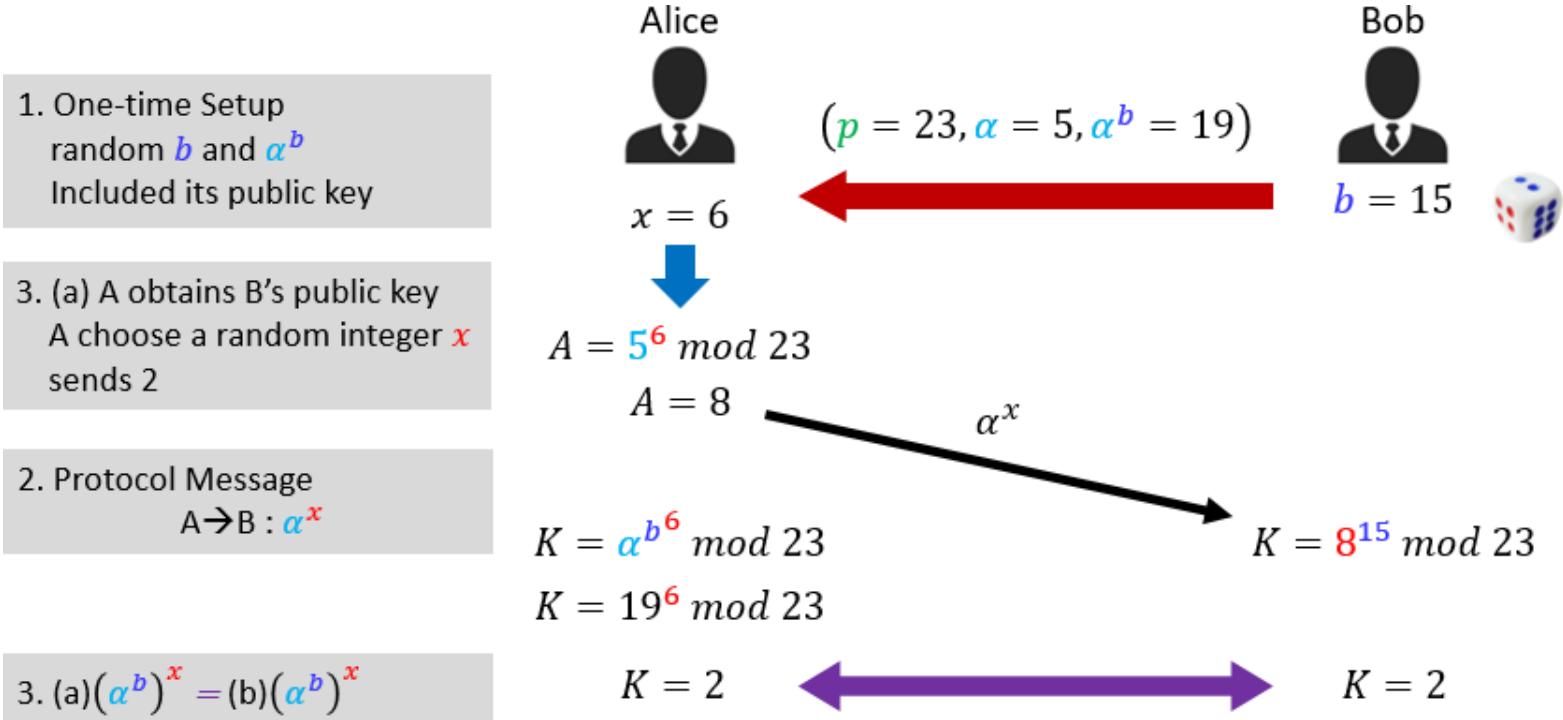
# Diffie-Hellman key agreement

- Vulnerability to not authenticated exponentials



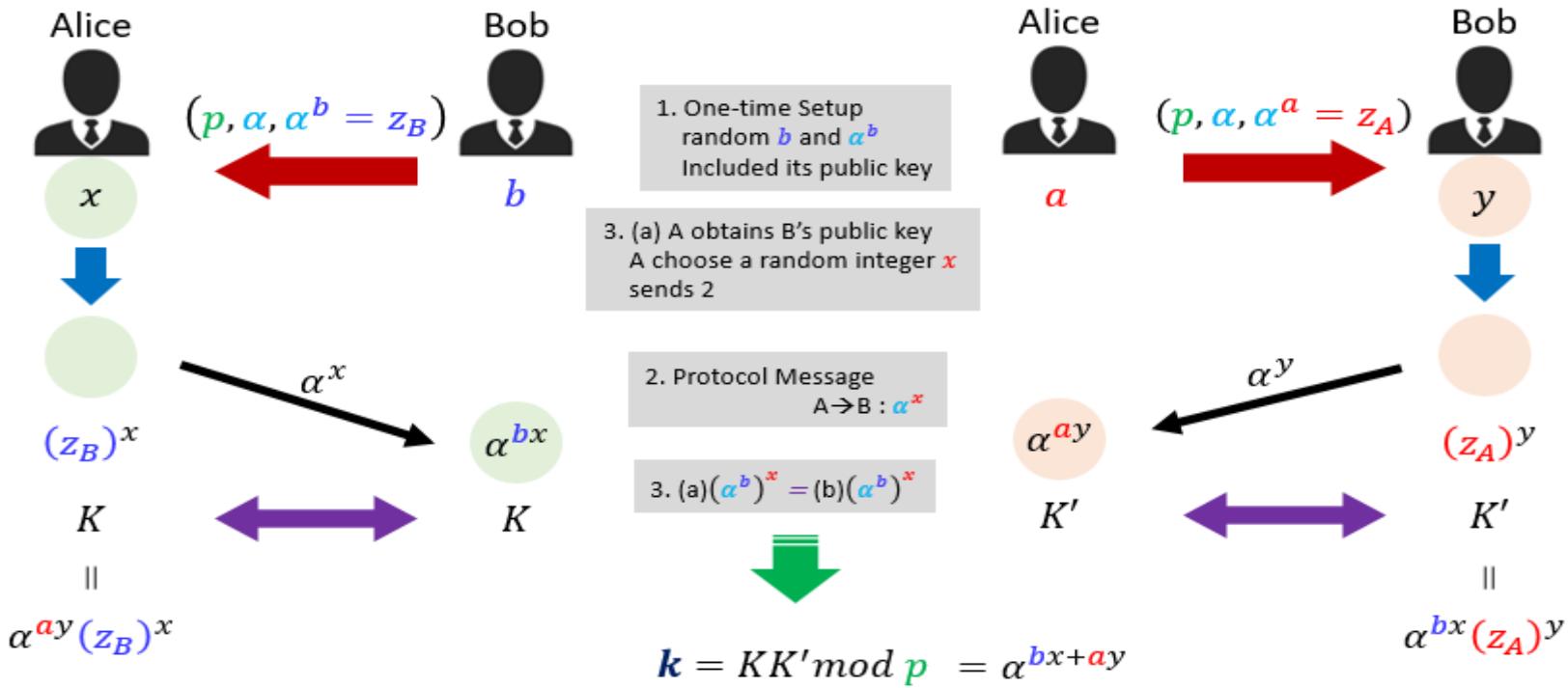
# Diffie-Hellman key agreement

- ElGamal key agreement in one-pass (half-certificated Diffie-Hellman)



# Diffie-Hellman key agreement

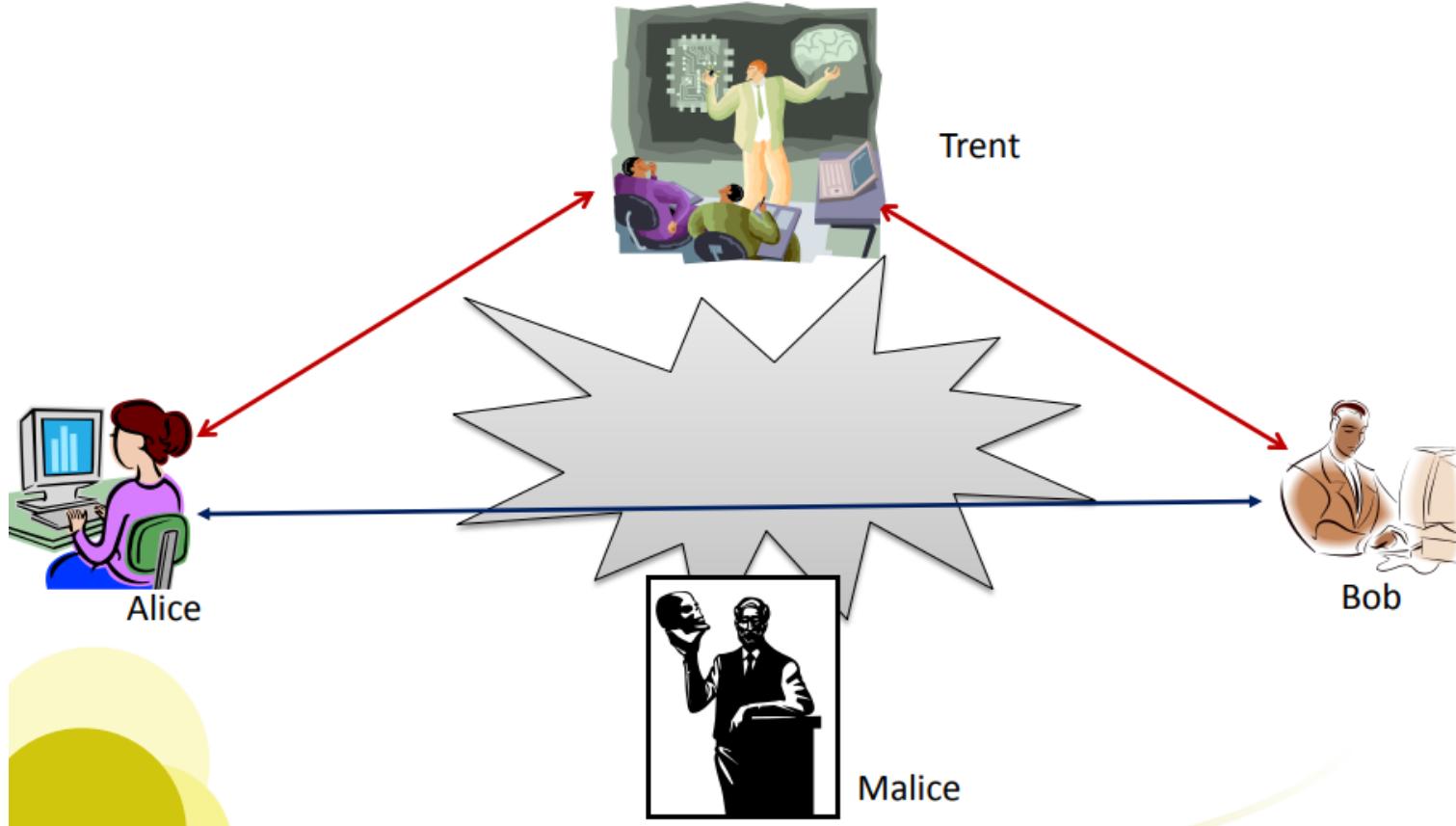
- MTI two-pass key agreement protocols



↓Protocol	$m_{AB}$	$m_{BA}$	$K_A$	$K_B$	key $K$
MTI/A0	$\alpha^x$	$\alpha^y$	$m_{BA}^a z_B^x$	$m_{AB}^b z_A^y$	$\alpha^{bx+ay}$
MTI/B0	$z_B^x$	$z_A^y$	$m_{BA}^{a^{-1}} \alpha^x$	$m_{AB}^{b^{-1}} \alpha^y$	$\alpha^{x+y}$
MTI/C0	$z_B^x$	$z_A^y$	$m_{BA}^{a^{-1}x}$	$m_{AB}^{b^{-1}y}$	$\alpha^{xy}$
MTI/C1	$z_B^{xa}$	$z_A^{yb}$	$m_{BA}^x$	$m_{AB}^y$	$\alpha^{abxy}$

# Key Transport Protocols

- Dolev-Yao Threat Model

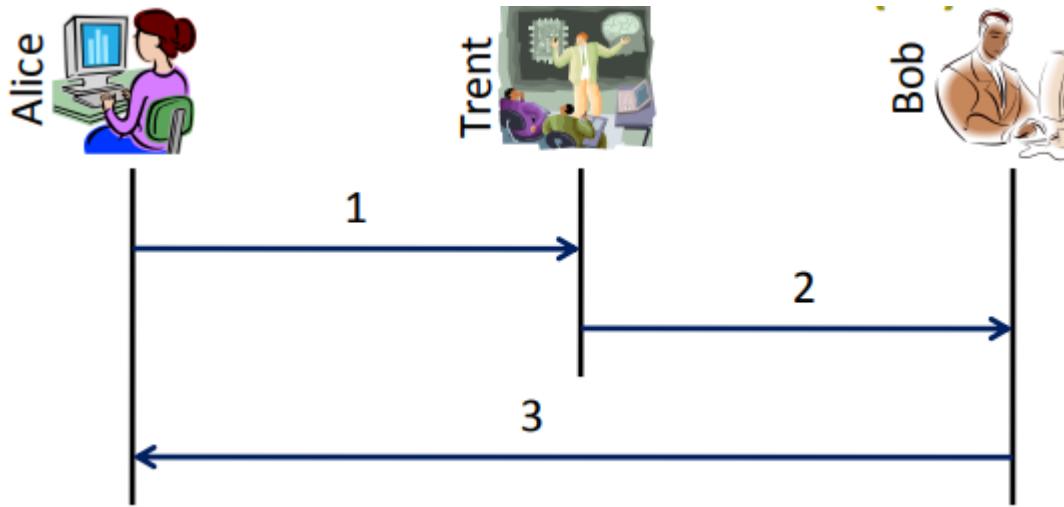


# Key Transport Protocols

## Protocol 1:

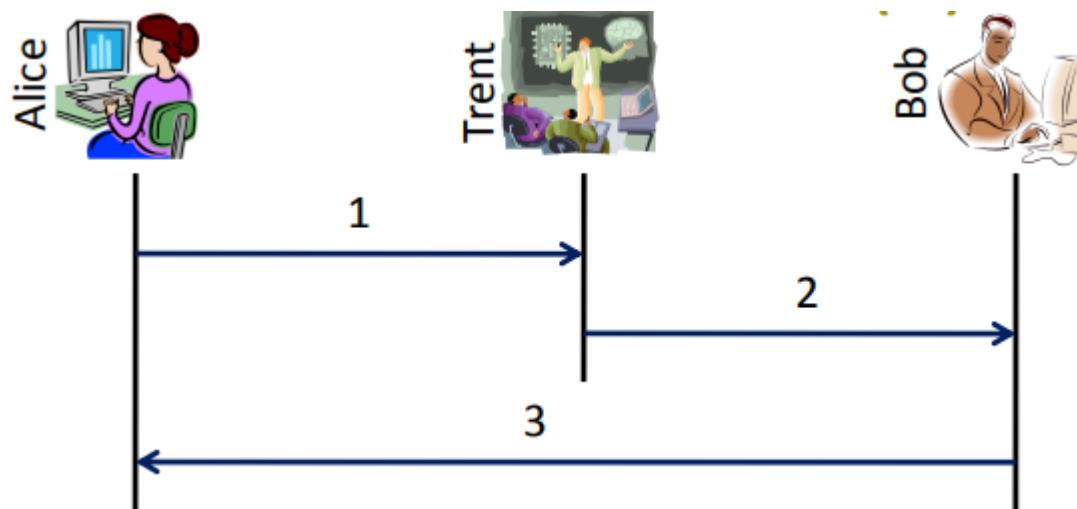
$K_{AT}, K_{BT}$  là key share của Alice và Trent, key share Bob và trent

1. Alice tạo session key  $K$  at random; Tạo  $\{K\}_{K_{AT}}$ ; gửi đến Trent: Alice, Bob,  $\{K\}_{K_{AT}}$
2. Trent tìm  $K_{AT}, K_{BT}$ ; giải mã  $\{K\}_{K_{AT}}$  tìm  $K$ , tạo  $\{K\}_{K_{BT}}$ ; gửi đến Bob: Alice, Bob,  $\{K\}_{K_{BT}}$
3. Bob giải mã  $\{K\}_{K_{BT}}$  được  $K$  và sử dụng  $K$  làm key giao tiếp alice:



# Key Transport Protocols

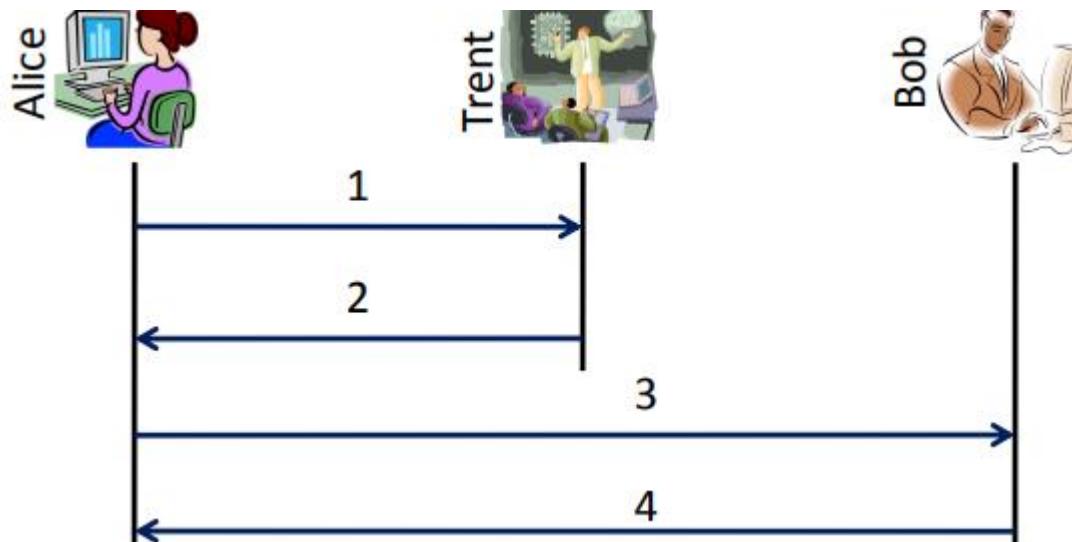
- K được tạo bởi Alice là không đủ mạnh
- Bob không tin tưởng Alice nên khó chấp nhận giao dịch



# Key Transport Protocols

## Protocol 2:

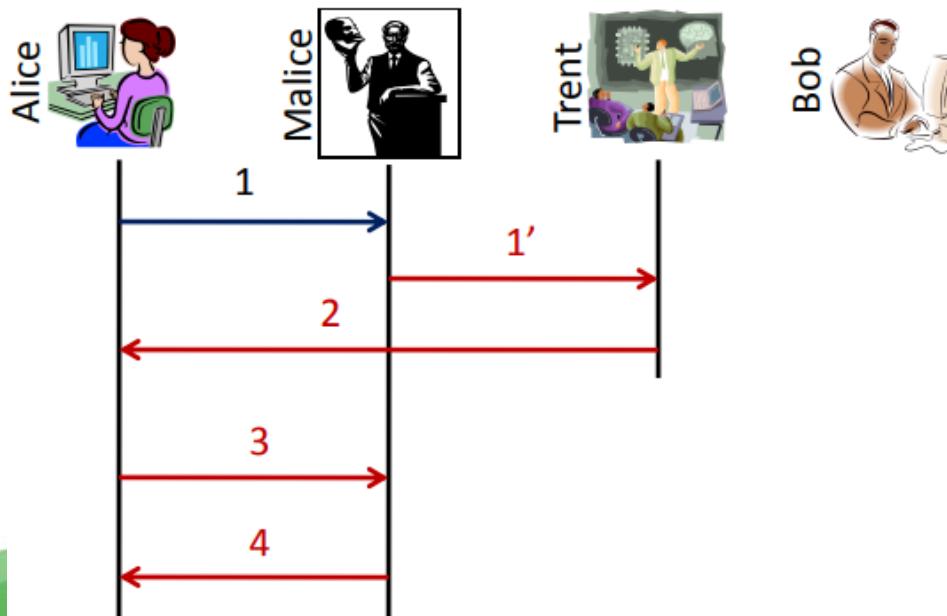
1. Alice gửi đến Trent: Alice, Bob
2. Trent tìm  $K_{AT}, K_{BT}$ ; tạo  $K$  ngẫu nhiên và gửi đến Alice:  $\{K\}_{K_{AT}}, \{K\}_{K_{BT}}$
3. Alice giải mã  $\{K\}_{K_{AT}}$ ; gửi đến Bob: Trent, Alice,  $\{K\}_{K_{BT}}$
4. Bob giải mã  $\{K\}_{K_{BT}}$  tìm  $K$ ; bắt đầu phiên giao dịch



# Key Transport Protocols

## Protocol 2 - attack:

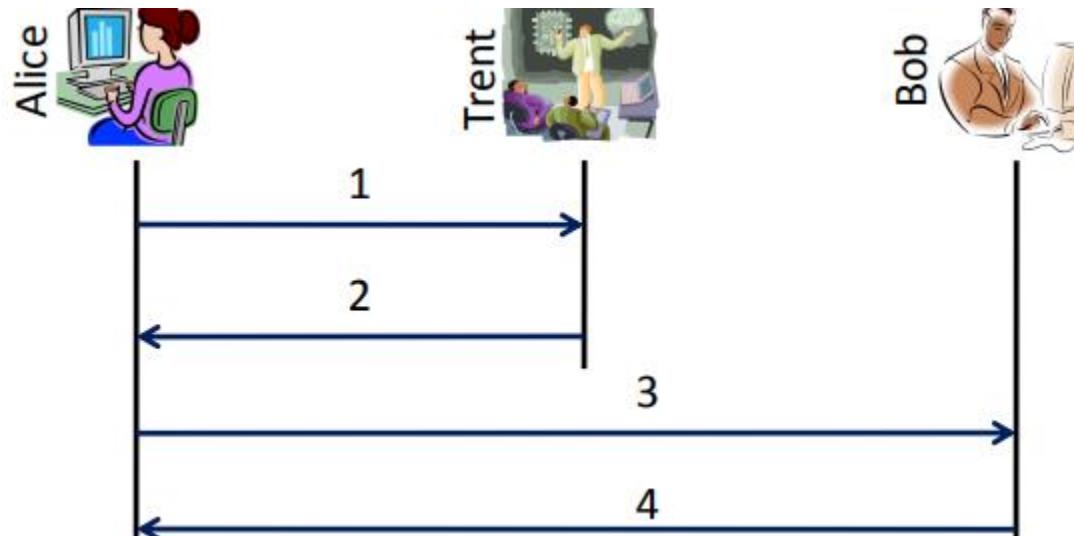
1. Alice gửi đến Malice("Trent"): Alice, Bob
- 1'. Malice("Alice") gửi cho Trent: Alice, Malice
2. Trent finds  $K_{AT}, K_{MT}$ ; tạo  $K$  ngẫu nhiên và gửi cho Alice:  $\{K\}_{K_{AT}}, \{K\}_{K_{MT}}$
3. Alice giải mã  $\{K\}_{K_{AT}}$ ; gửi đến Malice("Bob"): Trent, Alice,  $\{K\}_{K_{MT}}$
4. Malice("Bob") decrypts  $\{K\}_{K_{MT}}$  tìm  $K$ ; bắt đầu phiên giao dịch với tư cách Bob



# Key Transport Protocols

## Protocol 2 - fix:

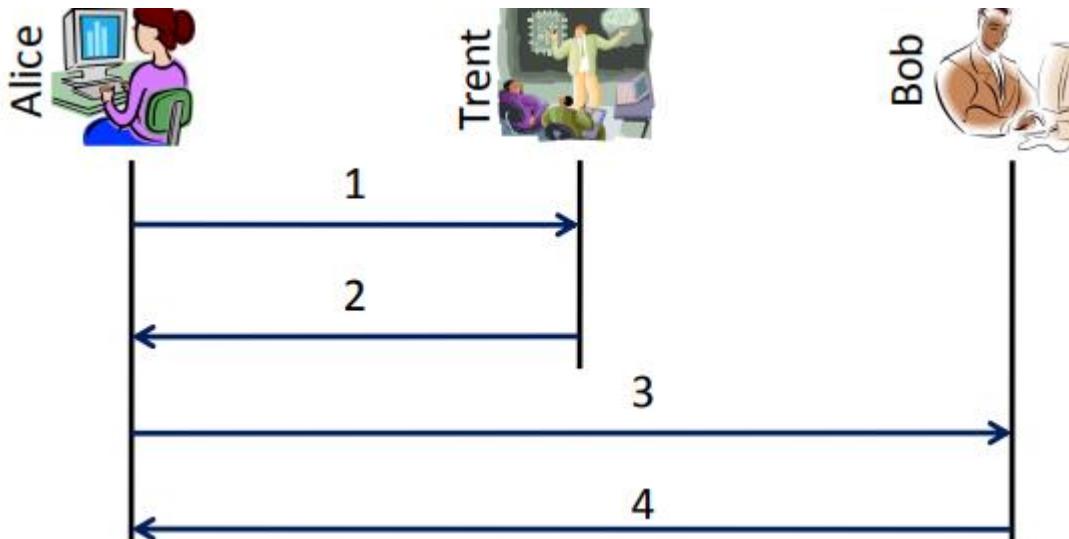
1. Alice gửi đến Trent: Alice,  $\{Bob\}_{K_{AT}}$
2. Trent tìm  $K_{AT}, K_{BT}$  bằng cách giải mã  $\{Bob\}_{K_{AT}}$ ; tạo  $K$  ngẫu nhiên và gửi đến Alice:  $\{K\}_{K_{AT}}, \{K\}_{K_{BT}}$
3. Alice giải mã  $\{K\}_{K_{AT}}$ ; gửi đến Bob: Trent, Alice,  $\{K\}_{K_{BT}}$
4. Bob giải mã  $\{K\}_{K_{BT}}$  tìm  $K$ ; bắt đầu phiên giao dịch



# Key Transport Protocols

## Protocol 2' - attack:

1. Alice sends to Malice: ~~Alice,  $\{Bob\}_{K_{AT}}$~~
- 1'. Malice("Alice") gởi đến Trent:  $Alice, \{Malice\}_{K_{AT}}$
2. Trent tìm  $K_{AT}$ , giải mã  $\{Bob\}_{K_{AT}}$  đã được *thay thế bởi*  $\{Malice\}_{K_{AT}}$ ; Trent tìm  $K_{MT}$ ; tạo K ngẫu nhiên Alice:  $\{K\}_{K_{AT}}, \{K\}_{K_{MT}}$
3. Alice giải mã K; gởi cho Bob:  $Trent, Alice, \{K\}_{K_{MT}}$
4. Malice giải mã K ; giao dịch với Alice với tư cách Bob



# Key Transport Protocols

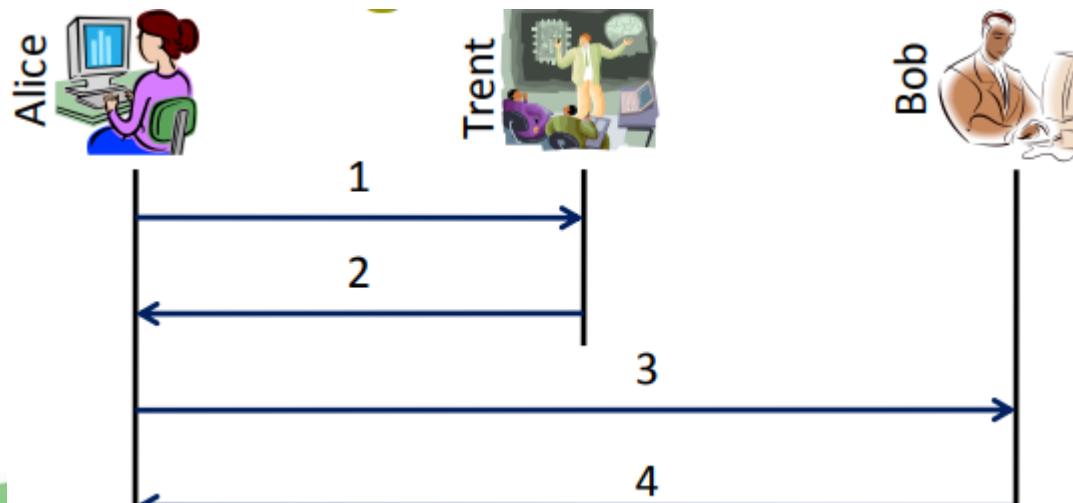
## Protocol 2' - attack:

- Malice có thể thay thế thông điệp mà không bị phát hiện.
- Giao thức cần có thêm cách chống giả mạo.
- Protocol “Message Authentication”

# Key Transport Protocols

## Message Authentication:

1. Alice gửi đến Trent: Alice, Bob
2. Trent tìm  $K_{AT}$ ,  $K_{BT}$ ; tạo  $K$  ngẫu nhiên và gửi cho Alice:  $\{Bob, K\}_{K_{AT}}$ ,  $\{Alice, K\}_{K_{BT}}$
3. Alice giải mã  $\{Bob, K\}_{K_{AT}}$ , kiểm tra thông tin Bob's identity và gửi cho Bob:  $Trent, \{Alice, K\}_{K_{BT}}$
4. Bob giải mã  $\{Alice, K\}_{K_{BT}}$ , kiểm tra Alice's identity và bắt đầu phiên giao dịch



# Key Transport Protocols

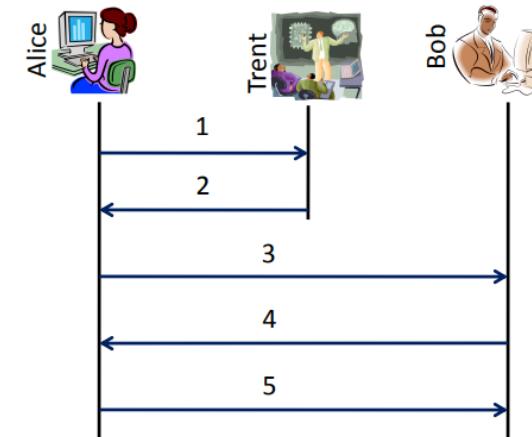
## Message Authentication:

- Malice không thể chỉnh sửa  $\{Bob, K\}_{K_{AT}}$ ,  $\{Alice, K\}_{K_{BT}}$  mà không bị phát hiện
- → replay attack
- Malice có thể nhận được bước 1 của Alice và thay thế như sau:
- $\{Bob, K\}_{K_{AT}}$ ,  $\{Alice, K\}_{K_{BT}} \rightarrow \{Bob, K'\}_{K_{AT}}$ ,  $\{Alice, K'\}_{K_{BT}}$
- K' là key ở phiên làm việc trước đó

# Key Transport Protocols

## Challenge-response:

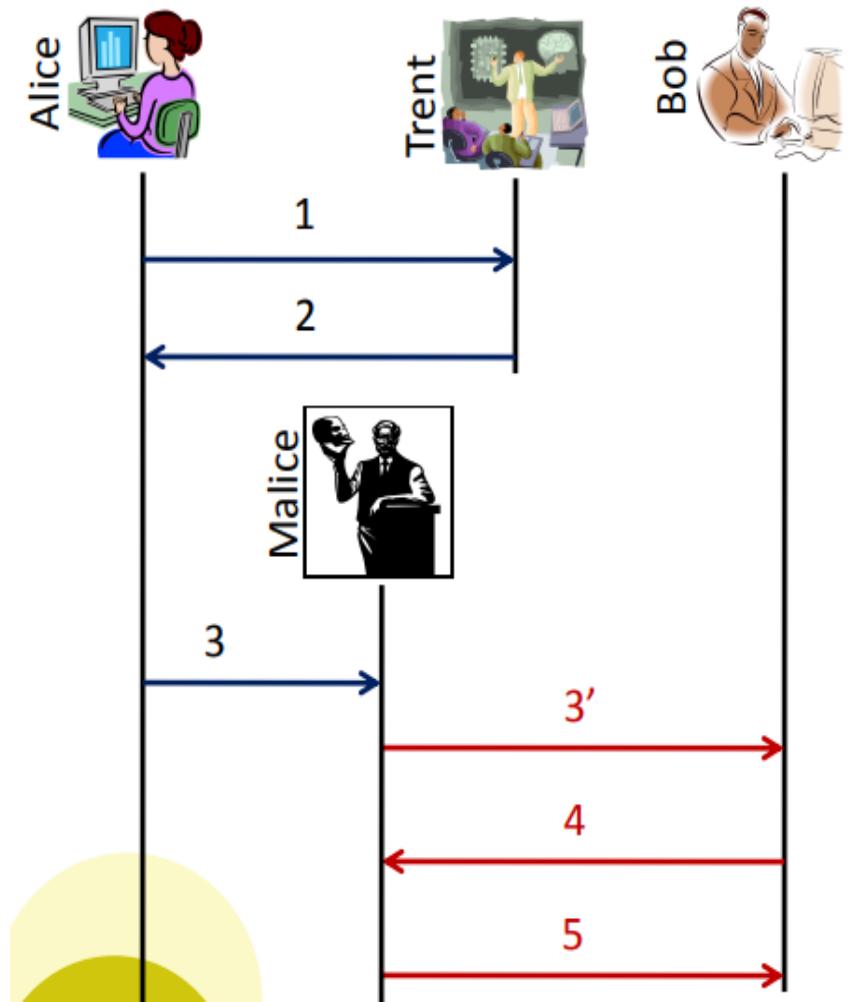
1. Alice tạo số NA ngẫu nhiên và gửi cho Trent: Alice, Bob, NA
2. Trent tạo khóa K ngẫu nhiên và gửi cho Alice: {NA , K, Bob, {K, Alice}  $K_{BT}$ }  $K_{AT}$
3. Alice giải mã, kiểm tra số NA , kiểm tra danh định của Bob, và gửi cho Bob: Trent, {K, Alice}  $K_{BT}$
4. Bob giải mã, kiểm tra danh định của Alice, tạo số NB ngẫu nhiên và gửi cho Alice: {I'm Bob! NB }K
5. Alice gửi cho Bob: {I'm Alice!NB -1}K



# Key Transport Protocols

## Challenge-response attack:

1. Alice gửi cho Trent: Alice, Bob, NA
2. Trent gửi cho Alice: {NA , K, Bob, {K, Alice}  $K_{BT}$ }  $K_{AT}$
3. Alice gửi cho Malice("Bob"): Trent, {K, Alice}  $K_{BT}$
- 3'. Malice("Alice") gửi cho Bob: Trent, {K', Alice}  $K_{BT}$
4. Bob giải mã, kiểm tra danh định của Alice, tạo số NB ngẫu nhiên và gửi cho Malice("Alice"): {I'm Bob! NB }K'
5. Malice("Alice") gửi cho Bob: {I'm Alice!NB -1}K'



# Key Transport Protocols

## Challenge-response với timestamp:

1. Alice gửi cho Trent: Alice, Bob
2. Trent gửi cho Alice:  $\{Bob, K, T, \{Alice, K, T\} K_{BT}\} K_{AT}$
3. Alice kiểm tra T và gửi cho Bob:  $\{Alice, K, T\} K_{BT}$
4. Bob kiểm tra T và gửi cho Alice:  $\{I'm Bob! NB\} K$
5. Alice gửi cho Bob:  $\{I'm Alice! NB -1\} K$

Kiểm tra T:  $|Clock - T| < \Delta t_1 + \Delta t_2$

- Clock: đồng hồ tại máy cá nhân
- T: timestamp, giờ tại Trent
- $\Delta t_1, \Delta t_2$ : độ lệch múi giờ và độ lệch thời gian cho phép

**Khó thực hiện vì phải đồng bộ thời gian**

# Key Transport Protocols

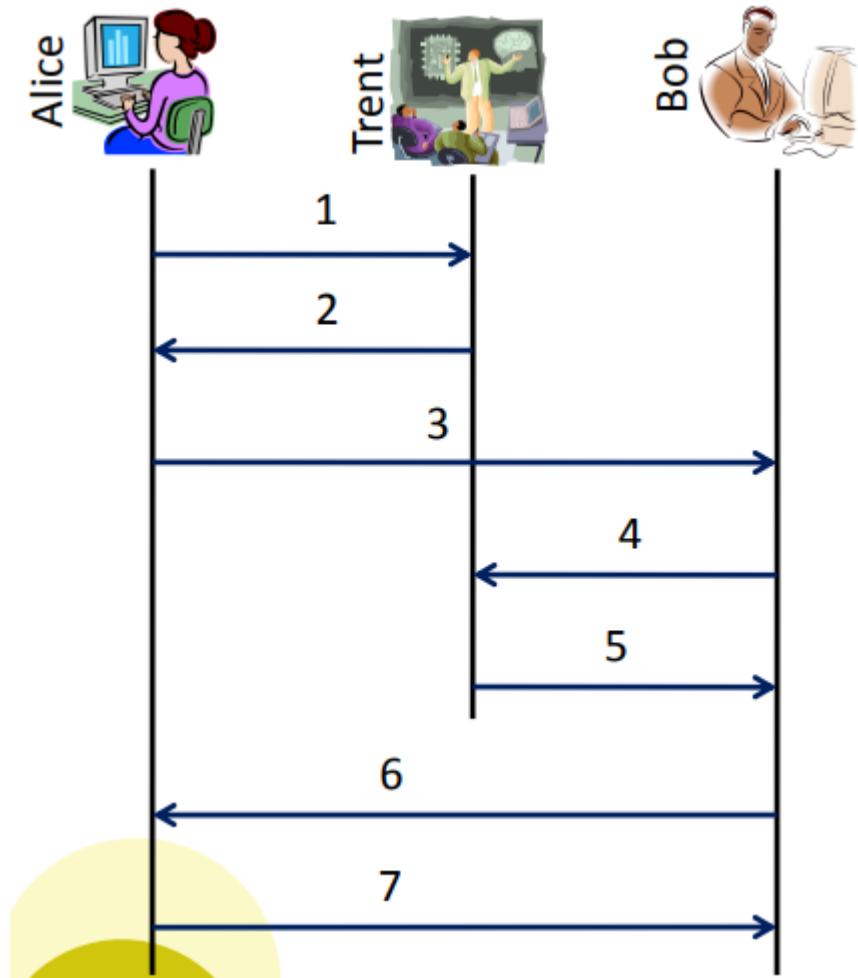
## Giao thức dùng mã công khai

- KA ,  $K^{-1}A$  : là khóa công khai và khóa bí mật của Alice
- KB ,  $K^{-1}B$  : là khóa công khai và khóa bí mật của Bob
- KM,  $K^{-1}M$ : là khóa công khai và khóa bí mật của Malice
- KT ,  $K^{-1}T$ : là khóa công khai và khóa bí mật của Trent
- $\{M\}KA$ : mã hóa M bằng khóa công khai A
- $\{M\} K^{-1}A$  : ký lên M bằng khóa bí mật A

# Key Transport Protocols

## Giao thức dùng mã công khai

1. Alice gửi cho Trent: Alice, Bob
2. Trent gửi cho Alice: {KB , Bob}  
 $K^{-1}T$
3. Alice kiểm tra chữ ký của Trent, tạo số NA và gửi cho Bob: {NA , Alice}KB
4. Bob giải mã, kiểm tra danh định của Alice và gửi cho Trent: Bob, Alice
5. Trent gửi cho Bob: {KA , Alice}  
 $K^{-1}T$
6. Bob kiểm tra chữ ký của Trent, tạo số NB và gửi cho Alice: {NA , NB }KA
7. Alice giải mã và gửi cho Bob: {NB }KB



# Key Transport Protocols

## Giao thức dùng mã công khai

Kết quả của giao thức là Alice và Bob cùng có chung hai số nonce NA và NB . Khóa chung bí mật được tạo thành từ 2 số này.