

Mật mã học (buổi I)

Các khái niệm chung và một số mật mã
cổ điển

Mục lục

- Các khái niệm cơ sở
- Các mô hình tấn công
- Một số mật mã cổ điển
 - Mã mono-alphabetic
 - Mã Vigenere
 - Mã One-time-pad

Mục tiêu và nguyên tắc chung

- Đảm bảo tính mật (Confidentiality)
 - Đảm bảo tài sản không bị truy cập trái phép bởi những người không có thẩm quyền
- Đảm bảo tính nguyên vẹn (Integrity)
 - Đảm bảo tài sản không thể bị sửa đổi, làm giả bởi những người không có thẩm quyền
 - Tính nguyên vẹn của dữ liệu (Data integrity)
 - Tính nguyên vẹn của chủ thể (Origin integrity)
- Tính khả dụng (Availability)
 - Đảm bảo tài sản là sẵn sàng để đáp ứng cho người có thẩm quyền

Các công cụ chung

- Mật mã
- Điều khiển bằng phần mềm
- Điều khiển bằng phần cứng
- Chính sách và các thủ tục
- Điều khiển vật lý

Thế nào là Crypto?

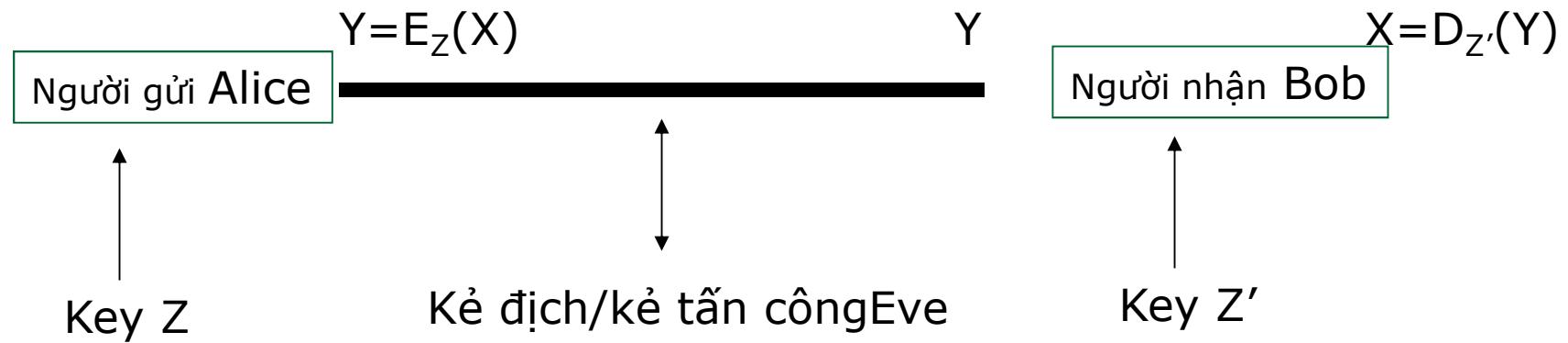
- Xây dựng và phân tích các giao thực mật mã để đạt được các mục tiêu về an toàn thông tin
- Một giao thức (hoặc một cơ chế) là một bộ các thủ tục cho biết các bên tham gia cần phải làm những gì
- Các nhà mật mã học phân tích các giao thức dưới các mô hình tấn công
 - Giả sử khả năng và các hành động có thể của kẻ tấn công
 - Chúng ta cần đứng trên vai trò của kẻ tấn công để suy nghĩ

Các thuật ngữ

- **Cryptography:** là môn học về các kỹ thuật toán học nhằm cung cấp các dịch vụ an toàn thông tin.
- **Cryptanalysis:** là môn học về các kỹ thuật toán học nhằm phá huỷ các dịch vụ an toàn thông tin.
- **Cryptology:** là môn học bao gồm cryptography và cryptanalysis.

Các thuật ngữ

- Plaintexts: bản rõ
- Ciphertexts: bản mã
- Keys: khoá
- Encryption: mã hoá
- Decryption: giải mã



Mật mã khoá bí mật: Secret-key cryptography

- Còn có tên là: mật mã khoá đối xứng -- symmetric cryptography
- Quá trình mã hoá và giải mã dùng cùng một khoá ($Z=Z'$) → *mật mã khoá đối xứng*
- Khoá cần phải giữ bí mật → *mật mã khoá bí mật*
- Vấn đề phân phối khoá: làm sao để chia sẻ khoá giữa A và B

Mật mã khoá công khai: Public-key cryptography

- Còn gọi là: mật mã khoá đối xứng -- asymmetric cryptography
- Khoá dùng để mã hoá và giải mã là khác nhau
 - Không thể suy ra khoá giải mã từ khoá dùng để mã hoá và ngược lại
- Tốn chi phí hơn khoá đối xứng

Is it a secure cipher system?

- **Why insecure**
 - **just break it under a certain reasonable attack model
(show failures to assure security goals)**
- **Why secure:**
 - Evaluate/prove that under the considered attack model, security goals are assured
 - Provable security: Formally show that (with mathematical techniques) the system is as secure as a well-known secure one (usually simpler).

Phá mã

- Có rất nhiều kiểu tấn công, phụ thuộc vào:
 - Kiểu thông tin mà kẻ tấn công có thể có
 - Tương tác với máy mã hoá
 - Năng lực tính toán của kẻ tấn công

Phá mã

- **Tấn công chỉ dựa vào bản mã (Ciphertext-only attack):**
 - Kẻ tấn công chỉ **biết bản mã**
 - Mục tiêu: tìm được bản rõ và khoá
 - Chú ý: một hệ thống bị tấn công bởi kiểu tấn công này thì hoàn hoàn toàn không an toàn
- **Tấn công dựa vào bản rõ (Known-plaintext attack):**
 - Kẻ tấn công biết một vài bản mã và các bản rõ tương ứng
 - Mục tiêu: tìm ra khoá đã được dùng để mã hoá
 - Hoặc tìm ra cách để giải mã các gói tin dùng cùng khoá với các gói tin đã bắt được

Phá mã...

- **Tấn công bản rõ có chọn lựa (Chosen-plaintext attack)**
 - Kẻ tấn công có thể chọn một số các bản rõ và nhận được các bản mã tương ứng
 - Mục tiêu: suy đoán khoá
- **Tấn công bản mã có chọn lựa (Chosen-ciphertext attack)**
 - Tương tự như trên nhưng kẻ tấn công có thể chọn một vài bản mã và nhận được các bản rõ tương ứng.
 - Sự lựa chọn của bản mã có thể thay đổi tùy vào bản rõ nhận được trước đó.

Models for Evaluating Security

- **Unconditional (information-theoretic) security**
 - Assumes that the adversary has unlimited computational resources.
 - Plaintext and ciphertext modeled by their distribution
 - Analysis is made by using probability theory.
 - For encryption systems: **perfect secrecy**, observation of the ciphertext provides no information to an adversary.

Models for Evaluating Security

- **Provable security:**

- Prove security properties based on assumptions that it is difficult to solve a well-known and supposedly difficult problem (NP-hard ...)
 - E.g.: computation of discrete logarithms, factoring

- **Computational security (practical security)**

- Measures the amount of computational effort required to defeat a system using the best-known attacks.
 - Sometimes related to the hard problems, but no proof of equivalence is known.

Models for Evaluating Security

■ Ad hoc security (heuristic security):

- Variety of convincing arguments that every successful attack requires more resources than the ones available to an attacker.
- Unforeseen attacks remain a threat.
- **THIS IS NOT A PROOF**

Mật mã cổ điển

Mã dịch Shift cipher (mã cộng additive cipher)

- Không gian khoá: [1 .. 25]
- Mã hoá với khoá K cho trước:
 - Mỗi ký tự của bản rõ P được mã hoá thành ký tự thứ K sau nó (dịch đi K bước về phía phải):
 - Cách định nghĩa khác: $Y = X \oplus K \rightarrow$ mã cộng
- Giải mã với khoá K cho trước:
 - Dịch trái

I love you -> L oryh brx

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

$$Y = (X + K) \bmod 26$$

$$X = (Y - K) \bmod 26$$

P = CRYPTOGRAPHYISFUN

K = 11

C = NCJAVZRCLASJTDQFY

Ví dụ

- Tìm bản rõ của bản mã sau
 - A B C D E F G H I J K L M N O P Q R S T U V
W X Y Z
 - Djqifs jt csplfo
 - Cipher is broken

Mã dịch: phá mã

■ Duyệt toàn bộ

- Không gian khoá nhỏ (≤ 26 khoá).
- Tìm được K \rightarrow giải mã dễ dàng

Mã thế một bảng thế

General Mono-alphabetical Substitution Cipher

- Không gian khoá: Toàn bộ hoán vị của bảng chữ cái $\Sigma = \{A, B, C, \dots, Z\}$
- Mã hoá với khoá π cho trước:
 - Mỗi ký tự X trong bản rõ P được thay thế bởi ký tự $\pi(X)$ tương ứng trong hoán vị π
- Giải mã với khoá π cho trước:
 - Mỗi ký tự Y trong bản mã C được thay thế bởi ký tự $\pi^{-1}(Y)$ tương ứng trong hoán vị π^{-1}
- **Example:**

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

BECAUSE → AZDBJSZ

Có vẻ an toàn

- Phương pháp duyệt toàn bộ là bất khả thi
 - Không gian khoá lớn: $26! \approx 4 * 10^{26}$
- Được sử dụng phổ biến ở thiên niên kỷ thứ nhất trước công nguyên
- Đã từng được cho là không thể phá được

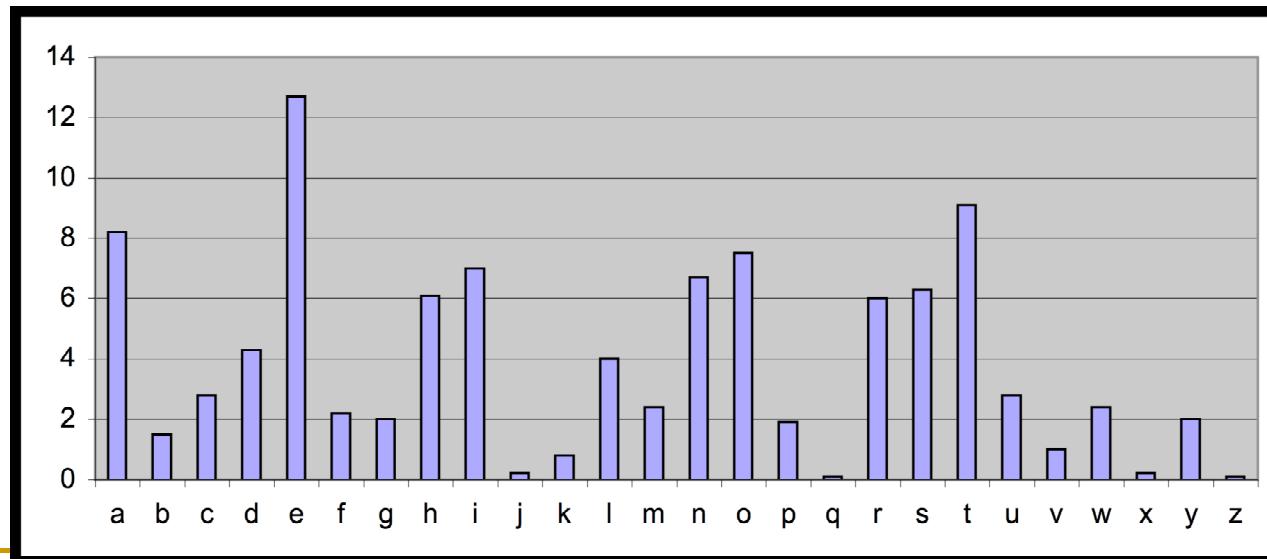
Phá mã một bảng thê

■ Ví dụ

- Bản mã:uxwk lakkvma xvk naac vuuvbdap vcp uxwk wk kwliya
- Biết số các ký tự trong bản rõ là:
 - a: 5, b:1, c:1 d:2, e:6, g:1, h:3, i:4, k:1, l:1, m:2, n:2, p:1, s:7, t:4, các ký tự còn lại không xuất hiện
- Hãy tìm bản rõ ?

Phá mã một bảng thế: phân tích tần số xuất hiện của các ký tự

- Mỗi ngôn ngữ đều có đặc trưng:
 - Tần số xuất hiện của các ký tự, của một nhóm 2 hay nhiều ký tự.
- Mã thế duy trì đặc trưng trên → có nguy cơ bị tấn công bằng cách phân tích tần số xuất hiện của các ký tự



Mã thế: phá mã

- The number of different ciphertext characters or combinations are counted to determine the frequency of usage.
- The cipher text is examined for patterns, repeated series, and common combinations.
- Replace ciphertext characters with possible plaintext equivalents using known language characteristics.
- Example:

THIS IS A PROPER SAMPLE FOR ENGLISH TEXT. THE FREQUENCIES OF LETTERS IN THIS SAMPLE IS NOT UNIFORM AND VARY FOR DIFFERENT CHARACTERS. IN GENERAL THE MOST FREQUENT LETTER IS FOLLOWED BY A SECOND GROUP. IF WE TAKE A CLOSER LOOK WE WILL NOTICE THAT FOR BIGRAMS AND TRIGRAMS THE NONUNIFORM IS EVEN MORE.

- Observations: $f_x=1$ và $f_A=15$.

Mã đa bảng thế (polyalphabetic cipher)

- Sử dụng nhiều bảng thế
- Khóa sẽ quyết định thứ tự hòa trộn của các bảng thế này
- Ví dụ:
 - Bảng thế với từ khóa là 1
 - Tin a b c d
 - Mã B D C A
 - Bảng thế với từ khóa là 2
 - Tin a b c d
 - Mã D C A B
 - Khóa: 21
 - Tin: abcdcbcda
 - Mã: ?

Mã Vigenere

- Một loại mã đa bảng thê

- **Định nghĩa:**

- Giả sử m là một số nguyên dương, $P = C = (Z_{26})^n$, và $K = (k_1, k_2, \dots, k_m)$ là khóa, thê thì:

- **Thuật toán mã hóa:**

$$e_k(p_1, p_2, \dots, p_m) = (p_1+k_1, p_2+k_2, \dots, p_m+k_m) \pmod{26}$$

- **Thuật toán giải mã:**

$$d_k(c_1, c_2, \dots, c_m) = (c_1-k_1, c_2-k_2, \dots, c_m-k_m) \pmod{26}$$

- **Example:**

Tin: C R Y P T O G R A P H Y

Khóa: L U C K L U C K L U C K

Mã: N L A Z E I I B L J J I

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

Ví dụ:

- Khóa: thisisthekey
- Tin: find the cipher of this text
- Mã ?





Mã Vigenere (phá mã)

- Mấu chốt: tìm ra độ dài p của khóa
- Chia bản mã thành p nhóm
 - Với $i=0, p-1$, nhóm i chứa các ký tự mã ở vị trí $kp+i$
⇒ mỗi nhóm có thể xem như là 1 bản mã với dịch
⇒ có thể sử dụng phương pháp thống kê để phá mã theo từng nhóm

Mã Vigenere

Cách tìm độ dài khóa

- Sử dụng chỉ số trùng nhau (Index of coincidence: IC)

- Xác xuất để hai thành phần ngẫu nhiên của một chuỗi có độ dài n là trùng nhau
 - Định nghĩa
 - $x=x\downarrow 1 \ x\downarrow 2 \ \dots \ x\downarrow n$, thế thì $IC(x)=Pr_{x\downarrow i=x\downarrow j \text{ với } i \text{ mọi } x\downarrow i, x\downarrow j \text{ chọn ngẫu nhiên}}$

Mã Vigenere

Cách tìm độ dài khóa

Nhận xét về chỉ số IC

- Tần số xuất hiện của các ký tự càng bằng nhau thì chỉ số IC càng nhỏ
- Tần số xuất hiện của các ký tự càng lệch nhau thì chỉ số IC càng lớn
- Chỉ số IC của ngôn ngữ tự nhiên: 0.068
- Quan hệ của chỉ số IC và p trong thống kê bản mã vigenere
 - p càng lớn ,IC càng nhỏ
 - $p=1 \rightarrow$ mã dịch, p lớn nhất

Key length (p)	1	2	3	4	5	...	10
IC	0.068	0.052	0.047	0.044	0.043	...	0.041

Mã Vigenere

Cách tìm độ dài khóa

- Tìm p sao cho chỉ số IC của các nhóm là lớn nhất
- Code
 1. Set $k=1$
 2. Check if p equals k
 - 2.a. Devide the cipher into k letter groups as before and compute the IC of each.
 - 2.b. If they all are quite the same and approximately equals to 0.068 then $p=k$
If they are quite different to each other and quite smaller than 0.068 then $p>k$
 3. Increase k by 1 and go back to step 2

Mã Vigenere

Cách tìm độ dài khóa

■ Công thức tính IC

$$IC(x) = \frac{\sum_{i=0}^{25} f_i (f_i - 1)}{n(n-1)}$$

Trong đó, f_i là tần số xuất hiện của ký tự alphabet thứ i ở trong x .

Thống kê tần suất xuất hiện của các ký tự trong tiếng anh

- The letters in the English alphabet can be divided into 5 groups of similar frequencies

I e

II t,a,o,i,n,s,h,r

III d,l

VI c,u,m,w,f,g,y,p,b

V v,k,j,x,q,z

- Some frequently appearing bigrams or trigrams

Th, he, in, an, re, ed, on, es, st, en at, to

The, ing, and, hex, ent, tha, nth, was eth, for, dth.

Ví dụ phá mã

- Pjmu mu b amtjfo rfsr. Mr jbu cffi fiaowtrfg cw rjf uvcurmrvrmqi amtjfo. Wqv bof xfow nvahw. Rjf amtjfo jbu cffi coqhfi.
- The letters in the English alphabet can be divided into 5 groups of similar frequencies
 - I e
 - II t,a,o,i,n,s,h,r
 - III d,l
 - VI c,u,m,w,f,g,y,p,b
 - V v,k,j,x,q,z
- Some frequently appearing bigrams or trigrams
 - Th, he, in, an, re, ed, on, es, st, en at, to
 - The, ing, and, hex, ent, tha, nth, was eth, for, dth.

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

- This is a cipher text. It has been encrypted by the substitution cipher. You are very lucky. The cipher has been broken.

One-Time Pad

- Khóa được chọn ngẫu nhiên
 - Plaintext $X = (x_1 \ x_2 \ \dots \ x_n)$
 - Key $K = (k_1 \ k_2 \ \dots \ k_n)$
 - Ciphertext $Y = (y_1 \ y_2 \ \dots \ y_n)$
-
- $e_k(X) = (x_1+k_1 \ x_2+k_2 \ \dots \ x_n+k_n) \text{ mod } m$
 - $d_k(Y) = (x_1-k_1 \ x_2-k_2 \ \dots \ x_n-k_n) \text{ mod } m$

One-time pad

■ Ví dụ

- Plaintext space = Ciphertext space = Keyspace = $\{0,1\}^n$
- Key is chosen randomly
- For example:
- Plaintext is 10001011
- Key is 00111001
- Then ciphertext is 10110010

One-Time Pad

- Khóa chỉ được dùng duy nhất một lần
 - Sau khi dùng sẽ bị hủy
- Khóa rất dài, thường có chiều dài bằng độ dài bản tin
 - Phi thực tế

Information Security

Introduction and course plan

Dr. Nguyen Khanh Van
Dept. of Software Engineering
Hanoi University of Technology

Introduction

- This intro (managerial approach): What the system designer/security architect should know
 - Components of computer security
 - Threats
 - Policies and mechanisms
 - Assurance
 - Operational and Human issues
 - The security life cycle
- Later (technical approach): the security engineer's

Basic Components

- Confidentiality
 - Keeping data and resources hidden
- Integrity
 - Data integrity (integrity)
 - Origin integrity (authentication)
- Availability
 - Enabling access to data and resources

Classes of Threats

- Disclosure
 - Snooping
- Deception
 - Modification, spoofing, repudiation of origin, denial of receipt
- Disruption
 - Modification
- Usurpation
 - Modification, spoofing, delay, denial of service

Snooping

- The unauthorized interception of information, is a form of disclosure.
 - Passive: some entity is listening to/reading communications ...
 - (Passive) wiretapping is a form of snooping in which a network is monitored (wire: the network)
 - Confidentiality services counter this threat.

Modification

- Or alteration, an unauthorized change of information, covers three classes of threats.
 - Deception: incorrect information is accepted as correct/ wrong decision is made.
 - Disruption and usurpation: If the modified data controls the operation of the system
- Active wiretapping is a form of modification in which data moving across a network is altered.
 - Example: the man-in-the-middle attack
- Integrity services counter this threat.

The man-in-the-middle attack

- An intruder reads messages from the sender and sends (possibly modified) versions to the recipient,
 - Succeeds if the recipient and sender don't realize his presence.

Repudiation of origin

- A false denial that an entity sent (or created) something.
 - Example: suppose a customer → a letter agreeing to pay for a product → the vendor ships the product and then demands payment → the customer denies having ordered the product and keep the unsolicited shipment without payment.
 - The customer has repudiated the origin of the letter. If the vendor cannot prove that the letter came from the customer, the attack succeeds.
 - Integrity mechanisms cope with this threat.

Denial of receipt

- A false denial that an entity received some information or message.
 - E.g. A customer orders an expensive product and pays in advance: customer pays → vendor ships. The customer then falsely asks the vendor when he will receive the product → denial of receipt attack.
 - The vendor can defend against this attack only by proving that the customer did, despite his denials, receive the product.
 - Integrity and availability mechanisms guard against these attacks.

Denial of service

- A long-term inhibition of service, so a form of usurpation, although often used with other mechanisms to deceive.
 - The attacker prevents a server from providing a service. The denial may occur at
 - the source (by preventing the server from obtaining the resources needed to perform its function),
 - at the destination (by blocking the communications from the server), or along the intermediate path (by discarding messages from either the client or the server, or both).
 - Availability mechanisms counter this threat.

Policies and Mechanisms

- Policy says what is, and is not, allowed
 - This defines “security” for the site/system/etc.
- Mechanisms enforce policies
- Composition of policies
 - If policies conflict, discrepancies may create security vulnerabilities

Policies and Mechanisms

- Policy: may be expressed in
 - natural language, which is usually imprecise but easy to understand;
 - mathematics, which is usually precise but hard to understand;
 - policy languages, which look like some form of programming language and try to balance precision with ease of understanding

Policies and Mechanisms

- Mechanisms: may be
 - technical, in which controls in the computer enforce the policy, e.g. a user has to supply a password to authenticate herself before using
 - procedural, in which controls outside the system enforce the policy; e.g. , firing someone for bringing in a game disk from an untrusted source
- The composition problem requires checking for inconsistencies among policies

Goals of Security

- Prevention
 - Prevent attackers from violating security policy
- Detection
 - Detect attackers' violation of security policy
- Recovery
 - Stop attack, assess and repair damage
 - Continue to function correctly even if attack succeeds

Assurance

- Assurance is a measure of how well the system meets its requirements; i.e. how much you can trust the system to do what it is supposed to do.
- Assurance techniques:
 - Specification
 - Design
 - Implementation

Specification

■ Specification

- Arise from Requirements analysis
- Statement of desired functionality: says what the system must do to meet those requirements. Can be
 - very formal (mathematical) or informal (natural language)
 - high-level or low-level
 - E.g. describing what the system as a whole is to do vs. what specific modules of code are to do

Design and Implementation

- Design: How to meet specification
 - Typically, the design is layered by breaking the system into abstractions, and then refining the abstractions (work down to the hardware).
 - An analyst must show the design matches the specification.
- Implementation
 - Actual coding of the modules and software components.
 - These must be correct (perform as specified), and their aggregation must satisfy the design.

Operational Issues

- Cost-Benefit Analysis
 - Is it cheaper to prevent or recover?
- Risk Analysis
 - Should we protect something?
 - How much should we protect this thing?
- Laws and Customs
 - Are desired security measures illegal?
 - Will people do them?

Human Issues

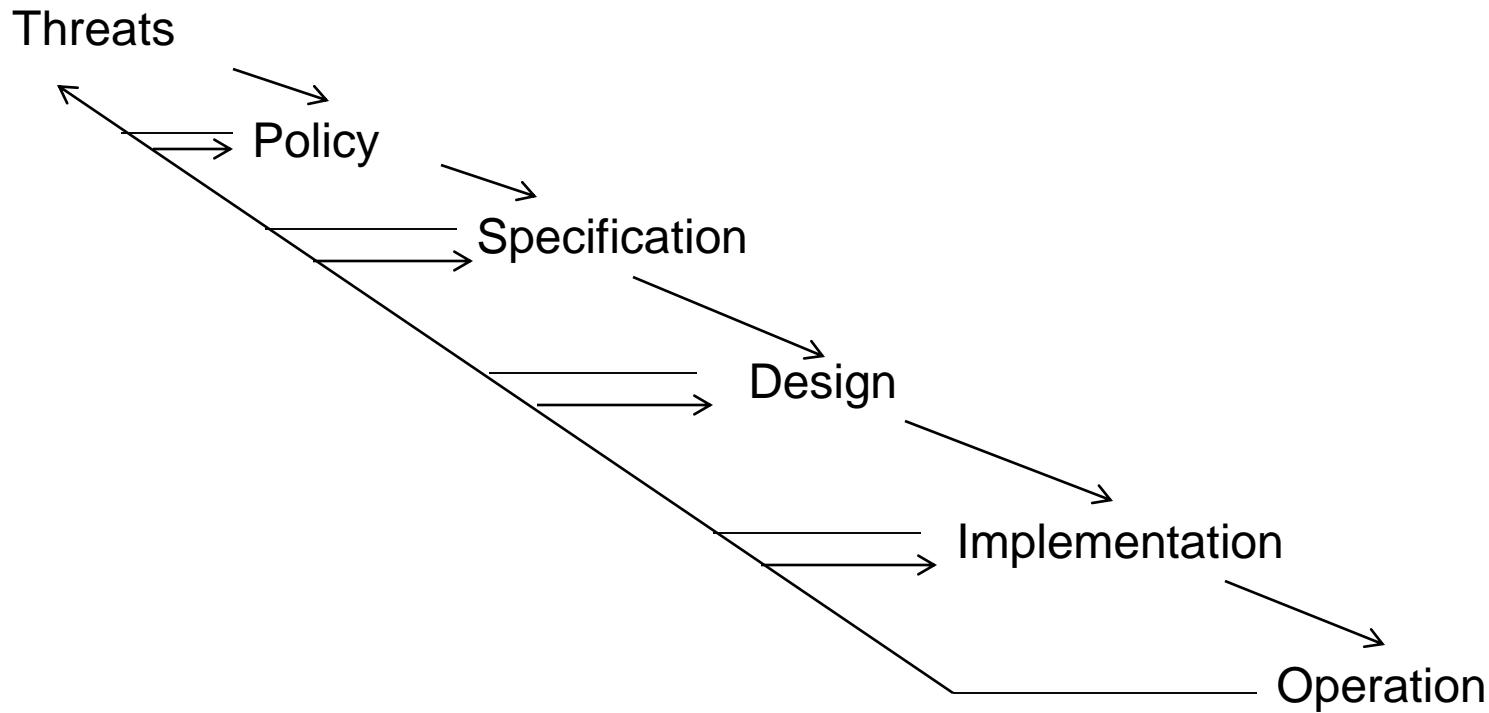
■ Organizational Problems

- Power and responsibility
 - those responsible for security have the power to enforce security (not responsibility without power or vice versa)
- Financial benefits
 - Tricky: security is not a direct financial incentive, only appreciated when loss occurs

Human Issues

- People problems
 - Outsiders and insiders
 - Social engineering

The security lifecycle



EXAMPLE

- A major corporation decided to improve its security.
 - Hired consultants → determined the threats → created a policy → derived specifications that the security mechanisms had to meet → developed a design that would meet the specifications.
- During the implementation phase
 - discovered [modems to the telephones] → firewall → the design had to be modified to divide systems into two classes: outside or behind the firewall

EXAMPLE

- When deployed, the operation and maintenance phase revealed several unexpected threats.
 - sensitive data sent across the Internet in the clear → crypto is very difficult to use → fixed implementation
 - several "trusted" hosts (allowed to log in without authentication) were physically outside the company's control
 - This violated policy, because of commercial reasons → modified the policy element about "trusted hosts"
 - Finally, the company detected proprietary material being sent to a competitor over electronic mail.
 - This added a threat that the company had earlier discounted. The company did not realize that it needed to worry about insider attacks.

Key Points

- Policy defines security, and mechanisms enforce security
 - Confidentiality
 - Integrity
 - Availability
- Trust and knowing assumptions
- Importance of assurance
- The human factor

SECURITY ENGINEER'S

Terminologies

- Vulnerabilities (weaknesses)
- Threats (potential scenario of attack)
- Attacks
- Controls (security measures)

What the security engineer should master

- The technical foundation of security such as cryptography and related technologies, access control methods, authentication methods and other security architectures
- Knowledge and skills in some major areas, i.e. security topics in
 - Computer networks
 - Databases and data mining
 - Software and programs
 - Web and Internet services
 - Electronic privacy and e-commerce

Methods of Defense

- Prevention
- Deterrence
- Reflection
- Detection
- Recovering

Controls

- Encryption
- Software controls
- Hardware controls
- Policies and procedures
- Physical controls

Cryptography

- Cryptographers work on formal methods in building secure systems of storing and processing information and communicating between its parts, ***all under the present of adversaries.***
 - Encryption:
 - Entity authentication tools
 - Message authentication with digital signatures
 - Key management
 - and many other cryptographic tools

Access Control Methods

- Scenario: Can user X use resource F with right (privilege) R?
 - Access Control Matrix
 - Discretionary Access Control
 - Mandatory Access Control
 - Role-based Access Control

Authentication Methods

- What entity knows (eg. password)
- What entity has (eg. Identity card, smart card)
- What entity is (eg. fingerprints, retinal characteristics)
- Where entity is (eg. In front of a particular terminal)

Network security

- Popular attacks, threats in networks
- Some popular strong attacks against Internet protocols
- Available secure packages/protocol suite:
IPsec, SSL/TLS
- Firewall, Intrusion Detection Systems

Security in Software/Programs/Web

- Malicious codes: Logic bomb, virus, worms,
...
- Unintentional common errors: buffer overflow
...
- Attacks exploiting common mistakes
- Web attacks: command injection e.g. SQL
injection, cross-site scripting ...

Database security

- Database Security Levels
- Reliability and Integrity
- Attacks against database
- Access Control

Privacy, DRM and E-payment

- Some hot topics related to the many aspects of Internet services and transactions
 - Controversies around level of user privacy: can perfect anonymity be allowed?
 - How to protect copyright of digital products?
 - How can we pay securely and efficiently when doing e-commerce?

What is This Course About?

- Learn to think about security
 - Threats, defenses, policies
 - Software, human and environment factors
- Think as an attacker:
 - Learn to identify threats
- Think as a security designer:
 - Learn how to prevent attacks and/or limit their consequences
 - Understand and apply security principles
 - Learn tools that can defend against specific attacks, no silver-bullet solution

Agenda

- A gentle intro to Cryptography
- Authentication methods
- Access control methods and mechanisms
- Software and Program security

Course Material

■ Lecturer's website

<http://sedic.soict.hust.edu.vn/vannk/>

<http://sedic.soict.hust.edu.vn/vannk/AntoanThongtin/ComputerSecurity.htm>

- **Introduction to Computer Security, Matt Bishop**, Addison-Wesley Professional
- Security in Computing, Charles P. Pfleeger, Prentice Hall
- Cryptography And Network Security: Principles and Practices, William Stallings, Prentice Hall

Course Website

The screenshot shows a web browser window with two tabs open. The left tab contains a list of achievements and research interests. The right tab displays course materials for Information Security.

Achievements:

- My paper "Distributed Shortcut Networks: Layout-aware Low-degree Topology" (with my student and Prof. Michihiro Koibuchi and Dr. Iki Fujiwara from NII, Japan) was accepted at the Conference on Parallel Processing, a pretty top conference)
- Officially got the government grant (MOET Ministry) for the WiSSim project.
- My student Nguyen Trung Hieu (K52, co-authored in 2 recent papers) has been admitted to University of California (US Top-20 CS grad school). All the bests to his future academic career!
- We've got funded from IREP and Dimage-Share, two Japanese IT companies.
- We (SEDIC) are having some great students, Hieu and Nhat from K52, graduated this year (2012) to work in new projects (we are training future PhD students for Top US schools).
- SEDIC Lab (Software Engineering and Distributed Computing) gets started in January 2013.
- No longer head of SE department (since 5/2012); more time for research from now on!

Teaching

Probability for Computing (Toán Chuyên đề/Mô hình và thuật toán Internet phổ biến/Network Theory)
Algorithm Analysis and Design
Information Security (Master program)
Object-Oriented Programming and Design

Information Security (Standard/ICT/The Talented Engineer Program)
Network Security (HEDSPI)
Introduction to Cryptography (USTH)

In Vietnamese:
Hướng dẫn viết luận văn tốt nghiệp (under construction)
Một số định hướng để tài ĐATN cho sinh viên năm cuối (bảo vệ 2010)

Research Interests

- Algorithms for networking and distributed computing, high performance computing
- Networking: wireless networks. P2P

Information Security

with a gentle Intro to Applied Cryptography

Course Intro: [Slides](#)

	Standard (Third year)	For ICT	For Talented Eng (KSTN)	New versions (more in Crypto)	Lecture Notes In Vietnamese
Intro to Crypto - Basic notions and classic ciphers	Slide	Slide Lec-1	Slide	New version	GT-Chuong
Symmetric Key Cryptography and Block Ciphers Operations	Slide	Slide Lec-2	Slide	New slides	GT-Chuong
Public Key Cryptography	Slide Slide	Slide Lec-3	Slide Slide	New slides	GT-Chuong
Key Management	Slide	Slide Lec-4	Slide		
Authentication	Slide	Slide	Slide		GT-Chuong
Access Control	Slide	Slide	Slide		GT-Chuong
Program Security	Slide	Slide	Slide		
Network Security	Slide		Slide		
Cryptographic Protocols			Slide-PartI Slide-PartII		
DoS Attack	Slide		Slide		

References

[Introduction to Computer Security](#). OR

Group Project

- Objectives
- The how-to-dos in:
 - Choosing topic
 - Doing survey and/or software product
 - Producing Write-up report
- Evaluation method

Cryptography II

Block ciphers and modes of operations

Review

■ Security Goals

- Confidentiality (secrecy, privacy)
 - Assure that data is accessible to only one who are authorized to know
- Integrity
 - Assure that data is only modified by authorized parties and in authorized ways
- Availability
 - Assure that resource is available for authorized users

Review

- Secret-key cryptography
 - symmetric cryptography
 - same key for both encryption & decryption ($Z=Z'$)
- Public-key cryptography
 - asymmetric cryptography
 - encryption key different from decryption key and

Review

■ Shift cipher (additive cipher)

- Key Space: [1 .. 25]
- Encryption given a key K:
 - $Y = X + K \pmod{26}$
- Decryption given K:
 - $X = Y - K \pmod{26}$
- Cryptanalysis
 - exhaustive search (≤ 26 possible keys).

Review

■ Mono-alphabetical Substitution Cipher

- The key space: all permutations of $\Sigma = \{A, B, C, \dots, Z\}$
- Encryption given a key π :
 - each letter X in the plaintext P is replaced with $\pi(X)$
- Decryption given a key π :
 - each letter Y in the ciphertext P is replaced with $\pi^{-1}(Y)$
- Cryptanalysis
 - frequency analysis attacks

Review

■ Polyalphabetic Substitution Ciphers (Vigenère cipher - published in 1586)

□ Definition:

- Given m , a positive integer, $P = C = (\mathbb{Z}_{26})^n$, and $K = (k_1, k_2, \dots, k_m)$ a key, we define:

□ Encryption:

- $e_k(p_1, p_2, \dots, p_m) = (p_1+k_1, p_2+k_2, \dots, p_m+k_m) \pmod{26}$

□ Decryption:

- $d_k(c_1, c_2, \dots, c_m) = (c_1-k_1, c_2-k_2, \dots, c_m-k_m) \pmod{26}$

□ Example:

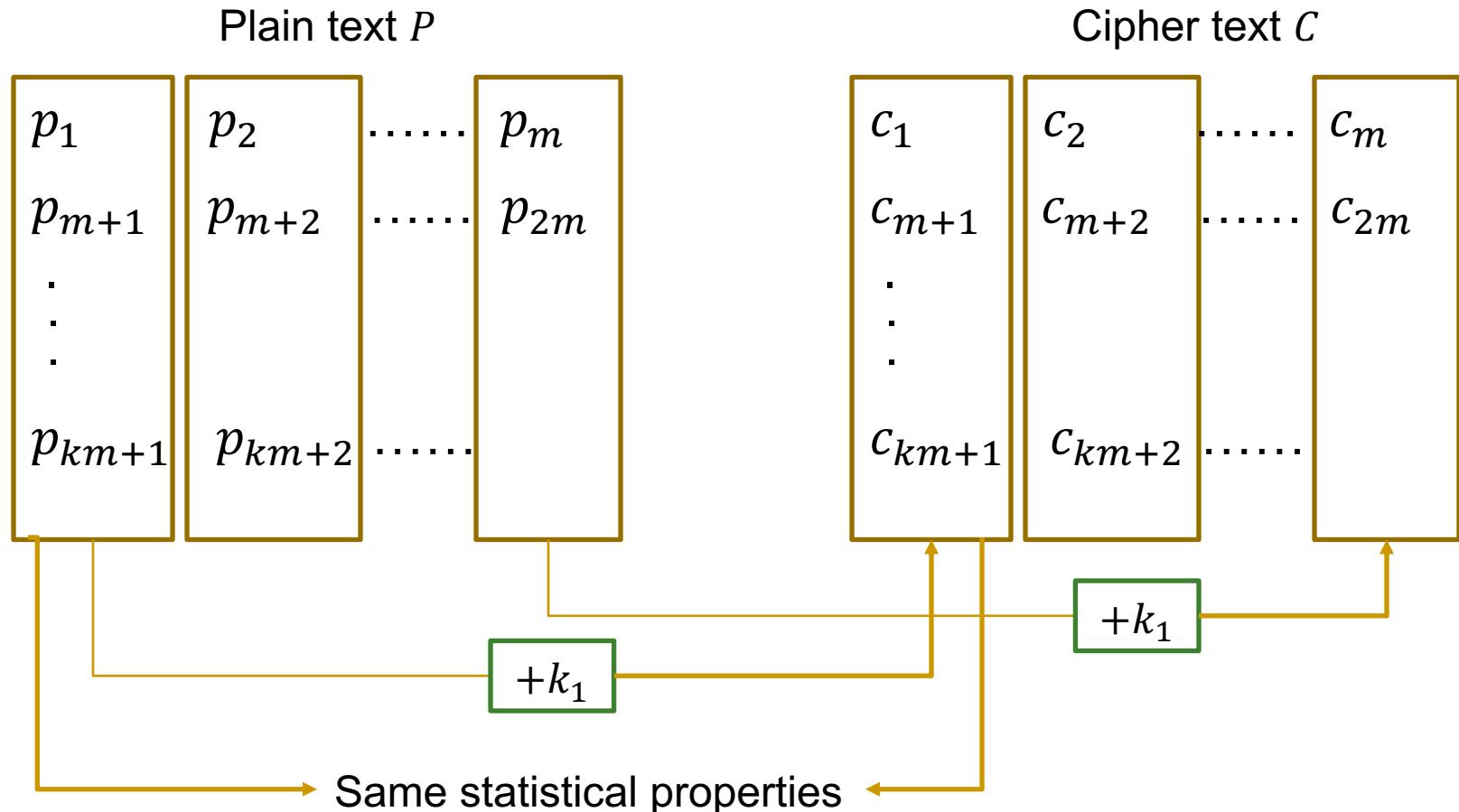
Plaintext: C R Y P T O G R A P H Y

Key: L U C K L U C K L U C K

Ciphertext: N L A Z E I I B L J J I

Vigenère cipher

Cryptanalysis



Can be broken by the statistical method once the key length is determined

Vigenère cipher

- How to determine the key length
 - The frequency of letters in $\{p_j, p_{m+j}, \dots, p_{km+j}\}$ is approximately the same as that in the plain text P
 - The frequency of letters in $\{c_j, c_{m+j}, \dots, c_{km+j}\}$ is the same as that in $\{p_j, p_{m+j}, \dots, p_{km+j}\}$
- The index of coincidence (IC)
 - Suppose $x = x_1x_2 \dots x_n$ is a string of alphabetic characters $\rightarrow IC(x)$ is the probability that two random elements of x are identical

Vigenère cipher

- The index of coincidence (IC)
 - Suppose the frequencies of A, B, \dots, Z in x are $f_0, f_1 \dots, f_{25}$
 - $IC(x) = \frac{\sum_{i=0}^{25} \binom{f_i}{2}}{\binom{n}{2}} = \sum_{i=0}^{25} \frac{f_i}{n} \frac{f_i - 1}{n - 1} \approx \sum_{i=0}^{25} (p_i)^2$

letter	probability
A	.082
B	.015
C	.028
D	.043
E	.127
F	.022
...	
Z	.001



For an English text
 $IC(x) \approx 0.065$

p_i : the frequency of the i-th letter

For a totally random string
 $IC(x) \approx \sum_{i=0}^{25} \frac{1}{26} = 0.038$

Vigenère cipher

- The index of coincidence (IC)
 - Let $P_j = \{p_j, p_{m+j}, \dots, p_{km+j}\}$; $C_j = \{c_j, c_{m+j}, \dots, c_{km+j}\}$
 - $IC(C_j) = IC(P_j) \approx 0.065$
- Cryptanalysis algorithm
 1. Set $m = 1$
 2. Check if m is indeed the key length
 - Divide the cipher into m letter group and compute the IC of each
 - If they are quite the same and approximately equals to 0.065 then m is the key length
 - If they are quite different and smaller than 0.065, then the key length should be greater
 3. Increase m by 1 and go to step 1

Vigenère cipher

- Kasiski method: a hint to find the key length
 - Observation: two identical segments of plaintext will be encrypted to the same cipher text wherever their occurrence in the plain text is δ position apart, $\delta \equiv 0(\text{mod } m)$

$p_1 \dots$ $p_j p_{j+1} \dots p_{j+u}$ p_m

$c_1 \dots$ $c_j c_{j+1} \dots c_{j+u}$ c_m

$p_{vm+1} \dots$ $p_{vm+j} p_{vm+j+1} \dots p_{vmj+u}$ p_{vm+m}

$c_{vm+1} \dots$ $c_{vm+j} c_{vm+j+1} \dots c_{vmj+u}$ c_{vm+m}

If these are the same

Then, these will be the same

Vigenère cipher

■ Kasiski method

- Search the cipher text for pairs of identical segments and record the distance between their starting positions
 - Suppose the obtained distances are $\delta_1, \dots, \delta_k$
- Then, m should divides the greatest common divisor of $\delta_1, \dots, \delta_k$

Vigenère cipher

■ Example

CHREEVOAHMAERATBIAXXWTNXBEEOPHBSBQMQEQRBW
RVXUOAKXAOSXXWEAHBWGJMMQMNKGRFVGXWTRZXWIAK
LXFPSKAUTEMNDCMGTSXMXBTUIADNGMGPSRELXNJELX
VRVPRTULHDNQWTWDTYGBPHXTFALJHASVBFXNGLLCHR
ZBWELEKMSJIKNBHWRJGNMGJSGLXFEYPHAGNRBIEQJT
AMRVLCRREMNDGLXRRIMGNSNRWCHRQHAEYEVTQAQEBCBI
PEEWEVKAKOEWADEMXTBHHCHRTKDNRZCHRCILOHP
WQAIIWXRNGWOIIIFKEE

Vigenère cipher

■ Example

CHR EEVOAHMAERATBIAXXWTNXBEEOPHBSBQMQEQRBW
RVXUOAKXAOSXXWEAHBWGJMMQMNKGRFVGXWTRZXWIAK
LXFPSKAUTEMNDCMGTSXMXBUIADNGMGPSRELXNJELX
VRVPRTULHDNQWTWDTYGBPHXTFALJHASVBFXNGLL CHR
ZBWELEKMSJIKNBHWRJGNMGJSGLXFEYPHAGNRBIEQJT
AMRVLCRREMNDGLXRRIMGNSNRW CHR QHAEYEVTAQEBBI
PEEWEVKAKOEWADREMXTBHH CHR TKDNVRZ CHR CLQOHP
WQAIIXNRMGWQIIFKEE

Kasiski method: CHR's occurrence positions: 1, 166, 236, 276 and 286
→ Distances: 165, 235, 275 and 285
→ $\text{Gcd}(165, 235, 275, 285) = 5$
→ The key length should divides 5

Vigenère cipher

■ Example

CHR EEVOAHMAERATBIAXXWTNXBEEOPHBSBQMQEQRBW
RVXUOAKXAOSXXWEAHBWGJMMQMNKGRFVGXWTRZXWIAK
LXFPSKAUTEMNDCMGTSXMXBTUIADNGMGPSRELXNJELX
VRVPRTULHDNQWTWDTYGBPHXTFALJHASVBFXNGLL CHR
ZBWELEKMSJIKNBHWRJGNMGJSGLXFEYPHAGNRBIEQJT
AMRVLCRREMNDGLXRRIMGNSNRW CHR QHAEYEVTAQEBBI
PEEWEVKAKOEWADREMXTBHH CHR TKDNVRZ CHR CLQOHP
WQAIIXWXNRMGWQIIFKEE

Confirmation of Kasiski method

$$M = 1 \rightarrow IC = 0.045$$

$$M = 2 \rightarrow IC_s = 0.046 \text{ and } 0.041$$

$$M = 3 \rightarrow IC_s = 0.043, 0.050, 0.047$$

$$M = 4 \rightarrow IC_s = 0.042, 0.039, 0.046, 0.040$$

$$M = 5 \rightarrow IC_s = 0.063, 0.068, 0.069, 0.061 \text{ and } 0.072$$

Vigenère cipher

i	value of $M_g(\mathbf{y}_i)$								
1	.035	.031	.036	.037	.035	.039	.028	.028	.048
	.061	.039	.032	.040	.038	.038	.045	.036	.030
	.042	.043	.036	.033	.049	.043	.042	.036	
2	.069	.044	.032	.035	.044	.034	.036	.033	.029
	.031	.042	.045	.040	.045	.046	.042	.037	.032
	.034	.037	.032	.034	.043	.032	.026	.047	
3	.048	.029	.042	.043	.044	.034	.038	.035	.032
	.049	.035	.031	.035	.066	.035	.038	.036	.045
	.027	.035	.034	.034	.036	.035	.046	.040	
4	.045	.032	.033	.038	.060	.034	.034	.034	.050
	.033	.033	.043	.040	.033	.029	.036	.040	.044
	.037	.050	.034	.034	.039	.044	.038	.035	
5	.034	.031	.035	.044	.047	.037	.043	.038	.042
	.037	.033	.032	.036	.037	.036	.045	.032	.029
	.044	.072	.037	.027	.031	.048	.036	.037	



$$K = (9, 0, 13, 4, 19) = JANET$$

$$M_g = \sum_{i=0}^{25} \frac{p_i f_{i+g}}{n'}$$

If $g \neq k_i$, then $M_g \ll 0.065$

The almond tree was in tentative blossom. The days were longer, often ending with magnificent evenings of corrugated pink skies. The hunting season was over, with hounds and guns put away for six months. The vineyards were busy again as the well-organized farmers treated their vines and the more lackadaisical neighbors hurried to do the pruning they should have done in November.

Exercises

■ Decode the following cipher texts

- Encrypted by shift cipher:
 - JBCRCLQRWCRVNBJENBWRWN
- Encrypted by substitution cipher:
 - Pjmu mu b amtjfo rfsr. Mr jbu cffi fiaowtrfg cw rjf uvcurmrvrmqi amtjfo. Wqv bof xfow nvahw. Rjf amtjfo jbu cffi coqhfi
 - **YIFQFMZRWQFYVECFMDZPCVMRZWNMDZVEJBTXCDDUMJ
NDIFEFDZCDMQZKCEYFCJMYRNCWJCSREXCHZUNMXZ
NZUCDRJXYYSMRTMEYIFZWDYVZVFZUMRZCRWNZDZJJ
XZWGCHSMRNMDHNCMFQCHZJMXJZWIEJYUCFWDJNZDIR**
 - Hints:
 - The letters in the English alphabet can be divided into 5 groups of similar frequencies
 - e
 - t,a,o,i,n,s,h,r
 - d,l
 - c,u,m,w,f,g,y,p,b
 - v,k,j,x,q,z
 - Some frequently appearing bigrams or trigrams
 - Th, he, in, an, re, ed, on, es, st, en at, to
 - The, ing, and, hex, ent, tha, nth, was eth, for, dth.

Exercises

- Decode the following cipher texts
 - Encrypted by substitution cipher:

YIFQFMZRQFYVECFMDZPCVMRZWNMDZVEJBTXCDDUMJ
NDIFEFDMDZCDMQZKCEYFCJMYRNCWJCSZREXCHZUNMXZ
NZUCDRJXXYSMRTMEYIFZWDYVZVFZUMRZCRWNZDZJJ
XZWGCHSMRNMDHNCMFQCHZJMXJZWIEJYUCFWDJNZDIR

letter	frequency	letter	frequency
A	0	N	9
B	1	O	0
C	15	P	1
D	13	Q	4
E	7	R	10
F	11	S	3
G	1	T	2
H	4	U	5
I	5	V	5
J	11	W	8
K	1	X	6
L	0	Y	10
M	16	Z	20

DZ and ZW: four times each

NZ and ZU: three times each

RZ, HZ, YZ, FZ, ZR, ZV, ZC, ZD, ZJ: twice each

Cryptography I

General concepts and some classical ciphers

- Basic concepts
- Attack models
- Classic ciphers: mono-alphabetic
- Vigenere cipher
- One-time-pad cipher

Security Goals

- Confidentiality (secrecy, privacy)
 - Assure that data is accessible to only one who are authorized to know
- Integrity
 - Assure that data is only modified by authorized parties and in authorized ways
- Availability
 - Assure that resource is available for authorized users

General tools

- Cryptography
- Software controls
- Hardware controls
- Policies and procedures
- Physical controls

What is Crypto?

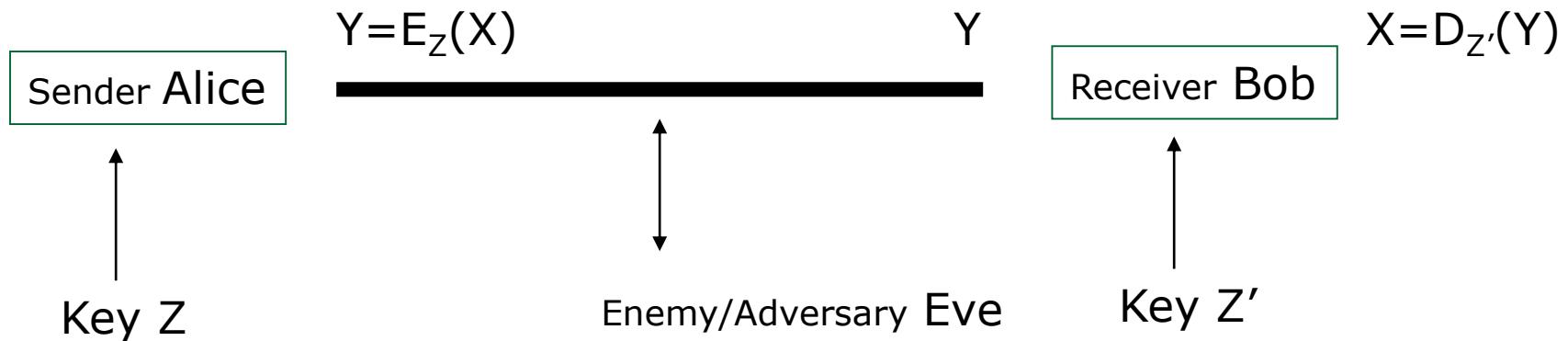
- Constructing and analyzing **cryptographic protocols** which enable parties to achieve security objectives
 - Under the present of adversaries.
- A protocol (or a scheme) is a suite of procedures that tell each party what to do
 - usually, computer algorithms
- Cryptographers devise and analyze protocols under **Attack model**
 - assumptions about the resources and actions available to the adversary
 - So, you need to think as an adversary

Terms

- **Cryptography:** the study of mathematical techniques for providing information security services.
- **Cryptanalysis:** the study of mathematical techniques for attempting to get security services breakdown.
- **Cryptology:** the study of cryptography and cryptanalysis.

Terms ...

- plaintexts
- ciphertexts
- keys
- encryption
- decryption



Secret-key cryptography

- Also called: symmetric cryptography
- Use the same key for both encryption & decryption ($Z=Z'$)
- Key must be kept secret
- Key distribution – how to share a secret between A and B very difficult

Public-key cryptography

- Also called: asymmetric cryptography
- Encryption key different from decryption key and
 - It is not possible to derive decryption key from encryption key
- Higher cost than symmetric cryptography

Is it a secure cipher system?

■ Why insecure

- just break it under a certain reasonable attack model
(show failures to assure security goals)

■ Why secure:

- Evaluate/prove that under the considered attack model,
security goals are assured
- Provable security: Formally show that (with mathematical
techniques) the system is as secure as a well-known secure
one (usually simpler).

Breaking ciphers ...

- There are different methods of breaking a cipher, depending on:
 - the type of information available to the attacker
 - the interaction with the cipher machine
 - the computational power available to the attacker

Breaking ciphers ...

■ Ciphertext-only attack:

- The cryptanalyst knows **only the ciphertext**.
- Goal: to find the plaintext and the key.
- NOTE: such vulnerable is seen completely insecure

■ Known-plaintext attack:

- The cryptanalyst knows **one or several pairs of ciphertext and the corresponding plaintext**.
- Goal: to find the key used to encrypt these messages
 - or a way to decrypt any new messages that use the same key (although may not know the key).

Breaking ciphers ...

■ Chosen-plaintext attack

- The cryptanalyst **can choose a number of messages and obtain the ciphertexts for them**
- Goal: deduce the key used in the other encrypted messages or decrypt any new messages (using that key).

■ Chosen-ciphertext attack

- Similar to above, but the cryptanalyst **can choose a number of ciphertexts and obtain the plaintexts.**

■ Both can be **adaptive**

- The choice of ciphertext may depend on the plaintext received from previous requests.

Models for Evaluating Security

- **Unconditional (information-theoretic) security**
 - Assumes that the adversary has unlimited computational resources.
 - Plaintext and ciphertext modeled by their distribution
 - Analysis is made by using probability theory.
 - For encryption systems: **perfect secrecy**, observation of the ciphertext provides no information to an adversary.

Models for Evaluating Security

■ **Provable security:**

- Prove security properties based on assumptions that it is difficult to solve a well-known and supposedly difficult problem (NP-hard ...)
 - E.g.: computation of discrete logarithms, factoring

■ **Computational security (practical security)**

- Measures the amount of computational effort required to defeat a system using the best-known attacks.
- Sometimes related to the hard problems, but no proof of equivalence is known.

Models for Evaluating Security

■ Ad hoc security (heuristic security):

- Variety of convincing arguments that every successful attack requires more resources than the ones available to an attacker.
- Unforeseen attacks remain a threat.
- **THIS IS NOT A PROOF**

Classic ciphers

Shift cipher (additive cipher)

- Key Space: [1 .. 25]
- Encryption given a key K:
 - each letter in the plaintext P is replaced with the K'th letter following corresponding number (shift right):
 - Another way: $Y = X \oplus K \rightarrow$ additive cipher
- Decryption given K:
 - shift left

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

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

P = CRYPTOGRAPHYISFUN

K = 11

C = NCJAVZRCLASJTDQFY

Shift Cipher: Cryptanalysis

- Easy, just do exhaustive search
 - key space is small (≤ 26 possible keys).
 - once K is found, very easy to decrypt

General Mono-alphabetical Substitution Cipher

- The key space: all permutations of $\Sigma = \{A, B, C, \dots, Z\}$
- Encryption given a key π :
 - each letter X in the plaintext P is replaced with $\pi(X)$
- Decryption given a key π :
 - each letter Y in the cipherext P is replaced with $\pi^{-1}(Y)$
- **Example:**

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

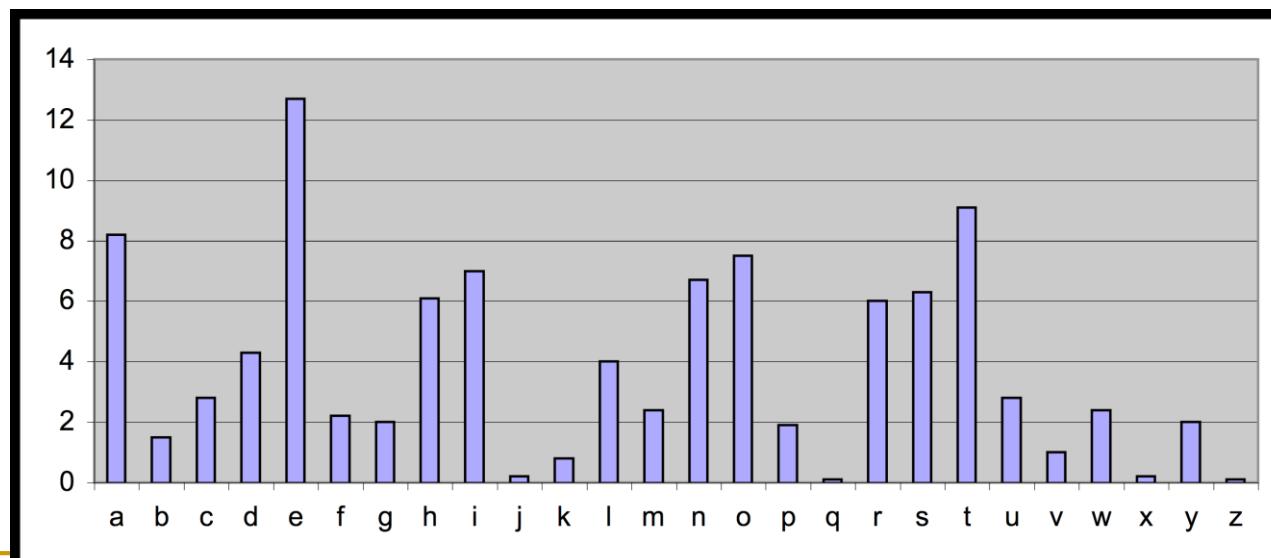
BECAUSE → AZDBJSZ

Looks secure, early days

- Exhaustive search is infeasible
 - key space size is $26! \approx 4*10^{26}$
- Dominates the art of secret writing throughout the first millennium A.D.
- Thought to be unbreakable by many back then

Cryptanalysis of Substitution Ciphers: Frequency Analysis

- Each language has certain features:
 - frequency of letters, or of groups of two or more letters.
- Substitution ciphers preserve the mentioned language features → vulnerable to frequency analysis attacks



Substitution Ciphers: Cryptanalysis

- The number of different ciphertext characters or combinations are counted to determine the frequency of usage.
- The cipher text is examined for patterns, repeated series, and common combinations.
- Replace ciphertext characters with possible plaintext equivalents using known language characteristics.
- Example:

THIS IS A PROPER SAMPLE FOR ENGLISH TEXT. THE FREQUENCIES OF LETTERS IN THIS SAMPLE IS NOT UNIFORM AND VARY FOR DIFFERENT CHARACTERS. IN GENERAL THE MOST FREQUENT LETTER IS E FOLLOWED BY A SECOND GROUP. IF WE TAKE A CLOSER LOOK WE WILL NOTICE THAT FOR BIGRAMS AND TRIGRAMS THE NONUNIFORM IS EVEN MORE.

- Observations: $f_x=1$ và $f_A=15$.

- The letters in the English alphabet can be divided into 5 groups of similar frequencies

I e

II t,a,o,i,n,s,h,r

III d,l

VI c,u,m,w,f,g,y,p,b

V v,k,j,x,q,z

- Some frequently appearing bigrams or trigrams

Th, he, in, an, re, ed, on, es, st, en at, to

The, ing, and, hex, ent, tha, nth, was eth, for, dth.

Example

<i>Letter:</i>	A	B	C	D	E	F	G	
<i>Frequency:</i>	5	24	19	23	12	7	0	■ $e \Rightarrow Z$
<i>Letter:</i>	H	I	J	K	L	M	N	$f_j = 29, f_v = 27$
<i>Frequency:</i>	24	21	29	6	21	1	3	$f_{jcz} = 8 \rightarrow t \Rightarrow J$
<i>Letter:</i>	O	P	Q	R	S	T	U	$h \Rightarrow C$
<i>Frequency:</i>	0	3	1	11	14	8	0	■ $a \Rightarrow V$
<i>Letter:</i>	V	W	X	Y	Z			(article a)
<i>Frequency:</i>	27	5	17	12	45			

$J, V, B, H, D, I, L, C \{t, a, o, i, n, s, h, r\}$

$t, a \qquad \qquad h$

$JZB = te ? \{ teo, tei, ten, ter, tes \} \rightarrow n \Rightarrow B$

- Observations:
 - A cipher system should not allow statistical properties of plaintext to pass to the ciphertext.
 - The ciphertext generated by a "good" cipher system should be statistically indistinguishable from random text.
- Idea for a stronger cipher (1460's by Alberti)
 - use more than one cipher alphabet, and switch between them when encrypting different letters → Poly-alphabetic Substitution Ciphers
 - Developed into a practical cipher by Vigenère (published in 1586)

■ **Definition:**

- Given m , a positive integer, $P = C = (Z_{26})^n$, and $K = (k_1, k_2, \dots, k_m)$ a key, we define:

■ **Encryption:**

$$e_k(p_1, p_2 \dots p_m) = (p_1+k_1, p_2+k_2 \dots p_m+k_m) \pmod{26}$$

■ **Decryption:**

$$d_k(c_1, c_2 \dots c_m) = (c_1-k_1, c_2-k_2 \dots c_m-k_m) \pmod{26}$$

■ **Example:**

Plaintext: C R Y P T O G R A P H Y

Key: L U C K L U C K L U C K

Ciphertext: N L A Z E I I B L J J I

Vigenere Cipher: Cryptanalysis

- Find the length of the key.
- Divide the message into that many shift cipher encryptions.
- Use frequency analysis to solve the resulting shift ciphers.

One-Time Pad

Key is chosen randomly

Plaintext $X = (x_1 \ x_2 \ \dots \ x_n)$

Key $K = (k_1 \ k_2 \ \dots \ k_n)$

Ciphertext $Y = (y_1 \ y_2 \ \dots \ y_n)$

$$e_k(X) = (x_1 + k_1 \ x_2 + k_2 \ \dots \ x_n + k_n) \text{ mod } m$$

$$d_k(Y) = (y_1 - k_1 \ y_2 - k_2 \ \dots \ y_n - k_n) \text{ mod } m$$

Example

Plaintext space = Ciphertext space =

Keyspace = $\{0,1\}^n$

Key is chosen randomly

For example:

Plaintext is 10001011

Key is 00111001

Then ciphertext is 10110010

Main points in One-Time Pad

- The key is never to be reused
 - Thrown away after first and only use
 - If reused → insecure!
- One-Time Pad uses a very long key, exactly the same length as of the plaintext
 - In old days, some suggest choose the key as texts from, e.g., a book → i.e. not **randomly chosen**
 - Not One-Time Pad anymore → this does not have perfect secrecy as in true One-Time-Pad and can be broken
 - Perfect secrecy means key length be at least message length
 - **Difficult in practice!**

- Shift ciphers are easy to break using brute force attacks (exhaustive key search)
- Substitution ciphers preserve language features (in N-gram frequency) and are vulnerable to frequency analysis attacks.
- Vigenère cipher are also vulnerable to frequency analysis once the key length is found.
 - In general poly-alphabetical substitution ciphers are not that secure
- OTP has perfect secrecy if the key is chosen randomly in the message length and is used only once.

<http://pc.vietica.com/van.nguyen/InfoSec-VietNhat/InfoSec.htm>

Block cipher

Mật mã khối

Giảng viên: Nguyễn Phi Lê

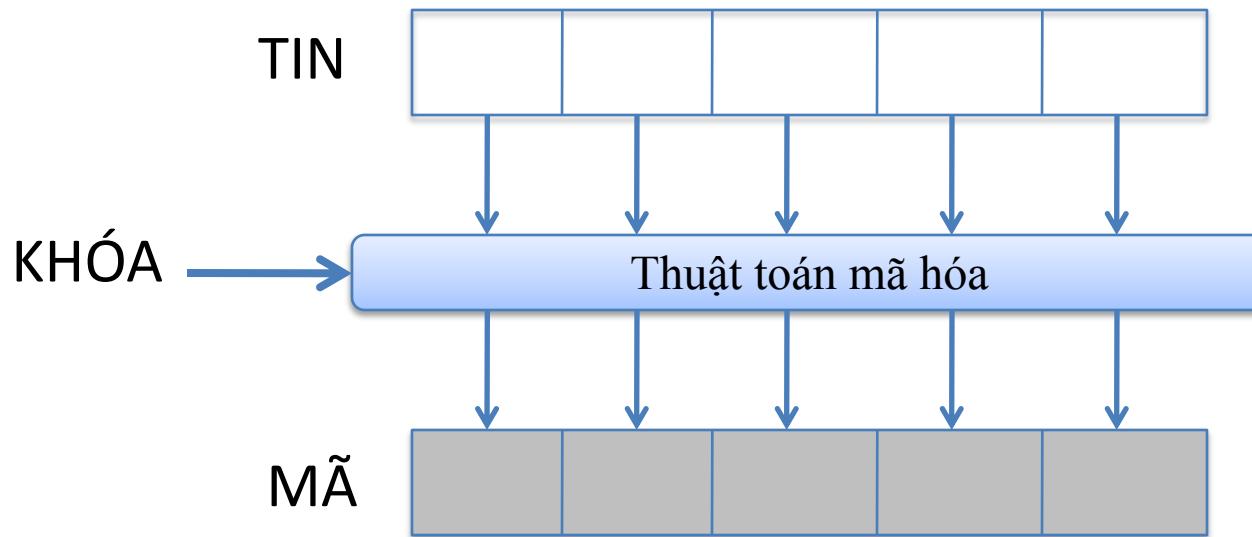
Mục lục

- Khái niệm
- Nguyên tắc thiết kế
 - Cấu trúc Feistel
- Mật mã DES

Overview (1/2)

- definition

- Mật mã khối là mật mã mà trong đó bản tin (plain text) được chia thành các khối có độ dài bằng nhau. Các khối này được mã hóa riêng biệt thành các khối mã có độ dài bằng khối đầu vào



Ví dụ

input key \	000	001	010	011	100	101	110	111
0	001	111	110	000	100	010	101	011
1	001	110	111	100	011	010	000	101
2	001	000	100	101	110	111	010	011
3	100	101	110	111	000	001	010	011
4	101	110	100	010	011	001	011	111

KHÓA : 1

TIN: 010100110111



TIN: (010) (100) (110) (111)

Ví dụ

input key \	000	001	010	011	100	101	110	111
0	001	111	110	000	100	010	101	011
1	001	110	111	100	011	010	000	101
2	001	000	100	101	110	111	010	011
3	100	101	110	111	000	001	010	011
4	101	110	100	010	011	001	011	111

KHÓA : 1

TIN: 010100110111



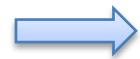
TIN: (010) (100) (110) (111)

Ví dụ

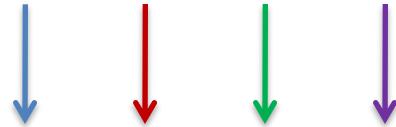
input key \	000	001	010	011	100	101	110	111
0	001	111	110	000	100	010	101	011
1	001	110	111	100	011	010	000	101
2	001	000	100	101	110	111	010	011
3	100	101	110	111	000	001	010	011
4	101	110	100	010	011	001	011	111

KHÓA : 1

TIN: 010100110111



TIN: (010) (100) (110) (111)



MÃ: (111) (011) (000) (000)

Ví dụ

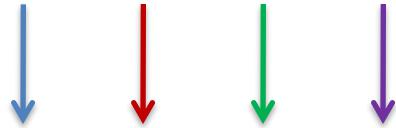
input key \	000	001	010	011	100	101	110	111
0	001	111	110	000	100	010	101	011
1	001	110	111	100	011	010	000	101
2	001	000	100	101	110	111	010	011
3	100	101	110	111	000	001	010	011
4	101	110	100	010	011	001	011	111

KHÓA : 1

TIN: 010100110111



TIN: (010) (100) (110) (111)



MÃ: (111) (011) (000) (000)

KHÓA : 3

TIN: 010100110111

Ví dụ

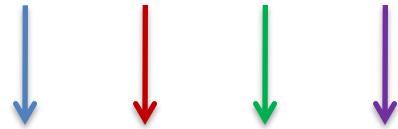
input key \	000	001	010	011	100	101	110	111
0	001	111	110	000	100	010	101	011
2	001	000	100	101	110	111	010	011
3	100	101	110	111	000	001	010	011
4	101	110	100	010	011	001	011	111

KHÓA : 1

TIN: 010100110111



TIN: (010) (100) (110) (111)



MÃ: (111) (011) (000) (000)

KHÓA : 3

TIN: 010100110111



MÃ: (110) (000) (100) (011)

Ví dụ

input key \	000	001	010	011	100	101	110	111
0	001	111	110	000	100	010	101	011
1	001	110	111	100	011	010	000	101
2	001	000	100	101	110	111	010	011
3	100	101	110	111	000	001	010	011
4	101	110	100	010	011	001	011	111

MÃ : 001

Ví dụ

input key \	000	001	010	011	100	101	110	111
0	001	111	110	000	100	010	101	011
1	001	110	111	100	011	010	000	101
2	001	000	100	101	110	111	010	011
3	100	101	110	111	000	001	010	011
4	101	110	100	010	011	001	011	111

MÃ : 001  TIN: (000) Hoặc (101)

Khái niêm (2/2)

- Điều kiện an toàn
 - Kích thước khối phải đủ lớn
 - Để chống lại tấn công theo kiểu thống kê
 - Tuy nhiên lại gây ra độ trễ lớn
 - Không gian khóa phải đủ lớn
 - Để chống lại tìm kiếm vét cạn
 - Tuy nhiên lại gây ra khó khăn trong lưu trữ, phân phôi,
 - ...

Nguyên tắc thiết kế

- Khuếch tán (diffusion)
- Hỗn loạn (Confusion)
 - Đề xuất bởi Shanon, năm 1945

Nguyên tắc thiết kế

- Khuếch tán (diffusion)
- Hỗn loạn (Confusion)

Nguyên tắc thiết kế

- **Khuếch tán (diffusion):**
 - Khuếch tán đặc tính thống kê của bản tin vào bản mã
 - Mỗi bit của bản tin và khóa phải ảnh hưởng lên nhiều bit của bản mã
 - ↔ Mỗi bit của bản mã bị ảnh hưởng bởi nhiều bit của bản tin
 - ⇒ gây khó khăn trong việc phá mã dựa trên đặc tính thống kê của bản tin

Nguyên tắc thiết kế

- **Khuếch tán (diffusion):**
 - Tin: $T = t_1t_2\dots$
 - Mã: $M_n = \left(\sum_{i=1}^k t_{n+i} \right) \bmod 26$

⇒ tần suất xuất hiện của các ký tự trong bản mã “gần giống nhau” hơn trong bản tin.

⇒ đặc tính về thống kê của các ký tự trong bản tin đã bị làm khuếch tán đi trong bản mã.

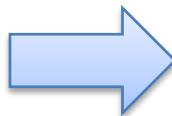
Nguyên tắc thiết kế

- **Khuếch tán (diffusion):**
 - Phương pháp thực hiện: Thực hiện nhiều lần thao tác: hoán vị + tác động thuật toán
⇒ các bit tại các vị trí khác nhau của bản tin cùng tác động lên một bit của bản mã

Nguyên tắc thiết kế

- **Khuếch tán (diffusion):**
 - Bản tin = “computer security”
 - Tạo hoán vị bằng cách viết tin thành các hàng độ dài 5, mã lấy theo cột từ phải sang trái

c	o	m	p	u
t	e	r	s	e
c	u	r	i	t
y				



uetpsimrroeuctcy
computersecurity



xtgfnrcjkthpucwx

Nguyên tắc thiết kế

- **Hỗn loạn (confusion):**
 - Sự phụ thuộc của bản mã đối với bản tin phải càng phức tạp càng tốt
 - ⇒ gây rắc rối, hỗn loạn đối với kẻ thù có ý định tìm quy luật phá mã
 - ⇒ mối quan hệ này tốt nhất là “phi tuyế”

Nguyên tắc thiết kế

- **Hỗn loạn (confusion):**

- Tin: $T = t_1 t_2$

- Mã: $M = m_1 m_2$

$$\begin{cases} m_1 = k_{11}t_1 + k_{12}t_2 \\ m_2 = k_{21}t_1 + k_{22}t_2 \end{cases}$$

$$T' = t'_1 t'_2$$

$$M' = m'_1 m'_2$$

$$\begin{cases} m'_1 = k_{11}t'_1 + k_{12}t'_2 \\ m'_2 = k_{21}t'_1 + k_{22}t'_2 \end{cases}$$



$$\begin{cases} m_1 = k_{11}t_1 + k_{12}t_2 \\ m'_1 = k_{11}t'_1 + k_{12}t'_2 \end{cases}$$

$$\begin{cases} m_2 = k_{21}t_1 + k_{22}t_2 \\ m'_2 = k_{21}t'_1 + k_{22}t'_2 \end{cases}$$

Nguyên tắc thiết kế

- **Hỗn loạn (confusion):**
 - Phương pháp thực hiện: Thực hiện bằng các thuật toán thay thế phức tạp

Mạng SPN

- Substitution and permutation network
 - Thuật toán sinh khoá (key schedule algorithm)
 - Input: khoá K
 - Output: N_r khoá con: $\{K^1, K^2, \dots, K^{N_r}\}$
 - Thuật toán mã hoá trong mỗi vòng lặp (round function)
 - $g(w^{r-1}, K^r)$

$$\begin{aligned} w^0 &\leftarrow x \\ w^1 &\leftarrow g(w^0, K^1) \\ w^2 &\leftarrow g(w^1, K^2) \end{aligned}$$

.....

$$\begin{aligned} w^{N_{r-1}} &\leftarrow g(w^{N_{r-2}}, K^{N_{r-1}}) \\ w^{N_r} &\leftarrow g(w^{N_{r-1}}, K^{N_r}) \\ y &\leftarrow w^{N_r} \end{aligned}$$

Mạng SPN

- Độ dài khối: lm
- Round function
 - Thay thế (Substitution): π_S
 - S-box: thay thế các khối l bit bằng l khác
 - Hoán vị (Permutation): π_P
 - Thay đổi vị trí của lm bit
- Thuật toán mã hóa trong 1 vòng

$$u^r \leftarrow w^{r-1} \oplus K^r$$

$$v^r \leftarrow \pi_S(u^r)$$

$$w^r \leftarrow \pi_P(v^r)$$

Mạng SPN

Algorithm 3.1: SPN($x, \pi_S, \pi_P, (K^1, \dots, K^{Nr+1})$)

Số khoá con: $N_r + 1$

$w^0 \leftarrow x$

for $r \leftarrow 1$ **to** $Nr - 1$

do $\begin{cases} u^r \leftarrow w^{r-1} \oplus K^r \\ \textbf{for } i \leftarrow 1 \textbf{ to } m \\ \quad \textbf{do } v_{*}^r \leftarrow \pi_S(u_{*}^r) \\ \quad w^r \leftarrow (v_{\pi_P(1)}^r, \dots, v_{\pi_P(\ell m)}^r) \end{cases}**$

$u^{Nr} \leftarrow w^{Nr-1} \oplus K^{Nr}$

for $i \leftarrow 1$ **to** m

do $v_{*}^{Nr} \leftarrow \pi_S(u_{*}^{Nr})**$

$y \leftarrow v^{Nr} \oplus K^{Nr+1}$

output (y)

Mang SPN

- Ví dụ: $l = m = N^r = 4$

z	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\pi_S(z)$	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

z	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\pi_P(z)$	1	5	9	13	2	6	10	14	3	7	11	15	4	8	12	16

$K = 0011\ 1010\ 1001\ 0100\ 1101\ 0110\ 0011\ 1111$

$K^1 = 0011\ 1010\ 1001\ 0100$

$K^2 = 1010\ 1001\ 0100\ 1101$

$K^3 = 1001\ 0100\ 1101\ 0110$

$K^4 = 0100\ 1101\ 0110\ 0011$

$K^5 = 1101\ 0110\ 0011\ 1111$

Plain text $x = 0010\ 0110\ 1011\ 0111$

Mang SPN

$w^0 = 0010\ 0110\ 1011\ 0111$

$K^1 = 0011\ 1010\ 1001\ 0100$

$u^1 = 0001\ 1100\ 0010\ 0011$

$v^1 = 0100\ 0101\ 1101\ 0001$

$w^1 = 0010\ 1110\ 0000\ 0111$

$K^2 = 1010\ 1001\ 0100\ 1101$

$u^2 = 1000\ 0111\ 0100\ 1010$

$v^2 = 0011\ 1000\ 0010\ 0110$

$w^2 = 0100\ 0001\ 1011\ 1000$

$K^3 = 1001\ 0100\ 1101\ 0110$

$u^3 = 1101\ 0101\ 0110\ 1110$

$v^3 = 1001\ 1111\ 1011\ 0000$

$w^3 = 1110\ 0100\ 0110\ 1110$

$K^4 = 0100\ 1101\ 0110\ 0011$

$u^4 = 1010\ 1001\ 0000\ 1101$

$v^4 = 0110\ 1010\ 1110\ 1001$

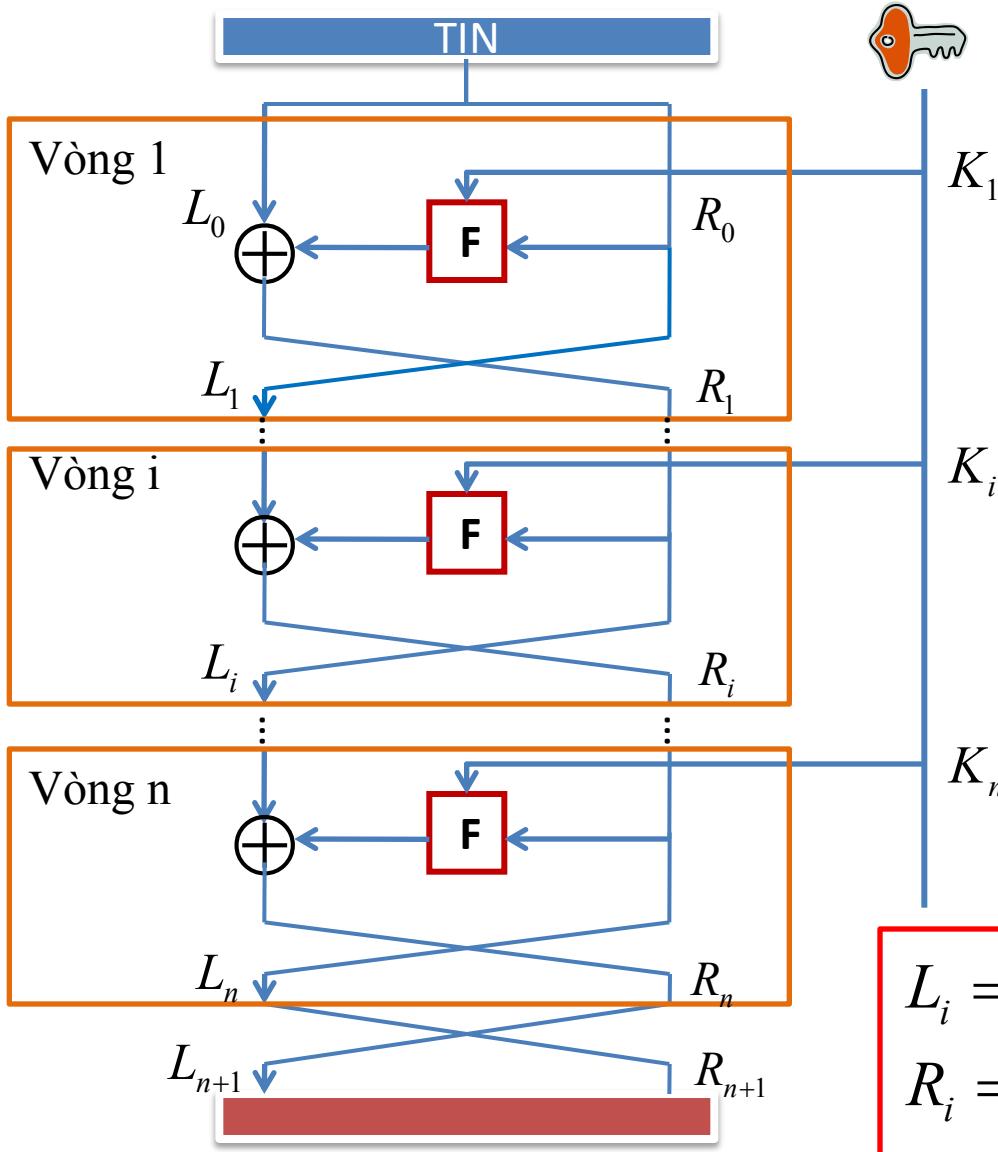
$K^5 = 1101\ 0110\ 0011\ 1111,$

$y = 1011\ 1100\ 1101\ 0110$

Cấu trúc Feistel

- Đầu tiên xuất bởi Feistel, năm 1973
- Được sử dụng bởi phần lớn các mật mã khối hiện nay
- Bao gồm nhiều vòng lặp,
 - Tại mỗi vòng lặp sẽ thực hiện các thao tác hoán vị và thay thế
 - Đầu vào của mỗi vòng lặp là đầu ra của vòng lặp trước đó và một khóa con được sinh ra từ khóa đầy đủ bởi thuật toán sinh quá
- Giải mã là một quá trình ngược với các khóa con cho mỗi vòng sẽ được phát sinh theo thứ tự ngược

Cấu trúc Feistel



$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

The Data Encryption Standard (DES)

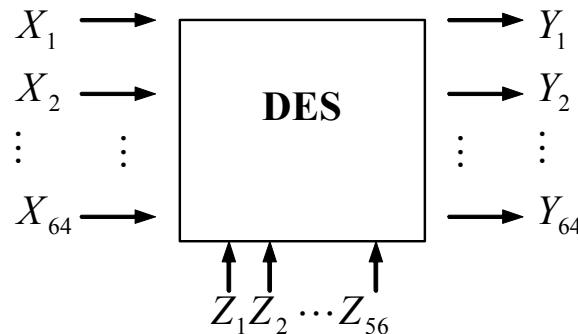
- Lịch sử
 - Vào những năm đầu thập kỷ 70, nhu cầu có một chuẩn chung về thuật toán mã hóa đã trở nên rõ ràng
 - Sự phát triển của công nghệ thông tin và của nhu cầu an toàn & bảo mật thông tin
 - Các thuật toán ‘cây nhà lá vườn’ (ad hoc) không thể đảm bảo được tính tin cậy đòi hỏi
 - Các thiết bị khác nhau đòi hỏi sự trao đổi thông tin mã hóa

The Data Encryption Standard (DES)

- Một chuẩn chung cần thiết phải có với các thuộc tính như
 - Bảo mật ở mức cao
 - Thuật toán được đặc tả và công khai hoàn toàn, tức là tính bảo mật không được phép dựa trên những phần che giấu đặc biệt của thuật toán
 - Việc cài đặt phải dễ dàng để đem lại tính kinh tế
 - Phải mềm dẻo để áp dụng được cho muôn vàn nhu cầu ứng dụng

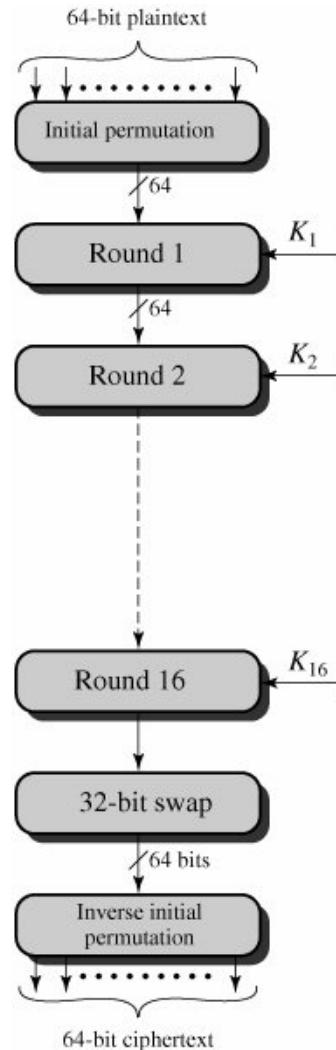
The Data Encryption Standard (DES)

- Năm 1973, Cục quản lý các chuẩn quốc gia của Mỹ đã có văn bản cổ động cho các hệ thống mã hóa ở cơ quan đăng ký liên bang của Mỹ. Điều đó cuối cùng đã dẫn đến sự phát triển của Data Encryption Standard, viết tắt là DES
- Sơ đồ chung



- Độ dài khối : 64 bit
- Độ dài khóa: 56 bit

The Data Encryption Standard (DES)

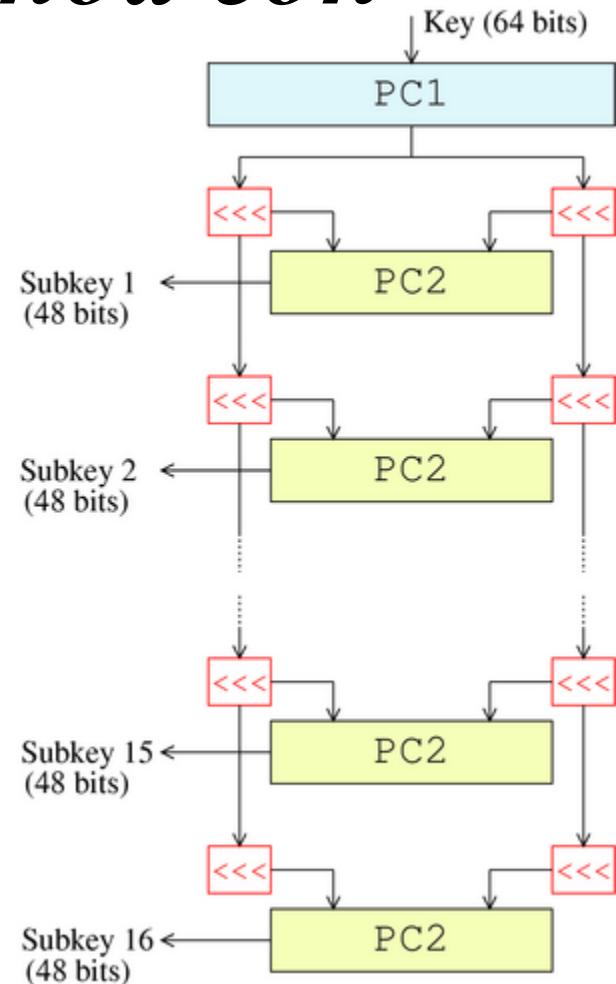


The Data Encryption Standard (DES)

- Bao gồm 16 bước lặp được kẹp bởi hai toán tử hoán vị IP, IP^{-1}
 - Các vòng lặp tuân theo cấu trúc Feistel
 - Toán tử IP, IP^{-1} mang ý nghĩa về mặt thực tiễn hơn là thuật toán (đơn giản hóa việc “chip hóa”)
- Hàm F là nguồn gốc của sức mạnh trong DES
 - Sự lặp lại nhiều lần các bước lặp với tác dụng của F là nhằm tăng cường thêm mãnh lực của F về mặt lượng

Thuật toán sinh khóa con

- 16 vòng lặp của DES chạy cùng thuật toán như nhau nhưng với các khóa khác nhau, được gọi là các khóa con
 - sinh ra từ khóa chính của DES bằng một thuật toán sinh khóa con.
- Khóa chính K, 64 bit, qua 16 bước biến đổi, mỗi bước sinh 1 khóa con 48 bit.
- Thực sự chỉ có 56 bit của khóa chính được sử dụng
 - 8 parity bits, lọc ra qua PC1.
 - Các bộ biến đổi PC1 và PC2 là các bộ vừa chọn lọc vừa hoán vị.
 - R1 và R2 là các phép đẩy bit trái 1 và hai vị trí.



Cấu trúc vòng lặp DES

- Mỗi vòng lặp của DES thực hiện trên cơ sở công thức sau:
$$(L_i, R_i) = (R_{i-1}, L_{i-1} \oplus f(R_{i-1}, K_i))$$
- Ta cũng có thể viết lại
$$(L_i, R_i) = T \bullet F (L_{i-1}, R_{i-1})$$
 - Trong đó F là phép thay thế L_{i-1} bằng $L_{i-1} \oplus f(R_{i-1}, K_i)$
 - T là phép đổi chỗ hai thành phần L và R.
 - Tức là mỗi biến đổi vòng lặp của DES có thể coi là một tích hàm số của F và T (trừ vòng cuối cùng không có T).
- Viết lại toàn bộ **thuật toán sinh mã DES** dưới dạng công thức:

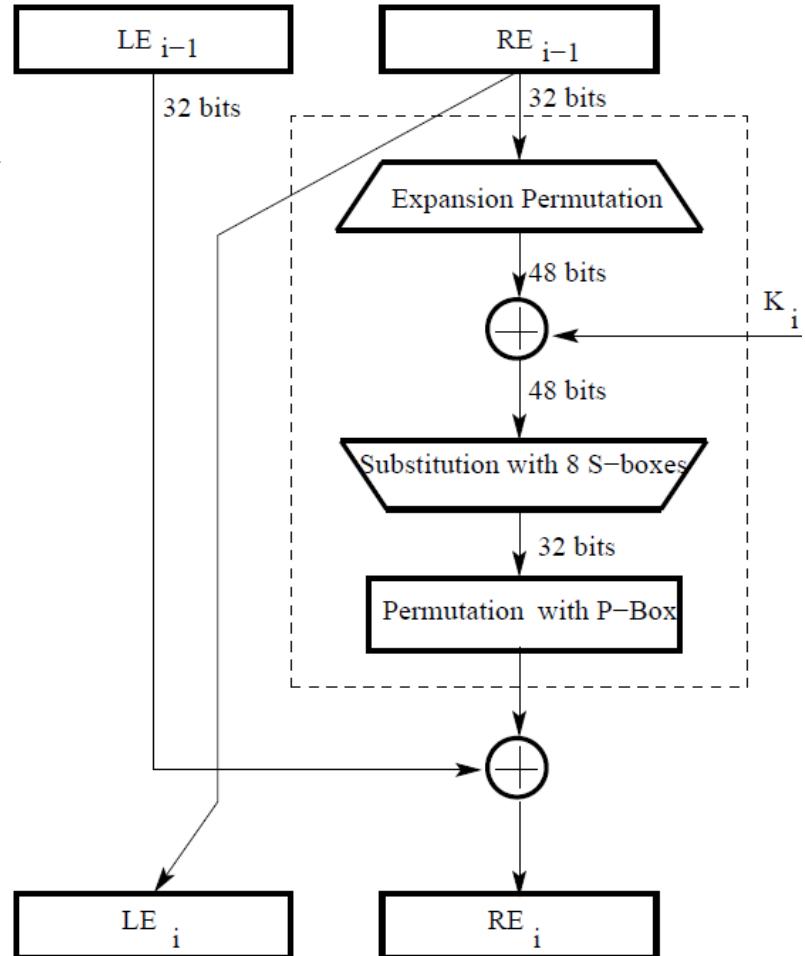
$$DES = (IP)^{-1} \bullet F_{16} \bullet T \bullet F_{15} \bullet T \bullet \dots \bullet F_2 \bullet T \bullet F_1 \bullet (IP)$$

Thuật toán giải mã DES

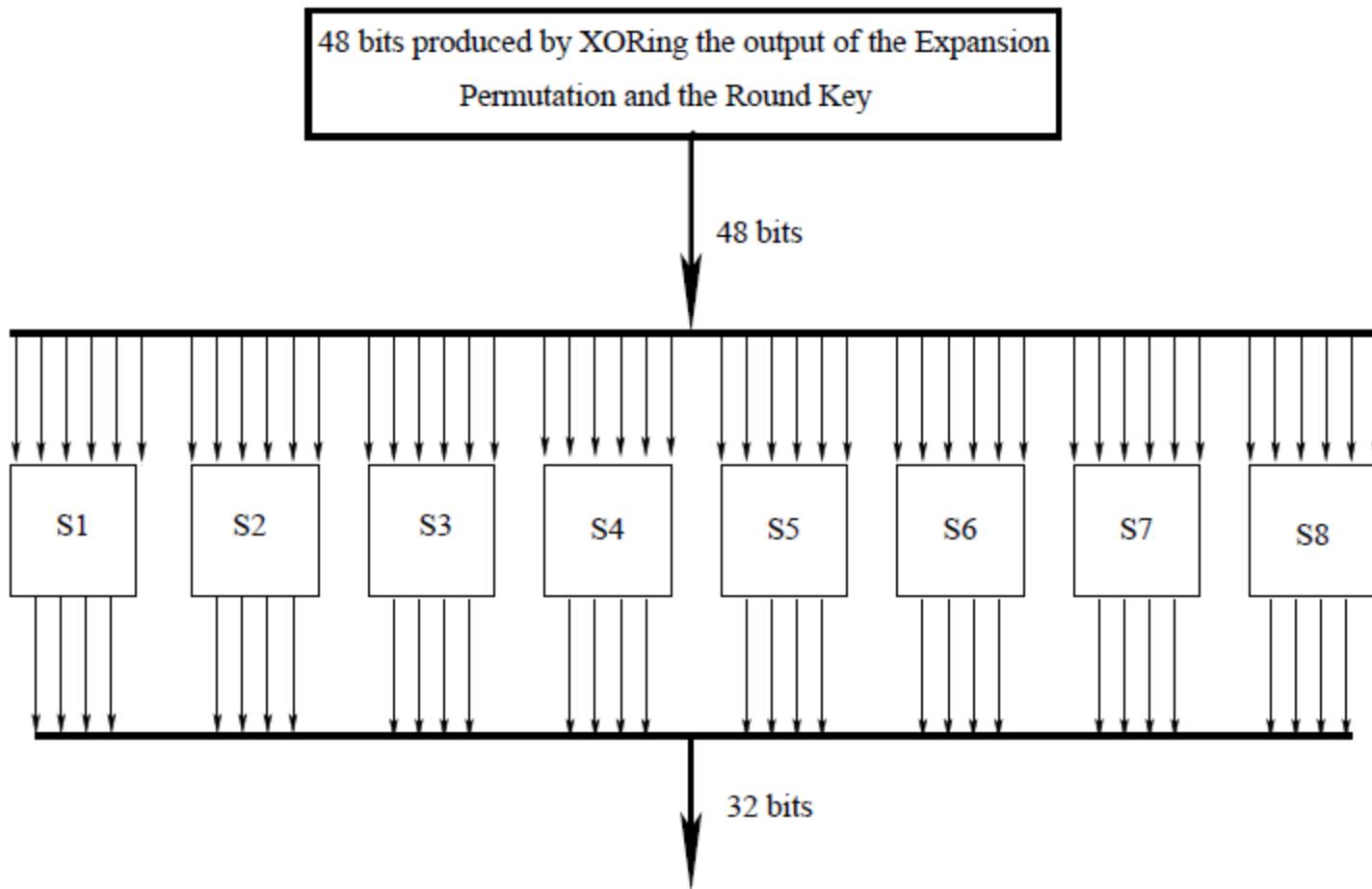
- Giống hệt như thuật toán sinh mã nhưng có các khóa con được sử dụng theo thứ tự ngược lại
 - Vì vậy, thuật toán giải mã có thể được viết lại dưới dạng công thức sau:
$$\text{DES}^{-1} = (\text{IP})^{-1} \bullet F_1 \bullet T \bullet F_2 \bullet T \bullet \dots \bullet F_{15} \bullet T \bullet F_{16} \bullet (\text{IP})$$
- Chú ý rằng mỗi hàm T hoặc F đều là các hàm có tính chất đối hợp ($f=f^{-1}$, hay $f(f(x))=x$) \rightarrow thực hiện DES • DES⁻¹ sẽ thu được phép đồng nhất.
 - Điều đó giải thích tại sao thuật toán giải mã lại giống hệt như sinh mã chỉ có khác thự từ dùng khóa con.

Cấu trúc cụ thể hàm f

- 32 bit của R_{i-1} được mở rộng thành 48 bit thông qua E rồi đệm XOR với 48 bit của K_i .
- 48 bit kết quả sẽ được phân thành 8 nhóm 6 bit; mỗi nhóm này sẽ qua một biến đổi đặc biệt, S-box, và biến thành 4 bit.
 - có 8 S-box khác nhau ứng với mỗi nhóm 6 bit
- 32 bit hợp thành từ 8 nhóm 4 bit (sau khi qua các S-box) sẽ được hoán vị lại theo P rồi đưa ra kết quả cuối cùng của hàm f (F_i).



S-boxes



Cấu trúc của các S-Box

- Mỗi S-box như một bộ biến đổi gồm 4 bảng biến đổi, mỗi bảng biến đổi 1 đầu vào 4 bit thành đầu ra cũng 4 bit (bảng 16 dòng).
 - Đầu vào 4 bit chính là lấy từ các bit 2-5 của nhóm 6 bit.
 - Các bit 1 và 6 sẽ dùng để xác định 1 trong 4 bảng biến đổi của S-box. Vì thế chúng được gọi là các bit điều khiển (CL và CR: left control và right control bit).

S ₅		Middle 4 bits of input															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Outer bits	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

Các thuộc tính của S-Box

- Các nguyên tắc thiết kế của 8 S-boxes được đưa vào lớp ‘Classified information’ ở Mỹ.
- NSA đã tiết lộ 3 thuộc tính của S-boxes, những thuộc tính này bảo đảm tính confusion & diffusion của thuật toán.
 1. Các bít ra (output bit) luôn phụ thuộc không tuyến tính vào các bít vào (input bit).
 2. Sửa đổi ở một bit vào làm thay đổi ít nhất là hai bit ra.
 3. Khi một bit vào được giữ cố định và 5 bit con lại cho thay đổi thì S-boxes thể hiện một tính chất được gọi là ‘phân bố đồng nhất’ (uniform distribution): so sánh số lượng bit số 0 và 1 ở các đầu ra luôn ở mức cân bằng.
 - Tính chất này khiến cho việc áp dụng phân tích theo lý thuyết thông kê để tìm cách phá S-boxes là vô ích.

- 3 tính chất này đảm bảo tốt confusion & diffusion.
 - Sau 8 vòng lặp tất cả các bit ra của DES sẽ chịu ảnh hưởng của tất cả các bit vào và tất cả các bit của khóa.
- Tuy nhiên cấu tạo của S-box đã gây tranh luận mạnh mẽ từ hàng thập kỷ qua về khả năng cơ quan NSA (National Security Agency), Mỹ, vẫn còn che dấu các một số đặc tính của S-box hay cài bén trong những cửa bẫy (trapdoor) mà qua đó họ có thể dễ dàng phá giải mã hơn người bình thường.

Cryptography II

Block ciphers and modes of operations

Block ciphers: getting the concept

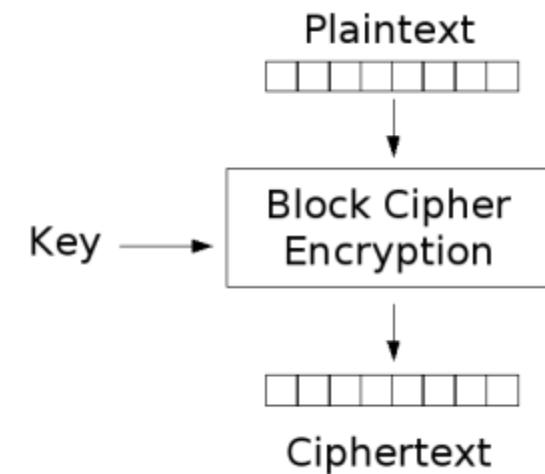
- Stream cipher vs. block cipher: single unit vs. block of units

key	000	001	010	011	100	101	110	111
0	001	111	110	000	100	010	101	011
1	001	110	111	100	011	010	000	101
2	001	000	100	101	110	111	010	011
3	100	101	110	111	000	001	010	011
4	101	110	100	010	011	001	011	111

- Plaintext= 010100110111= (010)(100)(110)(111)
 - → Ciphertext = 111 011 000 101 theo key=1
 - → Ciphertext = 100 011 011 111 theo key=4
- There are 5 keys, $2^2 < 5 < 2^3 \rightarrow$ need keys in 3 bits to present \rightarrow key size= block size= 3.
- Small sizes are dangerous, however: If Eve catches C=001 \rightarrow can infer P= 000 or 101.

Block cipher: an invertible map

- Map n-bit plaintext blocks to n-bit ciphertext blocks (n: block size/length).
- For n-bit plaintext and ciphertext blocks and a fixed key, the encryption function is a bijection:
 - $E : P_n \times K \rightarrow C_n$ s.t. for all key $k \in K$, $E(x, k)$ is an invertible mapping written $E_k(x)$.
- The inverse mapping is the decryption function, $y = D_k(x)$ denotes the decryption of plaintext x under k .



General condition in creating secure block ciphers

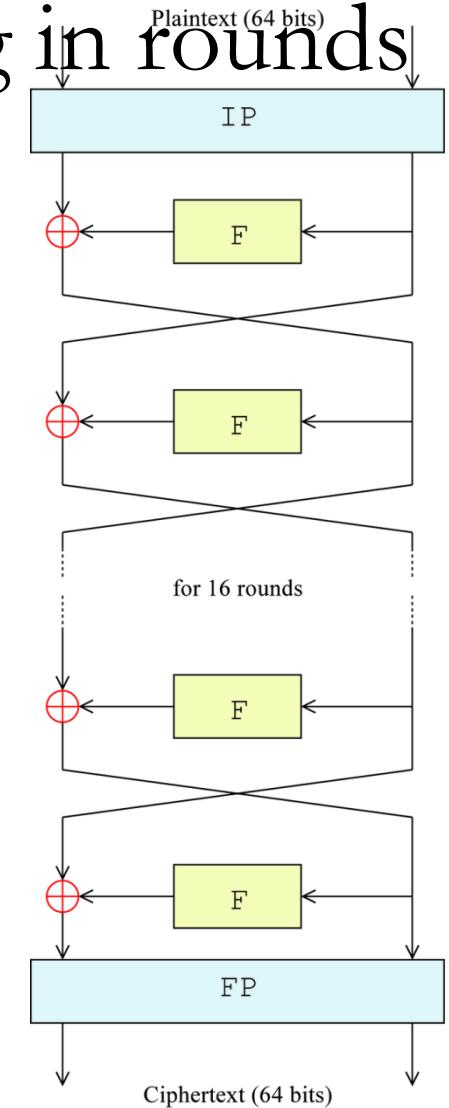
- The block size has to be large enough to prevent against statistical analysis
 - However, larger block size means slower processing
- The key space (then key length) must be large enough to prevent against exhaustive key search
 - However, key length shouldn't be too big that makes key distribution and management more difficult

General principles in designing secure block ciphers

- *Confusion:* As a function, the dependence of the ciphertext on the plaintext should be complex enough so that enemy can't find the rules
 - The function should be non-linear.
- *Diffusion:* The goal is to spread the information from the plaintext over the entire ciphertext so that changes in plaintext affect many parts in ciphertext
 - This makes it difficult for an enemy to break the cipher by using statistical analysis
- Confusion is made by usings substitutions while *diffusion* by transpositions and/or permutations.

The Feistel structure: processing in rounds

- Block ciphers are usually designed with many rounds where basic round accomplishes the core function f for basic confusion and diffusion.
 - The input of a round is the output of the previous round and a subkey which is generated by a key-schedule algorithm
- The decryption is a reverse process where the sub-keys are handled in the reverse order



The overall Feistel structure of DES

Block Ciphers Features

- Block size: in general larger block sizes mean greater security.
- Key size: larger key size means greater security (larger key space).
- Number of rounds: multiple rounds offer increasing security.
- Encryption modes: define how messages larger than the block size are encrypted, very important for the security of the encrypted message.

History of Data Encryption Standard (DES)

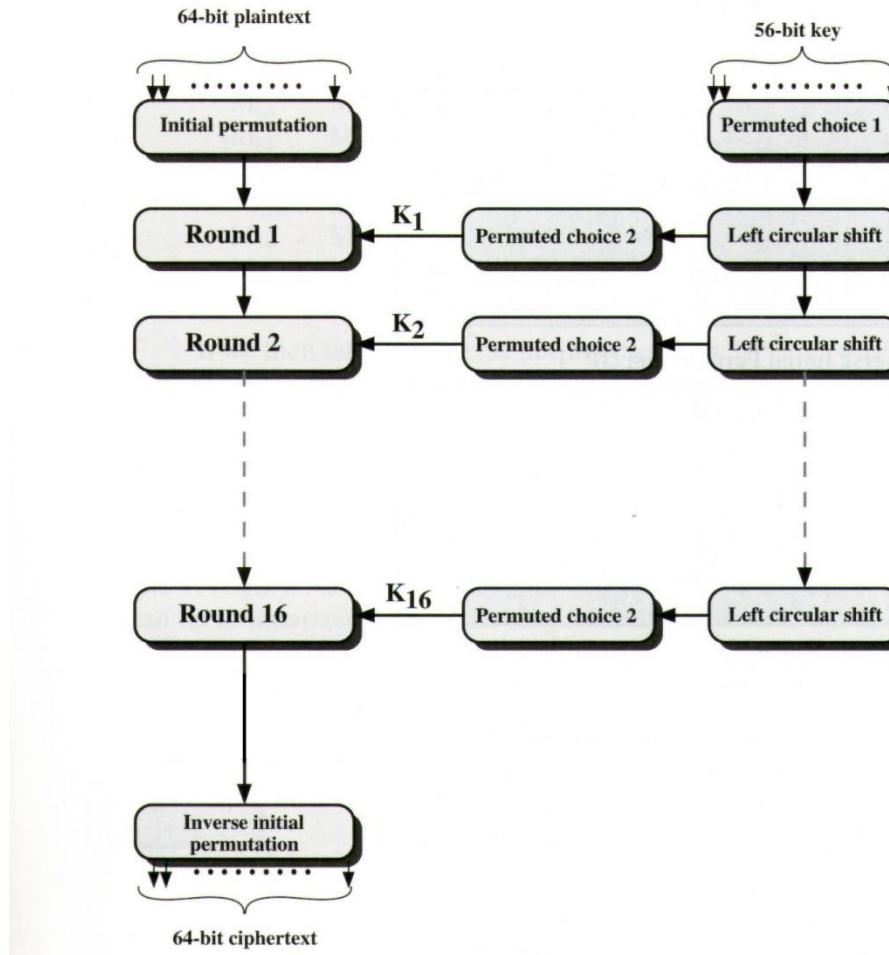
- 1967: Feistel at IBM
 - Lucifer: block size 128; key size 128 bit
- 1972: NBS asks for an encryption standard
- 1975: IBM developed DES (modification of Lucifer)
 - block size 64 bits; key size 56 bits
- 1975: NSA suggests modification
- 1977: NBS adopts DES as encryption standard in (FIPS 46-1, 46-2).
- 2001: NIST adopts Rijndael as replacement to DES

DES Features

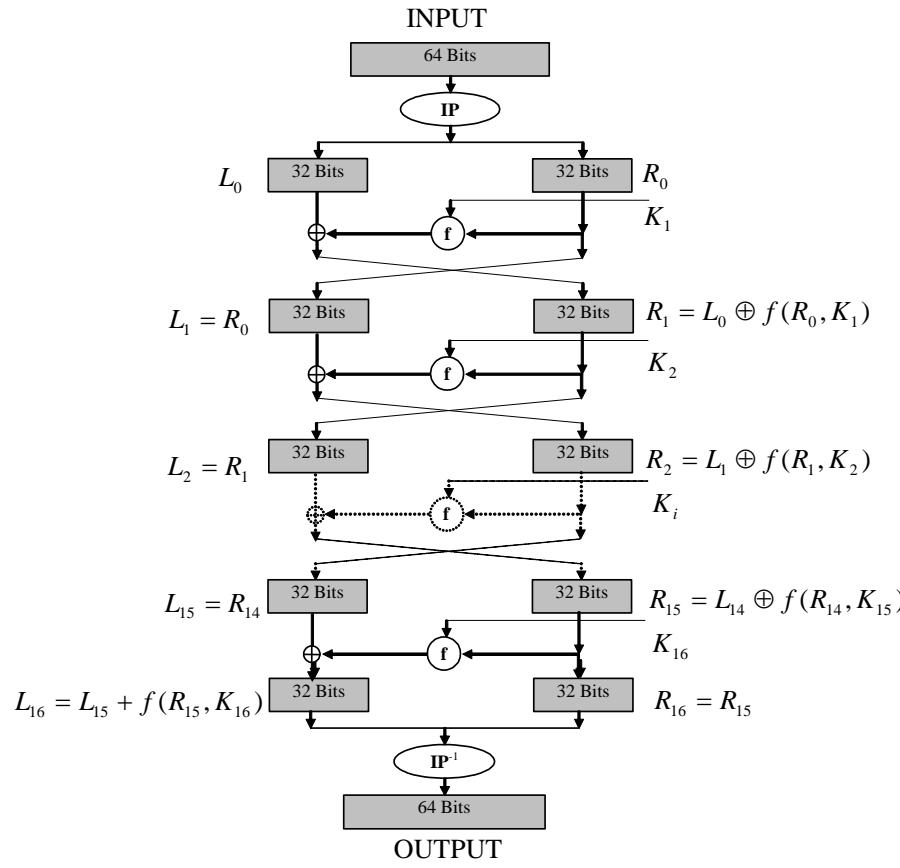
■ Features:

- **Block size = 64 bits**
- **Key size = 56 bits**
- **Number of rounds = 16**
- **16 intermediary keys, each 48 bits**

DES Rounds



DES encryption: A closer look



Decryption uses the same algorithm as encryption, except that the subkeys K_1, K_2, \dots, K_{16} are applied in reversed order

Cryptanalysis of DES

Brute Force:

- Known-Plaintext Attack
- Try all 2^{56} possible keys
- Requires constant memory
- Time-consuming
- DES challenges: (RSA)
 - msg=“the unknown message is :xxxxxxxx”
 - CT=“ C1 | C2 | C3 | C4”
 - 1997 Internet search: 3 months
 - 1998 EFF machine (costs \$250K): 3 days
 - 1999 Combined: 22 hours

Rijndael Features

- Designed to be efficient in both hardware and software across a variety of platforms.
- Uses a variable block size, **128, 192, 256-bits**, key size of **128-, 192-, or 256-bits**.
- 128-bit round key used for each round (Can be precomputed and cached for future encryptions).
- Note: AES uses a 128-bit block size.
- Variable number of rounds (10, 12, 14):
 - 10 if $B = K = 128$ bits
 - 12 if either B or K is 192 and the other is ≤ 192
 - 14 if either B or K is 256 bits

Rijndael Design

- Operations performed on State (4 rows of bytes).
- The 128 bit key is expanded as an array of 44 32bits words; 4 distinct words serve as a round key for each round; key schedule relies on the S-box
- Algorithms composed of three layers
 - Linear diffusion
 - Non-linear diffusion
 - Key mixing

Decryption

- The decryption algorithm is not identical with the encryption algorithm, but uses the same key schedule.
- There is also a way of implementing the decryption with an algorithm that is equivalent to the encryption algorithm (each operation replaced with its inverse), however in this case, the key schedule must be changed.

Rijandel Cryptanalysis

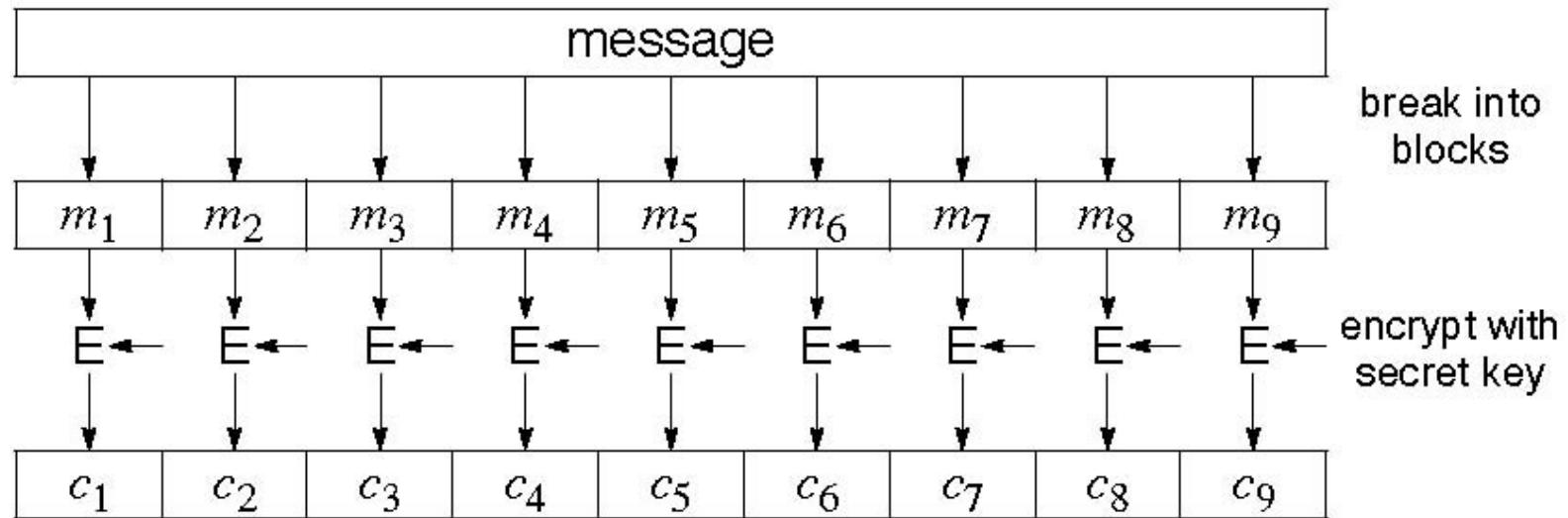
- Academic break on weaker version of the cipher, 9 rounds
- Requires 2^{224} work and 2^{85} chosen *related-key* plaintexts.
- Attack not practical.

Modes of Operation (Encryption modes)

- Mode of operation (or encryption mode):
 - A block cipher algorithm takes on a fixed-length input, i.e. a block, and produce an output, usually a block of the same fixed-length.
 - In practice, we want to encrypt files of various length → need to divide a file into block of that given fixed length → then call the encryption algorithms several times
 - Operation mode: the manner and structure in which we feed the encryption algorithm (several times) with blocks of the plaintext file and concatenate the resulted blocks to produce the ciphertext file.
- The popular modes:
 - ECB, CBC, OFB, CFB, CTR
- We now overview the properties of certain modes (privacy, integrity) and potential attacks against them.

Electronic Code Book (ECB)

- Each block is independently encoded



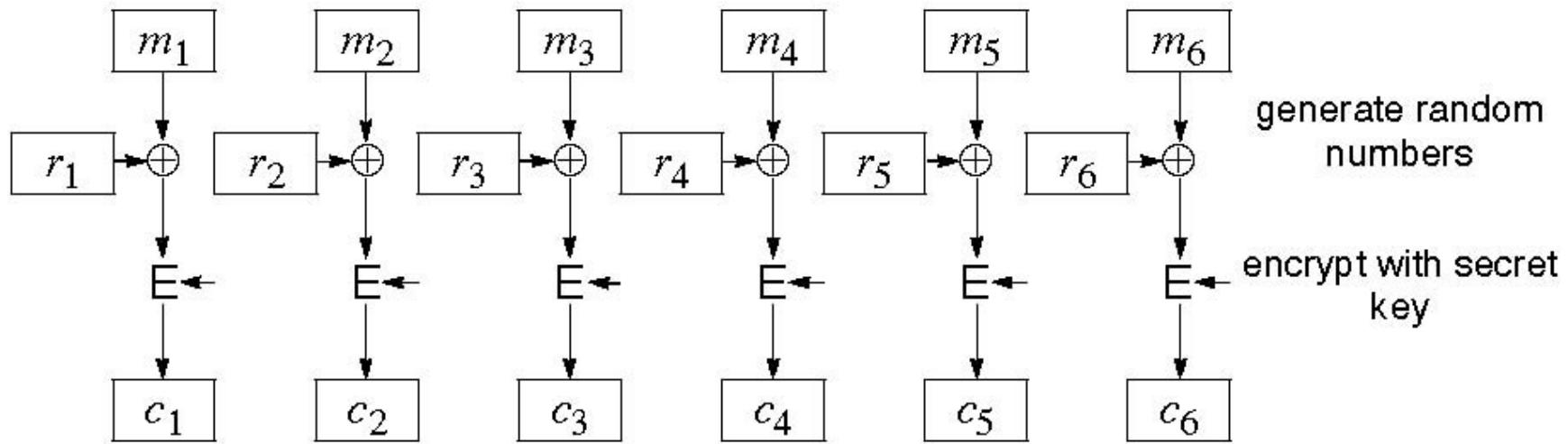
- Problem:
 - Identical Input \rightarrow Identical Output
 - Deterministic: the same data block gets encrypted the same way, reveals patterns of data when a data block repeats.

ECB critics

- Weakness: Replay/Manipulation attack
 - Can insert encoded blocks
 - Reordering ciphertext results in reordered plaintext.
- Strength:
 - Errors in one ciphertext block do not propagate.
- Usage:
 - not recommended to encrypt more than one block of data
 - Encryption in database

Cipher Block Chaining (CBC)

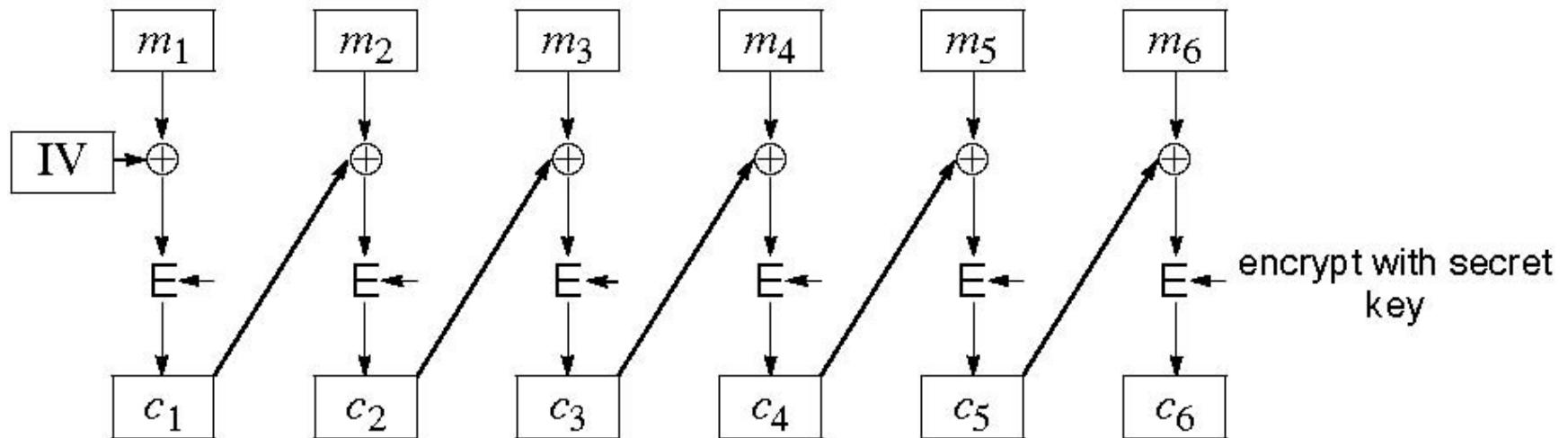
- Improving on ECB: think of adding a random number before encoding



CBC (cont.)

The main idea:

- Use C_i as random number block operation for $i+1$
- So, need a so called Initial Value (IV)
 - If no IV, then one can guess changed blocks

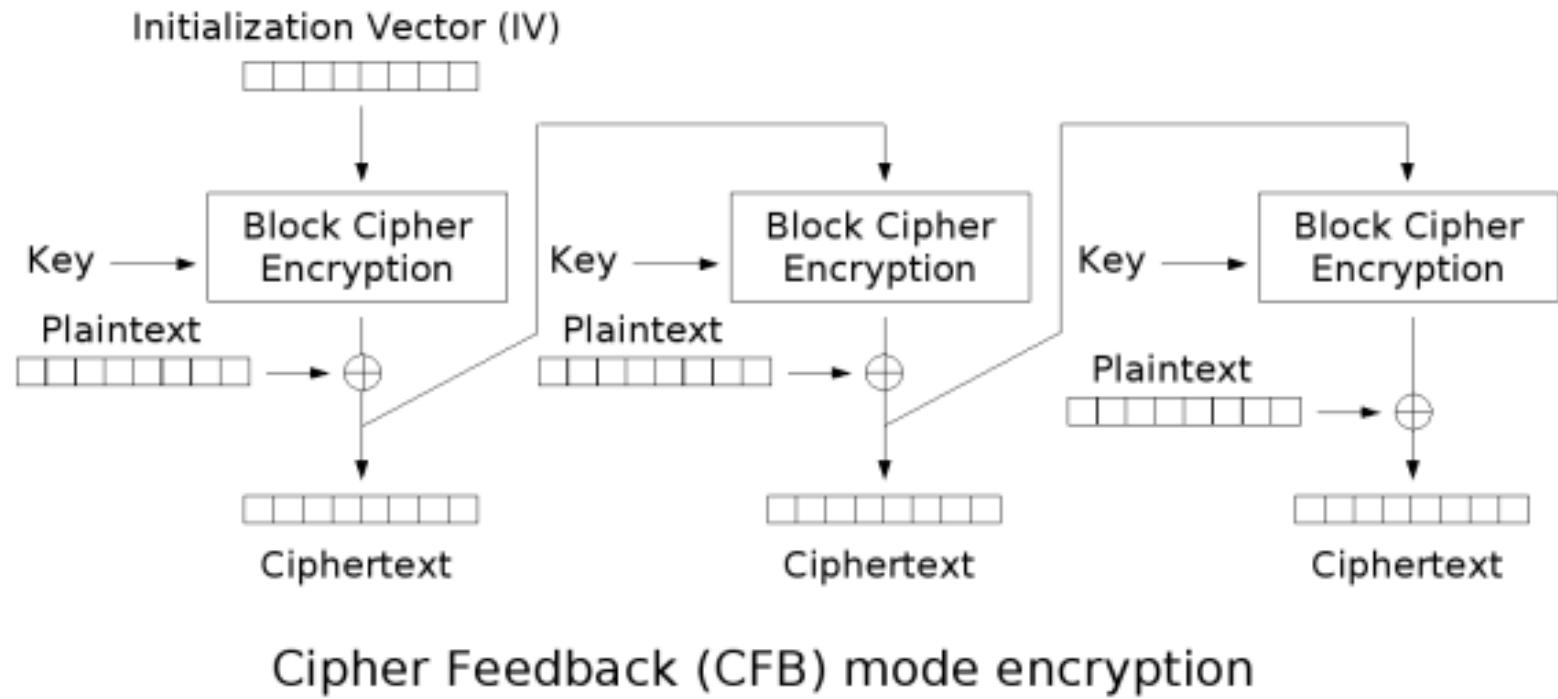


CBC critics

- Good
 - Randomized encryption: repeated text gets mapped to different encrypted data.
 - Can be proven to be “secure” assuming that the block cipher has desirable properties and that random IV’s are used
 - A ciphertext block depends on all preceding plaintext blocks
 - reorder affects decryption
- Bad
 - Errors in one block propagate to two blocks
 - one bit error in C_j affects all bits in M_j and one bit in M_{j+1}
 - Sequential encryption, cannot use parallel hardware
 - Observation: if $C_i = C_j$ then $E_k(M_i \oplus C_{i-1}) = E_k(M_j \oplus C_{j-1})$; thus $M_i \oplus C_{i-1} = M_j \oplus C_{j-1}$; thus $M_i \oplus M_j = C_{i-1} \oplus C_{j-1}$

Cipher Feedback (CFB)

- The message is XORed with the feedback of encrypting the previous block

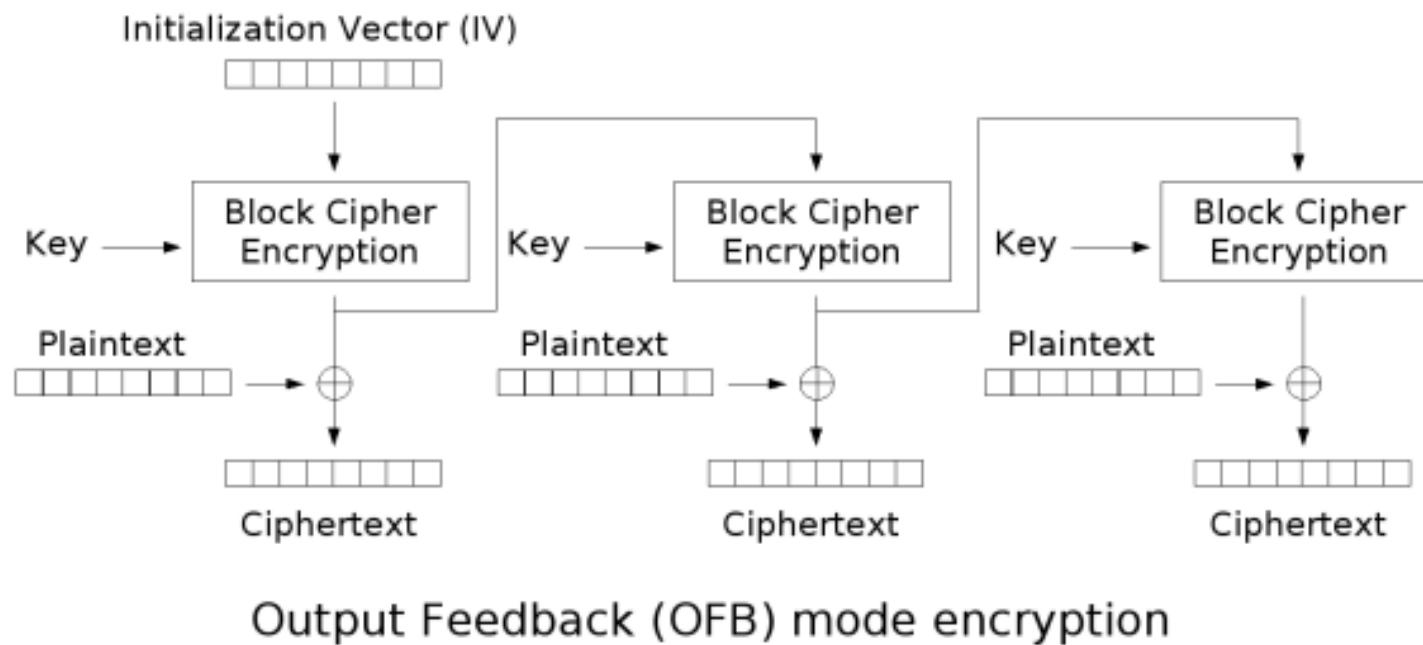


CFB critics

- Good
 - Randomized encryption
 - A ciphertext block depends on all preceding plaintext blocks; reorder affects decryption
- Bad
 - Errors propagate for several blocks after the error, but the mode is self-synchronizing (like CBC).
 - Decreased throughput.
 - Can vary the number of bits feed back, trading off throughput for ease of use
 - Sequential encryption

Output Feedback (OFB)

- IV is used to generate a stream of blocks
- Stream is used a one-time pad and XOR'ed to plain text

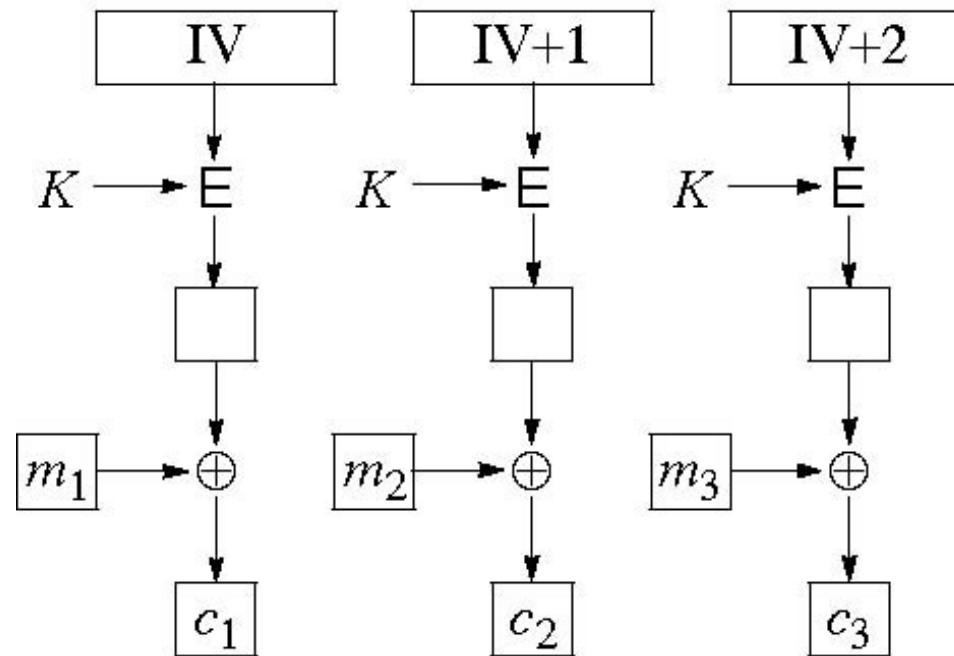


OFB critics

- Randomized encryption
- Sequential encryption, but preprocessing possible
- Error propagation limited
- Subject to limitation of stream cipher

Counter Mode (CTR)

- If the same IV and key is used again,
 - XOR of two encrypted messages = XOR of plain text
- IV is incremented and used to generated one-time pad



CTR critics

- Software and hardware efficiency: different blocks can be encrypted in parallel.
- Preprocessing: the encryption part can be done offline and when the message is known, just do the XOR.
- Random access: decryption of a block can be done in random order, very useful for hard-disk encryption.
- Messages of arbitrary length: ciphertext is the same length with the plaintext (i.e., no IV).



Mật mã bắt đối xứng

Public key cryptography

Điểm yếu của mã đối xứng

- Việc phân phối và quản lý khoá khó khăn, nhất là với các hệ thống có nhiều người sử dụng
 - N người dùng $\rightarrow n(n-1)/2$ mối quan hệ \rightarrow mỗi người cần quản lý $(n-1)$ khoá
- Không thể sử dụng vào chữ ký điện tử
 - Không thể đảm bảo được tính “không chối từ” (sẽ học ở các buổi sau)

- Alice – Bob
 - Share key: K
 - Plain text: x
 - A -> B: $E_K(x)$
 - B: $D_K(E_K(x)) = x$
- Alice: p_K, q_K
 - B->A: $E_{q_K}(x) = y$
 - A: $D_{p_K}(y) \neq x$
 - $D_{p_K}(y) = x$

Ý tưởng của Diffie-Hellman về hệ mã bất đối xứng (mã công khai)

- Về nguyên tắc, mã công khai được thiết kế trên quan điểm “hướng tới 1 người dùng”, chứ không phải hướng tới 1 cặp người dùng (như mã bí mật)
 - Được sử dụng với nhiều mục đích khác ngoài việc mã hoá
- Đầu tiên bởi Diffie và Hellman (1976) trong bài báo “New Directions in Cryptography”
 - Cơ chế mã hoá
 - Cơ chế phân phối khoá
 - Thuật toán phân phối khoá Diffie-Hellman
 - Chữ ký điện tử

Đề xuất của Diffie-Hellman

- Mỗi người dùng tạo 2 khoá: 1 khoá giữ bí mật (secret (private) key) và 1 khoá công khai cho tất cả mọi người khác (public key)
 - Khóa bí mật được dùng để mã hoá, khoá bí mật được dùng để giải mã
$$X = D(z, E(Z, X))$$
 - Khóa bí mật được dùng để tạo chữ ký điện tử, khoá công khai được dùng để xác thực chữ ký điện tử
$$X = E(Z, D(z, X))$$
- Mã công khai còn được gọi là mã bất đối xứng (asymmetric key cryptosystems)
 - Kẻ cờ biết được bản mã (cipher text) và khoá công khai (public-key) thì cũng không thể tính ngược lại được bản rõ và khoá bí mật

Nguyên tắc cấu tạo một hệ PK (trapdoor)

- Một hệ mã PKC có thể được tạo dựng trên cơ sở sử dụng một hàm kiểu one-way (1 chiều). Một hàm f được gọi là one-way nếu:
 1. Đối với mọi X tính ra $Y = f(X)$ là dễ dàng.
 2. Khi biết Y rất khó để tính ra X .
- Ví dụ. Cho n số nguyên tố p_1, p_2, \dots, p_n ta có thể dễ dàng tính được $N = p_1 * p_2 * \dots * p_n$, tuy nhiên khi biết N , việc tìm các thừa số nguyên tố của nó là khó khăn hơn rất nhiều
- Cần một hàm one-way đặc biệt, trang bị một trap-door (cửa bẩy), sao cho nếu biết trap-door này thì việc tính X khi biết $f(X)$ (tức là đi tìm nghịch đảo của f) là dễ, còn ngược lại thì khó
- Một hàm one-way có trap door như thế \rightarrow một hệ mã PKC
 - Lấy E_z (hàm sinh mã) là hàm one-way có trap-door
 - Trap-door chính là khoá mật, mà nếu biết nó thì có thể dễ dàng tính được cái nghịch đảo của E_z tức là biết D_z , còn nếu không biết thì rất khó tính được.

Hệ mã công khai RSA

- Phát minh năm **1978** bởi **Rivest, Adi Shamir and Leonard Adleman**
 - Được công bố bởi R L Rivest, A Shamir, L Adleman, "*On Digital Signatures and Public Key Cryptosystems*", Communications of the ACM, vol 21 no 2, pp120-126, Feb 1978
 - Tính an toàn được dựa trên độ khó của bài toán phân tích thừa số nguyên tố của một số rất lớn

Ý tưởng chính

- Thuật toán mã hoá và giải mã là các hàm đồng fduw của các luỹ thừa trong trường $Z_n = \{0,1,2,\dots n - 1\}$
 - Mã hoá : $Y \equiv X^e \pmod{n}$
 - Decryption: $X \equiv Y^d \pmod{n}$
 - Để đảm bảo tính đúng đắn của thuật toán
 - e & d phải thoả mãn: $X^{ed} \equiv X \pmod{n}$

Ý tưởng chính

- Định lý Euler: nếu $(X, n) = 1$ thì
 - $\varphi(n)$: số lượng các số $k: 0 < k < n | (k, n)$ $X^{\varphi(n)} \equiv 1 \pmod{n} = 1$
 - Nếu $n = p \times q$ (p, q nguyên tố) $\rightarrow \varphi(n) = (p-1)(q-1)$
- Chọn e và tìm d sao cho $ed \equiv 1 \pmod{\varphi(n)}$
 - $d \equiv e^{-1} \pmod{\varphi(n)}$
 - $X^{ed} \equiv X^{k\varphi(n)+1} \equiv (X^{\varphi(n)})^k \times X \equiv X \pmod{n}$
- Chú ý: để giải mã được \rightarrow cần biết $\varphi(n) \rightarrow$ cần biết $p, q \rightarrow$ vì n rất lớn nên việc phân tích n để tìm p, q là không khả thi
- X chia hết $p, q \rightarrow X^{ed} - X$ chia hết cho n
- $X = X' \cdot p^a$ (X' không chia hết cho q)
- $X^{ed} - X = (X' \cdot p^a)^{ed} - X' \cdot p^a = p^a(X'^{ed}(p^a)^{ed-1} - X')$
- $X'^{ed}(p^a)^{ed-1} - X'$ chia hết cho q
- $(p^a)^{ed-1} = (p^a)^{k\varphi(n)} = (p^a)^{k(q-1)(p-1)} \equiv 1 \pmod{q}$
- $X'^{ed}(p^a)^{ed-1} - X' \equiv X'^{ed} - X'$

Hệ mã RSA

■ Tạo khoá:

- Chọn 2 số nguyên tố rất lớn và có độ lớn tương đương (~512 bit): p, q
- Tính $n = pq$, và $\varphi(n) = (q - 1)(p - 1)$
- Chọn 1 số tự nhiên e tùy ý, sao cho $1 < e < \varphi(n)$, và $\gcd(e, \varphi(n)) = 1$
- Tìm d , sao cho $1 < d < \varphi(n)$ và $ed \equiv 1 \pmod{\varphi(n)}$
- **Khoá công khai : (e, n) ; khoá bí mật : d**
 - **Chú ý: p và q phải giữ bí mật**

Hệ mã RSA

■ Mã hoá

- Cho trước bản rõ M biểu diễn dưới dạng nhị phân → convert M sang hệ cơ số 10: $0 < M < n$
- Dùng khoá công khai (e, n) và mã hoá:

$$C = M^e \pmod{n}$$

■ Giải mã

- Cho bản mã C , sử dụng khoá bí mật (d) và giải mã:
 - $M = C^d \pmod{n}$

■ Tính đúng đắn

- $C^d \pmod{n} \equiv M^{ed} \pmod{n} \equiv M \pmod{n}$

Ví dụ

■ Parameters:

- Select $p = 11$ và $q = 13$
- $n = 11 * 13 = 143$; $m = (p - 1)(q - 1) = 10 * 12 = 120$
- Chọn $e = 37 \rightarrow \gcd(37, 120) = 1$
- Tìm d sao cho: $e \times d \equiv 1 \pmod{120} \rightarrow d = 13$ ($e \times d = 481$)

■ Mã hóa

- Cắt bản rõ thành các đoạn u bits, $2^u \leq 142 \rightarrow u = 7$
 - Mỗi đoạn sẽ là 1 số tự nhiên từ 1 đến 127
- Tính $Y = X^e \pmod{n}$

Ví dụ: $X = (0000010) = 2$, ta có $Y \equiv X^{37} \equiv 12 \pmod{143}$ $\square Y = (00001100)$

■ Giải mã: $X \equiv 12^{13} \pmod{143} = 2$

Cách tính nghịch đảo đồng dư

- Định lý Bézout: nếu $d = GCD(a, b)$ thì tồn tại 2 số x, y sao cho $d = xa + yb$ (đồng nhất thức Bézout), x, y được gọi là hệ số của a, b
- Nếu $1 = GCD(e, n) \rightarrow 1 = xe + yn \rightarrow xe \equiv 1(mod\ n) \rightarrow x \equiv e^{-1}(mod\ n)$
- Phương trình Diophantine: $ax+by=c$
 - Chỉ có nghiệm khi $c : d = \gcd(a, b)$

Cách tính nghịch đảo đồng dư

- Thuật toán Oclit tìm ước số chung lớn nhất của 2 số r_0, r_1

$$\begin{array}{lll} r_0 & = & q_1 r_1 + r_2, & 0 < r_2 < r_1 \\ r_1 & = & q_2 r_2 + r_3, & 0 < r_3 < r_2 \\ \vdots & & & \\ r_{m-2} & = & q_{m-1} r_{m-1} + r_m, & 0 < r_m < r_{m-1} \\ r_{m-1} & = & q_m r_m. & \end{array}$$

- Chứng minh được: $\gcd(r_0, r_1) = \gcd(r_1, r_2) = \cdots = \gcd(r_{m-1}, r_m) = r_m$

Cách tính nghịch đảo đồng dư

■ Ví dụ

- Tìm ước số chung lớn nhất của (252, 198)

$$252 = 198 \times 1 + 54$$

$$198 = 54 \times 3 + 36$$

$$54 = 36 \times 1 + 18$$

$$36 = 18 \times 2 + 0$$



$$\text{Gcd}(252, 198) = 18$$

Cách tính nghịch đảo đồng dư

■ Ví dụ

- Giải phương trình nghiệm nguyên: $252x+198y=18$

$$252 = 198 \times 1 + 54$$

$$198 = 54 \times 3 + 36$$

$$54 = 36 \times 1 + 18$$

$$36 = 18 \times 2 + 0$$



$$18 = 54 - 36$$

$$18 = 54 - (198 - 54 \times 3)$$

$$18 = 54 \times 4 - 198$$

$$18 = (252 - 198) \times 4 - 198$$

$$18 = 252 - 198 \times 5$$



$$(x, y) = 1, -5$$

Cách tính đồng dư luỹ thừa

- Tính $x^a \pmod n$
- Cách đơn giản nhất:
 - $x^a \pmod n = x \pmod n \times x \pmod n \times \dots \times x \pmod n$
 - → thực hiện phép lấy đồng dư và phép nhân a lần
- Sử dụng phương pháp **bình phương và nhân**

Cách tính đồng dư luỹ thừa

PP: bình phương và nhân

- Biểu diễn a dưới dạng nhị phân: $a = \sum_{i=0}^l a_i 2^i$

```
z ← 1  
For i = l down to 0  
    z ← z2 mod n  
    if ai = 1 then  
        z ← (z×x) (mod n)  
    end if  
End for  
Return z
```

Tìm số dư của x^{19} khi chia cho n
 $19 = 16 + 2 + 1 = 2^4 + 2^1 + 2^0 = 10011$

$z \leftarrow 1$
 $i = 4; a_4 = 1; z \leftarrow z^2 \times x \equiv 1^2 \times x \equiv x$
 $i = 3; a_3 = 0; z \leftarrow z^2 \equiv x^2$
 $i = 2; a_2 = 0; z \leftarrow z^2 \equiv x^4$
 $i = 1; a_1 = 1; z \leftarrow z^2 \times x \equiv x^8 \times x \equiv x^9$
 $i = 0; a_0 = 1; z \leftarrow z^2 \equiv x^{18} \times x \equiv x^{19}$

Tìm số dư của 3^{19} khi chia cho 5
 $19 = 10011$

$z \leftarrow 1$
 $i = 4; a_4 = 1; z \leftarrow 1^2 \times 3 \equiv 3$
 $i = 3; a_3 = 0; z \leftarrow 3^2 \equiv -1$
 $i = 2; a_2 = 0; z \leftarrow (-1)^2 \equiv 1$
 $i = 1; a_1 = 1; z \leftarrow 1^2 \times 3 \equiv 3$
 $i = 0; a_0 = 1; z \leftarrow 3^2 \times 3 \equiv -3 \equiv 2$

Bài tập

1. Tính

1. $28^{-1} \bmod 75$
2. $17^{-1} \bmod 101$
3. $357^{-1} \bmod 1234$
4. $3125^{-1} \bmod 9987$
2. Chứng minh: $X^{(p-1)(q-1)} \equiv 1 \pmod{pq}$ với p, q nguyên tố
3. Viết đoạn giả mã của thuật toán tính nghịch đảo đồng dư
4. Chứng minh tính đúng đắn của phương pháp bình phương và nhân
5. Tính
 1. $9726^{3533} \pmod{11413}$

Bài tập lớn

1. Viết chương trình phá mã thế (1 bảng thế) bằng phương pháp thống kê
2. Viết chương trình phá mã vigenere (mã đã bảng thế)
3. Viết chương trình mã hoá và phá mã RSA như sau

1. Mã hoá:
 1. Input: bản rõ, khoá công khai (d, n)
 2. Bản rõ là 1 văn bản tiếng anh. Mỗi từ được encode theo bảng chữ cái, ví dụ như sau:
 - DOG $\rightarrow 3 \times 26^2 + 14 \times 26 + 6 = 2398$
 - CAT $\rightarrow 2 \times 26^2 + 0 \times 26 + 6 = 19$
 - Mỗi số (tương ứng với 1 word) trong bản rõ được mã hoá bằng mã RSA với khoá công khai (d, n)
 - Áp dụng giải thuật bình phương và nhân để tính đồng dư luỹ thừa
2. Phá mã
 1. Phân tích n thành tích của 2 thừa số nguyên tố
 2. Tính $\varphi(n)$
 3. Tìm khoá bí mật e
 - Áp dụng thuật toán Oclit mở rộng
 4. Tìm bản rõ
 - Áp dụng giải thuật bình phương và nhân để tính đồng dư luỹ thừa

Nguyên tắc cấu tạo một hệ PK (trapdoor)

- Một hệ mã PKC có thể được tạo dựng trên cơ sở sử dụng một hàm kiểu one-way (1 chiều). Một hàm f được gọi là one-way nếu:
 1. Đối với mọi X tính ra $Y = f(X)$ là dễ dàng.
 2. Khi biết Y rất khó để tính ra X .
- Ví dụ. Cho n số nguyên tố p_1, p_2, \dots, p_n ta có thể dễ dàng tính được $N = p_1 * p_2 * \dots * p_n$, tuy nhiên khi biết N , việc tìm các thừa số nguyên tố của nó là khó khăn hơn rất nhiều
- Cần một hàm one-way đặc biệt, trang bị một trap-door (cửa bẩy), sao cho nếu biết trap-door này thì việc tính X khi biết $f(X)$ (tức là đi tìm nghịch đảo của f) là dễ, còn ngược lại thì khó
- Một hàm one-way có trap door như thế \rightarrow một hệ mã PKC
 - Lấy E_z (hàm sinh mã) là hàm one-way có trap-door
 - Trap-door chính là khoá mật, mà nếu biết nó thì có thể dễ dàng tính được cái nghịch đảo của E_z tức là biết D_z , còn nếu không biết thì rất khó tính được.

Trapdoor Knapsack dựa trên bài toán đóng thùng

- 1978, hai ông Merkle - Hellman đã đề xuất một thuật toán mã hoá PKC dựa trên bài toán ĐÓNG THÙNG như sau:
 - Cho 1 tập hợp các số dương a_i , $1 \leq i \leq n$ và 1 số T dương. Hãy tìm 1 tập hợp chỉ số $S \subset \{1, 2, \dots, n\}$ sao cho: $\sum_{i \in S} a_i = T$
- Bài toán này là một bài toán khó, theo nghĩa là chưa tìm được thuật toán nào tốt hơn là thuật toán thử-vét cạn
 - Thời gian xử lý vét cạn có thể tỉ lệ luỹ thừa theo kích thước input n .
 - VD: $(a_1, a_2, a_3, a_4) = (2, 3, 5, 7)$ $T = 7$.
Như vậy ta có 2 đáp số $S = (1, 3)$ và $S = (4)$.

Hệ PKC Merkle - Hellman

- Từ bài toán đóng thùng này chúng ta sẽ khảo sát các khả năng vận dụng để tạo ra thuật toán mã khôi PKC. Sơ đồ đầu tiên như sau:
 - Chọn một vector $a = (a_1, a_2, \dots, a_n)$ - được gọi là vector mang (cargo vector)
 - Với một khối tin $X = (X_1, X_2, X_3, \dots, X_n)$, ta thực hiện phép mã hoá như sau: $T = \sum a_i X_i$ (*)
 - Việc giải mã là: Cho mã T , vector mang a , tìm các X_i sao cho thoả mãn (*).
- Sơ đồ này thể hiện một hàm one-way với việc sinh mã rất dễ dàng nhưng việc giải mã là rất khó \rightarrow cơ sở xây dựng một trapdoor

Hệ PKC Merkle - Hellman

- Merkle sử dụng một mẹo là áp dụng một vector mang đặc biệt là vector siêu tăng (super-increasing)
 - thành phần $i+1$ là lớn hơn tổng giá trị của các thành phần đứng trước nó ($1 \div i$).
- Việc giải mã có thể diễn ra dễ dàng như ví dụ bằng số sau:

Vector mang siêu tăng: $a=(1,2,4,8)$

Cho $T=11$, ta sẽ thấy việc tìm $X=(X_1, X_2, X_3, X_4)$ sao cho $T = \sum a_i X_i$ là dễ dàng:

Đặt $T=T_0$

$$X_4=1 \quad T_0=T_0-X_4=3 \quad \rightarrow (X_1 \ X_2 \ X_3 \ 1)$$

$$X_3=0 \quad T_2=T_1=3 \quad \rightarrow (X_1 \ X_2 \ 0 \ 1)$$

$$X_2=1 \quad T_3=T_2-2=1 \quad \rightarrow (X_1 \ 1 \ 0 \ 1)$$

$$X_1=1 \quad \rightarrow (1 \ 1 \ 0 \ 1)$$

Hệ PKC Merkle - Hellman

- Bài toán được giải quyết dần qua các bước.
 - Ở bước i , tổng đích là T_i (tức là phải tìm các a_j để tổng bằng T_i). Ta đem so sánh T_i với thành phần lớn nhất trong phần còn lại của vector, nếu lớn hơn thì thành phần này được chọn tức là X_i tương ứng bằng 1, còn ngược lại thì X_i tương ứng bằng 0. Sau đó tiếp tục chuyển sang bước sau với $T_{i+1} = T_i - X_i$.
- Cần chủ động “ngụy trang” vector siêu tăng để chỉ có người chủ mới biết còn người ngoài không thể giải mã được.

Hệ PKC Merkle – Hellman: Cơ chế nguy trang

■ Tạo khoá:

Alice chọn một vector siêu tăng:

$$a' = (a_1', a_2', \dots, a_n')$$

a' được giữ bí mật tức là một thành phần của khoá bí mật

- Sau đó chọn một số nguyên $m > \sum a_i'$, gọi là mô-dul đồng dư và một số nguyên ngẫu nhiên ω , gọi là nhân tử, sao cho nguyên tố cùng nhau với m .
- Khoá công khai của Alice sẽ là vector a là tích của a' với nhân tử ω :

$$a = (a_1, a_2, \dots, a_n)$$

$$a_i = \omega \times a_i' \pmod{m}; i=1, 2, 3, \dots, n$$

- Còn khoá bí mật sẽ là bộ ba (a', m, ω)

Sơ đồ cụ thể Merkle-Hellman dựa trên bài toán đóng thùng.

■ Sinh mã:

- Khi Bob muốn gửi một thông báo X cho Alice, anh ta tính mã theo công thức:

$$T = \sum a_i X_i$$

■ Giải mã:

- Alice nhận được T, giải mã như sau:

Để bỏ lớp nguy trang cô ta trước hết tính ω^{-1} (là giá trị nghịch đảo của ω , tức là $\omega \times \omega^{-1} = 1 \text{ mod } m$, sẽ giới thiệu thuật toán tính sau), rồi tính $T' = T \times \omega^{-1} \text{ (mod } m)$

- Alice biết rằng $T' = a'.X$ nên cô ta có thể dễ dàng giải ra được X theo siêu tăng a'.

■ Chú thích: ở đây ta có

$$\begin{aligned} T' &= T \times \omega^{-1} = \sum a_i X_i \omega^{-1} = \sum a_i' \omega X_i \omega^{-1} \\ &= \sum (a_i' \omega \omega^{-1}) X_i = \sum a_i' X_i = a'.X \end{aligned}$$

■ Brute Force Attack (tấn công vũ phu)

- ❑ Với những kẻ không biết trapdoor (a' , m , ω), giải mã đòi hỏi phải tìm kiếm vét cạn qua 2^n khả năng của X.

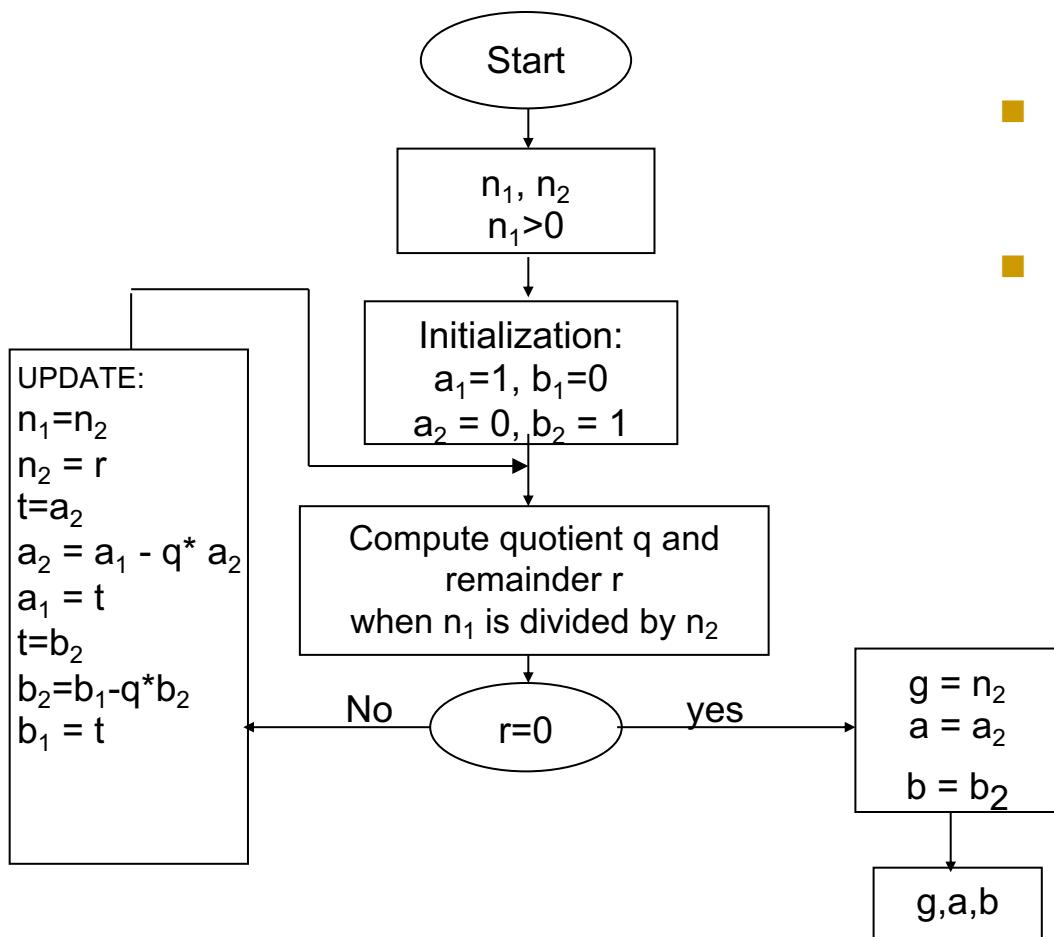
■ Sự đỗ vỡ của giải pháp dùng Knapsack (1982-1984).

- ❑ Shamir-Adleman đã chỉ ra chỗ yếu của GP này bằng cách đi tìm 1 cặp (ω', m') sao cho nó có thể biến đổi ngược a' về a (từ Public key về Private key).
- ❑ 1984, Brickell tuyên bố sự đỗ vỡ của hệ thống Knapsack với dung lượng tính toán khoảng 1 giờ máy Cray -1, với 40 vòng lặp chính và cỡ 100 trọng số.

Thuật toán tìm giá trị nghịch đảo theo modul đồng dư

- Việc xây dựng Knapsack với cửa bẫy đòi hỏi phải tính giá trị nghịch đảo của ω theo modul m .
- Thuật toán tìm $x = \omega^{-1} \text{ mod } m$, sao cho $x * \omega = 1 \pmod{m}$ được gọi là thuật toán GCD mở rộng hay Euclide mở rộng (GCD - Greatest common divisor - ước số chung lớn nhất).
 - Trong khi đi tìm USCLN của hai số nguyên n_1 và n_2 , người ta sẽ tính luôn các giá trị a, b sao cho $GCD(n_1, n_2) = a.n_1 + b.n_2$.
 - Từ đó suy ra nếu ta đã biết $(n_1, n_2) = 1$ thì thuật toán này sẽ cho ta tìm được a, b thoả mãn $a.n_1 + b.n_2 = 1$, tức là n_1 chính là nghịch đảo của a theo modulo n_2 (tức là m)

Hãy chứng minh tính đúng đắn của thuật toán GCD mở rộng



- Ví dụ tính bằng số: Tìm nghịch đảo của 11 theo modulo 39
- Đặt $n_1=39, n_2=11$ ta có bảng tính minh họa các bước như sau:

n_1	n_2	r	q	a_1	b_1	a_2	b_2
39	11	6	3	1	0	0	1
11	6	5	1	0	1	1	-3
6	5	1	1	1	-3	-1	4

Nhận xét chung về PKC

- Kể từ năm 1976, nhiều giải pháp cho PKC đã được nêu ra nhưng khá nhiều đã bị phá vỡ: chứng minh được là không an toàn.
- Một hệ thống PKC có thể đáp ứng 2 mục đích:
 - Bảo mật thông tin và truyền tin.
 - Chứng thực và chữ ký điện tử.
- Hai thuật toán đáp ứng các ứng dụng trên thành công nhất là RSA và El-Gamal.
- Nói chung PKC chậm, không thích hợp cho on-line encryption
 - Cần khi yêu cầu tính an toàn cao và chấp nhận tốc độ chậm.
 - Ngoài ra người ta thường sử dụng kết hợp PKC và SKC:
 - dùng PKC để tạo khóa bí mật thông nhất chung giữa hai bên truyền tin để thực hiện pha truyền tin chính bằng SKC sau đó.



ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Chữ ký số và hàm băm

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM HÀ NỘI

Số: 141 /ĐHSPHN-SĐH
V/v thông báo nhập học cao học khóa 2011-2013 (K21)

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

Hà Nội, ngày 17 tháng 2 năm 2012

THÔNG BÁO NHẬP HỌC

Cao học khóa 2011 – 2013 (tại Trường Đại học Cần Thơ)

Thi hành “Quy chế đào tạo trình độ thạc sĩ” của Bộ trưởng Bộ GD&ĐT (Thông tư số: 10/2011/QĐ-BGDDT ngày 28/2/2011), Trường Đại học Sư phạm Hà Nội thông báo như sau:

1. Anh (chị) đã được Hiệu trưởng Trường Đại học Sư phạm Hà Nội công nhận trúng tuyển cao học khóa 2011-2013 (K21), hệ đào tạo chính quy tập trung theo quyết định trúng tuyển số: 3383/QĐ-ĐHSPHN ngày 5/10/2011.

2. Ngày nhập học: **8 giờ 00', ngày 6 tháng 3 năm 2012 (Thứ ba)**, tại **Trường ĐH Cần Thơ**

3. Học viên tự sắp xếp nơi ở trong quá trình đào tạo.

4. Các khoản học viên phải đóng góp: **40.023.000 đồng**, trong đó:

4.1. Kinh phí đào tạo: **14.625.000 đồng**

4.2. Kinh phí do tổ ch

5. Thời gian nộp kinh phí:

Trường Đại học Sư pham
biên lai tài chính cho học v

- Đợt 1, ngày 6/3/2012

- Đợt 2, ngày 29/6/2012 (ngày thi hết chuyên đề đợt 1): **20.023.000 đồng**

6. Thủ tục đăng ký nhập học gồm:

- Quyết định cử đi học của Thủ trưởng cơ quan quản lý;

- 02 ảnh 4 x 6;

- Thủ tục nhập học: 100.000đ;

- Thẻ học viên, thẻ thư viện: 50.000đ.

Lưu ý: Sau 15 ngày kể từ ngày nhập học nếu anh (chị) không có mặt và không liên hệ với cơ sở đào tạo sẽ xem như anh (chị) bỏ không đăng ký theo khóa học.

Có gì chưa rõ, học viên liên hệ với Phòng Sau đại học, Trường ĐHSP Hà Nội, điện thoại: 043.7547823, máy lẻ 427; 0982.022.306 - chuyên viên: Đăng Ngọc Phúc; Trường ĐH Cần Thơ, Nguyễn Hữu Giao Tiên: 0907.289.008).

KT. HIỆU TRƯỞNG
PHÓ HIỆU TRƯỞNG

PGS.TS Trần Văn Ba

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM HÀ NỘI

Số: 141 /ĐHSPHN-SĐH

V/v thông báo nhập học cao học khóa 2011-2013 (K21)

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

Hà Nội, ngày 17 tháng 2 năm 2012

THÔNG BÁO NHẬP HỌC

Cao học khóa 2011 – 2013 (tại Trường Đại học Cần Thơ)

Thi hành “Quy chế đào tạo trình độ thạc sĩ” của Bộ trưởng Bộ GD&ĐT (Thông tư số: 10/2011/QĐ-BGDDT ngày 28/2/2011), Trường Đại học Sư phạm Hà Nội thông báo như sau:

1. Anh (chị) đã được Hiệu trưởng Trường Đại học Sư phạm Hà Nội công nhận trúng tuyển cao học khóa 2011-2013 (K21), hệ đào tạo chính quy tập trung theo quyết định trúng tuyển số: 3383/QĐ-ĐHSPHN ngày 5/10/2011.

2. Ngày nhập học: **8 giờ 00', ngày 6 tháng 3 năm 2012 (Thứ ba)**, tại **Trường ĐH Cần Thơ**

3. Học viên tự sắp xếp nơi ở trong quá trình đào tạo.

4. Các khoản học viên phải đóng góp: **40.023.000 đồng**, trong đó:

4.1. Kinh phí đào tạo: **14.625.000 đồng**

00 đồng
ong ĐH Cần Thơ
u trực tiếp và cắp
g

- Đợt 2, ngày 29/6/2012 (ngày thi hết chuyên đề đợt 1): **20.023.000 đồng**

6. Thủ tục đăng ký nhập học gồm:

- Quyết định cử đi học của Thủ trưởng cơ quan quản lý;

- 02 ảnh 4 x 6;

- Thủ tục nhập học: 100.000đ;

- Thẻ học viên, thẻ thư viện: 50.000đ.

Lưu ý: Sau 15 ngày kể từ ngày nhập học nếu anh (chị) không có mặt và không liên hệ với cơ sở đào tạo sẽ xem như anh (chị) bỏ không đăng ký theo khóa học.

Có gì chưa rõ, học viên liên hệ với Phòng Sau đại học, Trường ĐHSP Hà Nội, điện thoại: 043.7547823, máy lẻ 427; 0982.022.306 - chuyên viên: Đăng Ngọc Phúc; Trường ĐH Cần Thơ, Nguyễn Hữu Giao Tiên: 0907.289.008).

KT. HIỆU TRƯỞNG
PHÓ HIỆU TRƯỞNG

PGS.TS Trần Văn Ba



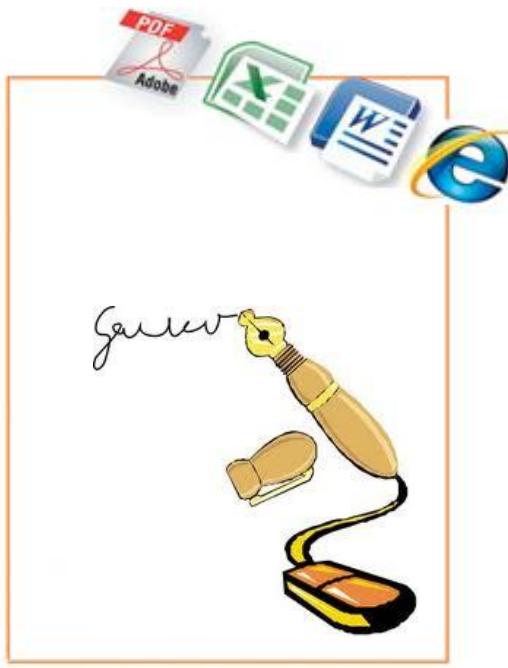
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Chữ ký viết tay

<p>BỘ GIÁO DỤC VÀ ĐÀO TẠO TRƯỜNG ĐẠI HỌC SƯ PHẠM HÀ NỘI Số: 141 /ĐHSPHN-SĐH V/v thông báo nhập học cao học khóa 2011-2013 (K21)</p> <p>THÔNG BÁO NHẬP HỌC Cao học khóa 2011 – 2013 (tại Trường Đại học Cần Thơ)</p> <p>Thi hành “Quy chế đào tạo trình độ thạc sĩ” của Bộ trưởng Bộ GD&ĐT (Thông tư số: 10/2011/QĐ-BGD&ĐT ngày 28/2/2011), Trường Đại học Sư phạm Hà Nội thông báo như sau:</p> <p>1. Anh (chị) đã được Hiệu trưởng Trường Đại học Sư phạm Hà Nội công nhận trúng tuyển cao học khóa 2011-2013 (K21), hệ đào tạo chính quy tập trung theo quyết định trúng tuyển số: 3383/QĐ-ĐHSPHN ngày 5/10/2011.</p> <p>2. Ngày nhập học: 8 giờ 00', ngày 6 tháng 3 năm 2012 (Thứ ba), tại Trường ĐH Cần Thơ</p> <p>3. Học viên tự sắp xếp nơi ở trong quá trình đào tạo.</p> <p>4. Các khoản học viên phải đóng góp: 40.023.000 đồng, trong đó:</p> <p>4.1. Kinh phí đào tạo: 14.625.000 đồng</p> <p>4.2. Kinh phí do tổ chức lớp học tại Trường Đại học Cần Thơ: 25.398.000 đồng</p> <p>5. Thời gian nộp kinh phí đào tạo và kinh phí do tổ chức lớp học tại Trường ĐH Cần Thơ</p> <p>Trường Đại học Sư phạm Hà Nội cử cán bộ đến Trường Đại học Cần Thơ thu trực tiếp và cấp biên lai tài chính cho học viên trong 2 đợt vào các ngày:</p> <p>- Đợt 1, ngày 6/3/2012 (ngày nhập học): 20.000.000 đồng - Đợt 2, ngày 29/6/2012 (ngày thi hết chuyên đề đợt 1): 20.023.000 đồng</p> <p>6. Thủ tục đăng ký nhập học gồm:</p> <p>- Quyết định cử đi học của Thủ trưởng cơ quan quản lý; - 02 ảnh 4 x 6; - Thủ tục nhập học: 100.000đ; - Thẻ học viên, thẻ thư viện: 50.000đ.</p> <p>Lưu ý: Sau 15 ngày kể từ ngày nhập học nếu anh (chị) không có mặt và không liên hệ với cơ sở đào tạo sẽ xem như anh (chị) bỏ không đăng ký theo khóa học. Có gì chưa rõ, học viên liên hệ với Phòng Sau đại học, Trường ĐHSP Hà Nội, điện thoại: 043.7547823, máy lẻ 427; 0982.022.306 - chuyên viên: Đặng Ngọc Phúc; Trường ĐH Cần Thơ, Nguyễn Hữu Giao Tiên- 0907.289.008).</p>	<p>CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM Độc lập - Tự do - Hạnh phúc</p> <p>Hà Nội, ngày 17 tháng 2 năm 2012</p> <p>KT, HIỆU TRƯỞNG PHÓ HIỆU TRƯỞNG</p>  <p>PGS.TS Trần Văn Ba</p>
---	--

1. Xác minh người tạo ra chữ ký

2. Xác thực nội dung được ký



Làm thế nào để định nghĩa một chữ ký cho các văn bản số, với các tính chất tương tự như chữ ký viết tay ?

Nội dung

❖ Chữ ký số

- Yêu cầu
- Tính chất
- Mô hình
- Chữ ký số dựa trên mật mã khóa công khai

❖ Hàm băm

- Định nghĩa
- Tính chất
- Ứng dụng vào chữ ký số

Chữ ký số



Yêu cầu của chữ ký số

- ❖ Xác thực nội dung được ký
 - Không thể thay đổi
 - Không thể dùng lại
- ❖ Xác minh người tạo ra chữ ký

Yêu cầu của chữ ký số

- ❖ Xác thực
- ❖ Xác thực nội dung được ký
 - Không thể thay đổi
 - Không thể thay đổi nội dung của bản tin đã được ký
 - Không thể dùng lại
- ❖ Xác minh người tạo ra chữ ký

Yêu cầu của chữ ký số

- ❖ Xác thực nội dung được ký
 - Không thể thay đổi
 - Không thể dùng lại
 - Không thể dùng lại chữ ký cho 1 bản tin khác
- ❖ Xác minh người tạo ra chữ ký

Yêu cầu của chữ ký số

- ❖ Xác thực nội dung được ký
 - Không thể thay đổi
 - Không thể dùng lại
- ❖ Xác minh người tạo ra chữ ký
 - Không thể làm giả
 - Không thể từ chối

Yêu cầu của chữ ký số

- ❖ Xác thực nội dung được ký
 - Không thể thay đổi
 - Không thể dùng lại
- ❖ Xác minh người tạo ra chữ ký
 - Không thể làm giả
 - A không thể giả mạo chữ ký của B
 - Không thể từ chối

Yêu cầu của chữ ký số

❖ Xác thực nội dung được ký

- Không thể thay đổi
- Không thể dùng lại

❖ Xác minh người tạo ra chữ ký

- Không thể làm giả
- Không thể từ chối
 - Nếu A đã ký thì sau đó A không thể chối bỏ là đã ký

Tính chất 1

- ❖ Là một chuỗi ký tự, có nội dung phụ thuộc vào nội dung bản tin được ký



Bản tin



hQlmaw9dfDAWEpmj9h8
7onwe1jd03nDf0

Chữ ký

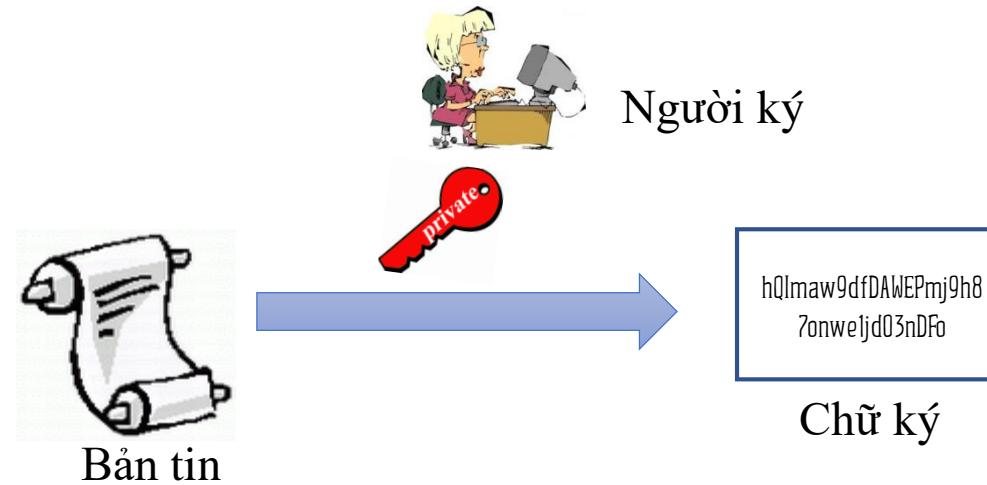


12khmlik0jh72nu8om

- ✓ *khó thay đổi*
 - ✓ *khó dùng lại*
- ⇒ *Xác thực nội dung bản tin được ký*

Tính chất 2

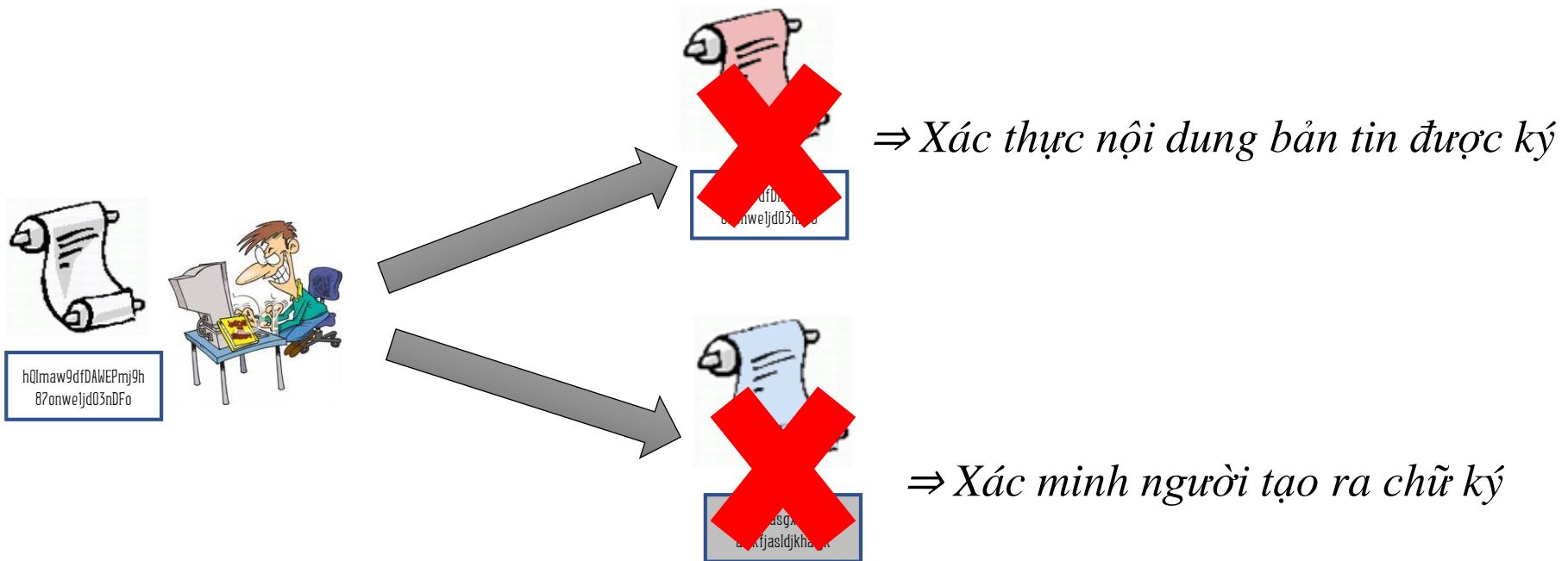
- ❖ Sử dụng thông tin mà chỉ có người ký mới có



- ✓ khó giả mạo
 - ✓ khó chối từ
- ⇒ Xác minh người tạo ra chữ ký

Tính chất 3

- ❖ Gần như không thể giả mạo chữ ký



So sánh chữ ký viết tay và chữ ký số

Chữ ký viết tay	Chữ ký số
Chữ ký cố định	Chữ ký thay đổi theo nội dung được ký
Gắn liền với nội dung được ký	Có thể tách khỏi nội dung được ký

Mô hình



S_A : Hàm sinh chữ ký

(m)

Chào B, tôi là A.
Dưới đây là
thông tin cá
nhân của tôi:

$S_A(m) = s$

Người ký A

(s)

hQlmaw9dfDAWEpmj9
h87onweijd03n

Chào B, tôi là A.
Dưới đây là
thông tin cá
nhân của tôi:

hQlmaw9dfDAWEpmj9
h87onweijd03n

true:

- + nội dung không bị thay đổi
- + chữ ký không bị giả mạo

false:

Nội dung không bị thay đổi
hoặc chữ ký không bị giả mạo

$V_A(m, s)$

Chào B, tôi là A.
Dưới đây là
thông tin cá
nhân của tôi:

hQlmaw9dfDAWEpmj9
h87onweijd03n

Chào B, tôi là A.
Dưới đây là
thông tin cá
nhân của tôi:

hQlmaw9dfDAWEpmj9
h87onweijd03n

Hàm xác nhận chữ ký: V_A



Người xác minh B

Yêu cầu

- ❖ S_A là hàm bí mật ; V_A là hàm công khai
- ❖ $V_A(m, s) = \text{true}$ nếu và chỉ nếu $S_A(m, s) = s$

Nhắc lại về mật mã khóa công khai



Chủ thể A



Khóa bí mật pr_A

Khóa công khai pb_A

E : Thuật toán mã hóa/giải mã

Tính đối hợp

$$m = E_{pr_A} \left(E_{pb_A} (m) \right) = E_{pb_A} \left(E_{pr_A} (m) \right)$$

Chữ ký số dựa trên mật mã khóa công khai

- ❖ Sinh chữ ký

$$s = E_{pr_A}(m)$$

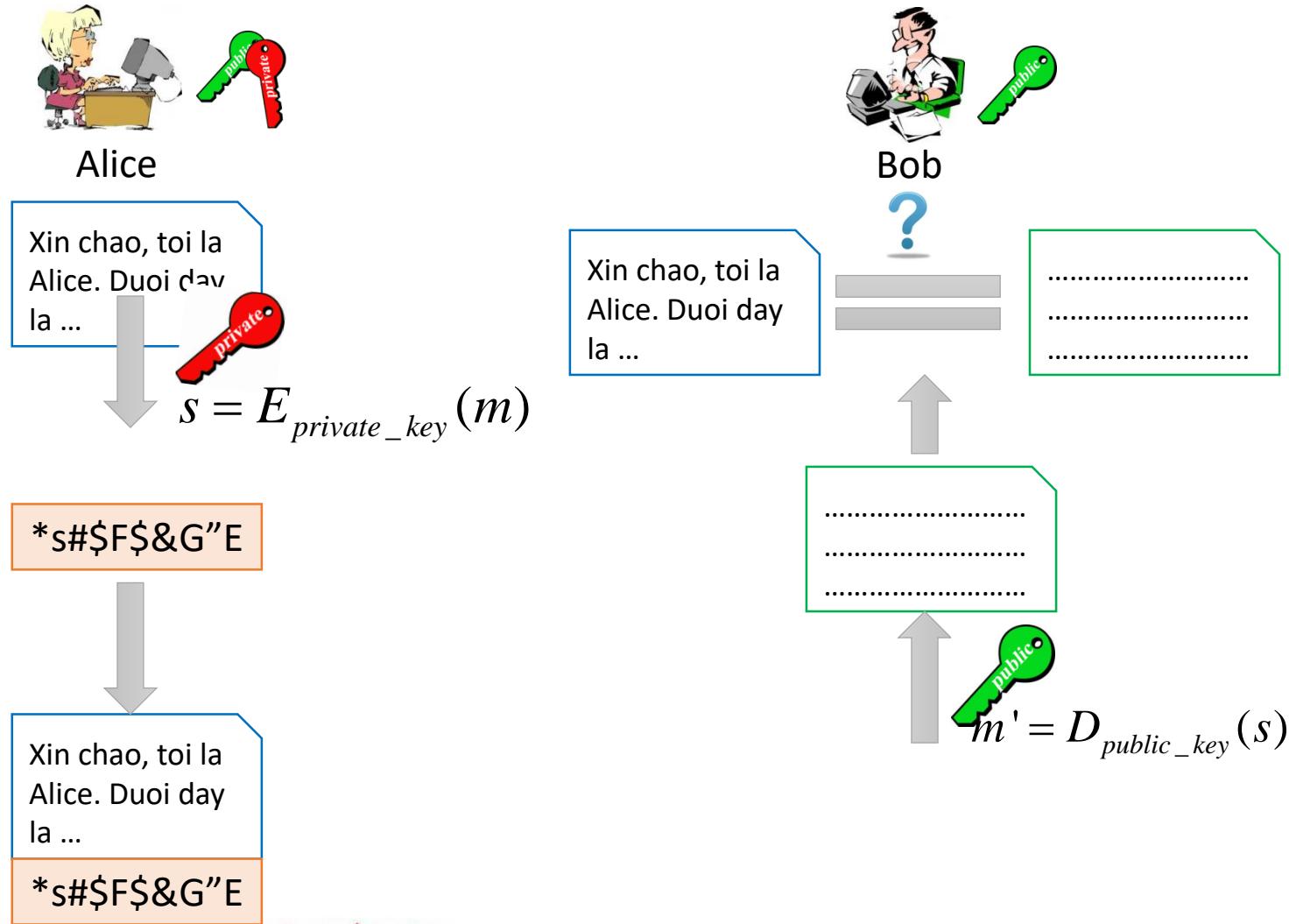
pr_A : Khóa bí mật của A

- ❖ Xác minh chữ ký

$$D_{pb_A} \text{ eq } m = \begin{cases} \text{true, if } D_{pb_A}(s) = m \\ \text{false, if } D_{pb_A}(s) \neq m \end{cases}$$

pb_A : Khóa công khai của A

Chữ ký số dựa trên mật mã khóa công khai



Vấn đề

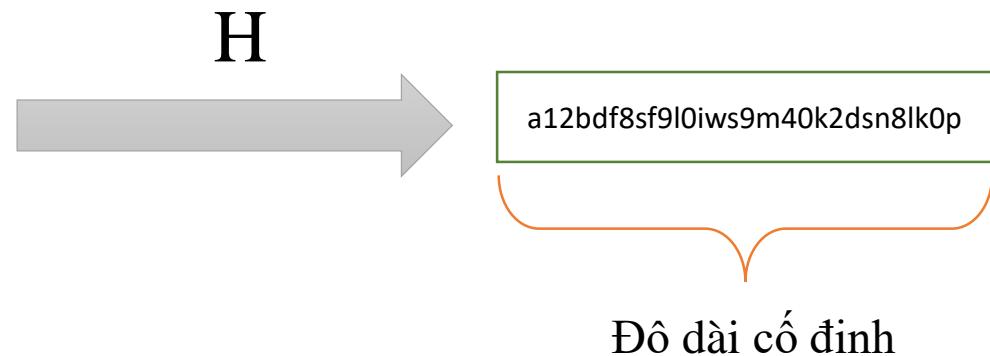
- ❖ Tốc độ chậm
- ❖ Kích thước chữ ký lớn
- ❖ Vấn đề với bản tin quá dài
 - ⇒ chia thành nhiều bản tin nhỏ và ký trên từng bản tin nhỏ
 - ⇒ tấn công: thay đổi thứ tự, thêm bớt các bản tin nhỏ

Hàm băm

Định nghĩa

- ❖ Là hàm biến đổi một chuỗi ký tự có độ dài bất kỳ thành một chuỗi ký tự có độ dài cố định
 - m : bản tin
 - $n = H(m)$: giá trị băm của m (*message digest, hash code*)

A digital signature is another means to ensure integrity, authenticity, and non-repudiation. A digital signature is derived by applying a mathematical function to compute the message digest of an electronic message or document, and then encrypt the result of the computation with the signer's private key. Recipients can verify the digital signature with the use of the sender's public key. A digital signature is another means to ensure integrity, authenticity,



Tính chất

❖ *Tính kiểm tra lỗi:*

- Thay đổi 1 bit bất kỳ của bản tin đầu vào sẽ thay đổi hoàn toàn giá trị đầu ra

Message: "A hungry brown fox jumped over a lazy dog"

SHA1 hash code: a8e7038cf5042232ce4a2f582640f2aa5caf12d2

Message: "A hungry brown fox jumped over a lazy dog"

SHA1 hash code: d617ba80a8bc883c1c3870af12a516c4a30f8fda

Tính chất

❖ *Tính một chiều:*

- Biết giá trị hàm băm \Rightarrow gần như không thể suy ngược giá trị bản tin

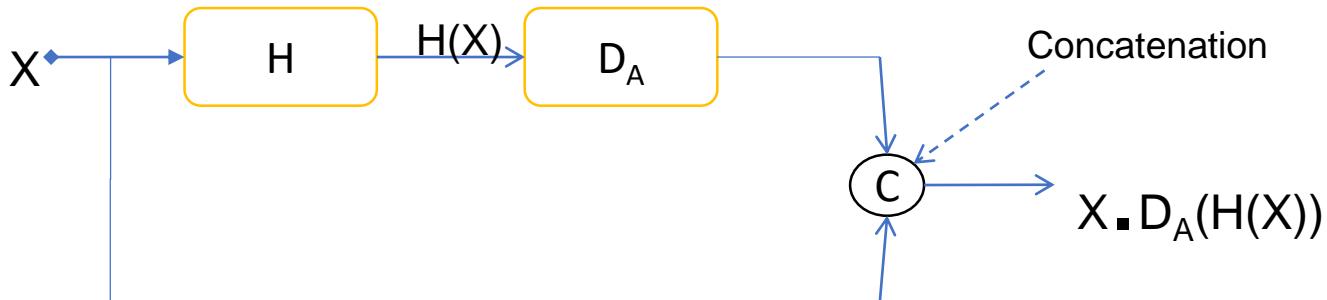
❖ *Tính không trùng lặp:*

- Với bản tin X cho trước \Rightarrow gần như không thể tìm được Y sao cho $H(x) = H(y)$
 - *Tính không trùng lặp yếu*
- Gần như không thể tìm được (X, Y) sao cho $H(x) = H(y)$
 - *Tính không trùng lặp mạnh*

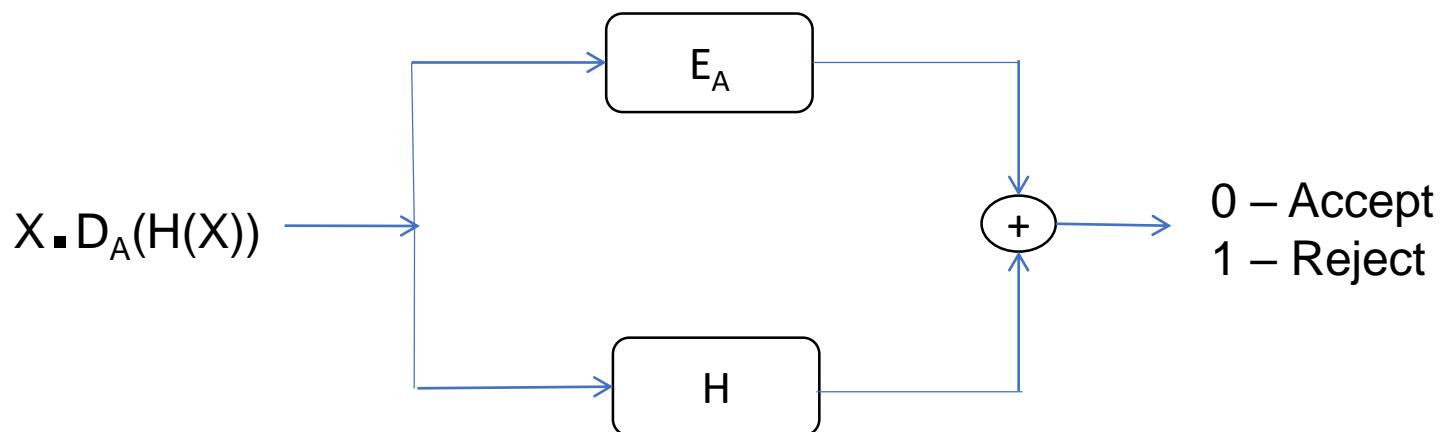
Chữ ký số với hàm băm

- ❖ Ý tưởng chính
 - Ký trên giá trị hàm băm
- ❖ Sinh chữ ký
 - $s = E_{pr_A}(H(m))$
- ❖ Xác minh chữ ký
 - $D_{pb_A} \text{ eq } H(m) = \begin{cases} \text{true}, & \text{if } D_{pb_A}(s) = H(m) \\ \text{true}, & \text{if } D_{pb_A}(s) \neq H(m) \end{cases}$

Chữ ký số với hàm băm



Signature Generator



Signature Verifier

Một số phương pháp tạo hàm băm

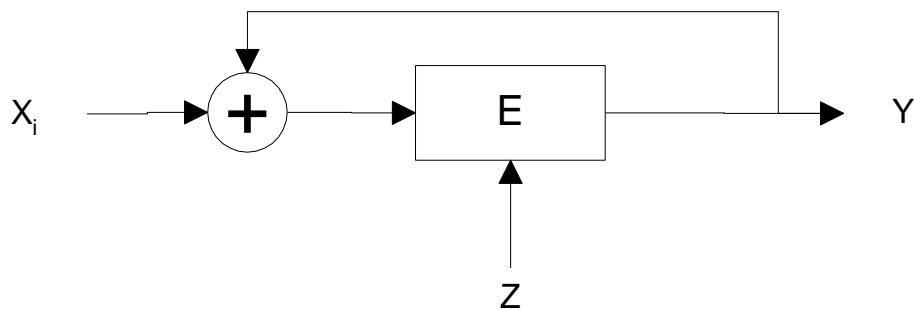
❖ Sử dụng SKC

- Ví dụ: Dùng SKC với chế độ CBC

❖ Sử dụng đồng dư (modulo arithmetic operations)

❖ Một số hàm băm thông dụng

- MD4, MD5, SHA



$$\begin{aligned} X &= X_1 \ X_2 \ X_3 \ \dots \ X_n \\ Y_i &= E_z(X_i \oplus Y_{i-1}) \\ H(X) &= Y_n \end{aligned}$$

Tính dụng độ của hàm băm

- ❖ Việc tránh dụng độ hoàn toàn là không thể (theo lý thuyết)
 - Nguyên lý Dirichlet: bỏ $n+1$ con thỏ vào n rọ \rightarrow ít nhất 2 con thỏ chung 1 roj
 - Nếu thử vét cạn $|Y|+1$ bản tin \rightarrow tìm được 2 bản tin có cùng giá trị băm ($H:X \rightarrow Y$)
- ❖ Thực tế
 - Để đảm bảo tính an toàn cho hàm băm thì cần chọn $|Y|$ đủ lớn sao cho việc vét cạn là không thể
 - Nhược điểm: $|Y|$ quá lớn sẽ làm tăng kích thước chữ ký, chậm quá trình tạo chữ ký
 - Tuy nhiên, vẫn khó tránh khỏi dụng độ

Tấn công theo kiểu nghịch lý ngày sinh

❖ Giá trị băm là 64 bits có đủ an toàn?

- Không gian tìm kiếm là 2^{64} → tương đối lớn để có thể thực hiện duyệt toàn bộ
- Tuy nhiên, có thể tấn công theo nghịch lý ngày sinh
 - Thực tế, kẻ tấn công chỉ cần tạo 2^{32} gói tin và có thể tấn công với xác suất thành công khá cao

Tấn công theo kiểu nghịch lý ngày sinh

- ❖ Mục tiêu: Với hàm băm H , tìm x, x' sao cho $H(x) = H(x')$
- ❖ Thuật toán:
 - Tạo tập S gồm q giá trị ngẫu nhiên thuộc X
 - Với mỗi $x \in S$, tính $h_x = H(x)$
 - Nếu $h_x = h_{x'}$ với $x' \neq x \rightarrow$ thành công trong tìm ra đụng độ (x, x')
- ❖ Xác suất thành công trung bình
$$\varepsilon = 1 - \exp\left(\frac{q(q-1)}{2|Y|}\right)$$
 - Giả sử Y có 2^m phần tử, chọn $q \approx 2^{m/2} \rightarrow \varepsilon$ xấp xỉ 0.5

Nghịch lý ngày sinh

- ❖ Cho một nhóm người → số người nhỏ nhất của nhóm đó, sao cho
 - Xác suất để có 2 người trong nhóm có cùng ngày sinh là 50%

Là 23

- ❖ → tại sao lại gọi là nghịch lý?
- ❖ Chứng minh
 - $1 - (1 - 1/365)(1 - 2/365) \dots (1 - 22/365) = 1 - 0.493 = 0.507$

Nghịch lý ngày sinh

- ❖ Cho 1 hàm băm có kích thước đầu ra là M . Lấy 1 tập ngẫu nhiên các bản rõ với độ dài q thì xác suất để tồn tại 2 bản rõ có cùng giá trị băm là: $1 - e^{\frac{-q(q-1)}{2M}}$
- ❖ Chứng minh
 - Xác suất để tất cả q bản rõ có giá trị băm khác nhau là:
 - $1 \times \frac{M-1}{M} \times \dots \times \frac{M-(q-1)}{M} = \left(1 - \frac{1}{M}\right) \dots \left(1 - \frac{q-1}{M}\right)$
 - Xác suất để tồn tại 2 bản rõ có cùng giá trị băm là:
 - $1 - \left(1 - \frac{1}{M}\right) \dots \left(1 - \frac{q-1}{M}\right) \approx 1 - e^{\left(-\frac{1}{M}\right) \times \left(-\frac{2}{M}\right) \times \dots \times \left(-\frac{q-1}{M}\right)} = 1 - e^{-\frac{q(q-1)}{2M}}$

Nghịch lý ngày sinh

- ❖ Cho 1 hàm băm có kích thước đầu ra là M . Lấy 1 tập ngẫu nhiên các bản rõ với độ dài q thì xác suất để tồn tại 2 bản rõ có cùng giá trị băm là: $1 - e^{\frac{-q(q-1)}{2M}}$
 - Với $q \approx \sqrt{2M \ln \frac{1}{1-\varepsilon}}$, xác suất xấp xỉ $1 - \varepsilon$
 - Với $q \approx 1.17\sqrt{M}$, xác suất xấp xỉ 0.5

MAC: mã xác thực bản tin

- ❖ Hàm băm được công khai, khoá được giữ bí mật giữa người gửi và người nhận
 - Người gửi tính $mac1 = MAC(M, H, K)$, và gửi cùng bản tin M
 - Người nhận tính $mac2 = MAC(M, H, K)$ và kiểm tra xem $mac1 = mac2$?
 - Nếu đúng \rightarrow gói tin được xác thực
 - Nếu không \rightarrow gói tin không được xác thực \rightarrow Không nhận/huỷ gói tin
- ❖ Không thể tính được MAC nếu không biết khoá bí mật giữa người gửi và người nhận
 - Cơ chế này vừa đảm bảo tính toàn vẹn và tính xác thực người gửi

Cryptography III

Public-key systems, digital signatures,
hash functions

Weaknesses of symmetric cryptosystems

- Managing and distributing shared secret keys is so difficult in a model environment with too many parties and relationships
 - N parties → $n(n-1)/2$ relationships → each manages $(n-1)$ keys
- No way for digital signatures
 - No non-repudiation service

Diffie-Hellman new ideas for PKC

- In principle, a PK cryptosystem is designed for a single user, not for a pair of communicating users
 - More uses other than just encryption
- Proposed in Diffie and Hellman (1976) “New Directions in Cryptography”
 - public-key encryption schemes
 - public key distribution systems
 - Diffie-Hellman key agreement protocol
 - digital signature

Diffie-Hellman's proposal

- Each user creates 2 keys: a secret (private) key and a public key → published for everyone to know
 - The PK is for encryption and the SK for decryption
 $X = D(z, E(Z, X))$
 - The SK is for creating signatures and the PK for verifying these signatures
 $X = E(Z, D(z, X)) \rightarrow D() \text{ for creating signatures, } E \rightarrow \text{verifying}$
- Also, called asymmetric key cryptosystems
 - Knowing the public-key and the cipher, it is computationally infeasible to compute the private key

RSA Algorithm

- Invented in **1978** by Ron **Rivest**, Adi **Shamir** and Leonard **Adleman**
 - Published as R L Rivest, A Shamir, L Adleman, "*On Digital Signatures and Public Key Cryptosystems*", Communications of the ACM, vol 21 no 2, pp120-126, Feb 1978
 - Security relies on the difficulty of factoring large composite numbers
- Essentially the same algorithm was discovered in 1973 by Clifford Cocks, who works for the British intelligence

Main idea

- Encryption and decryption functions are modulo exponential in the field $Z_n = \{0, 1, 2, \dots, n-1\}$
 - Encryption: $Y = X^e \text{ mod } n$ (or $\pm n$)
 - $a = b \pm n \rightarrow a = b + k * n, a \in Z_n, k = 1, 2, 3, \dots$ e.g. $7 = 37 \pm 10$
 - Decryption: $X = Y^d \pm n$
 - The clue is that e & d must be selected such that
 $X^{ed} = X \pmod{n}$

Main idea

- The way to create such e&d is by using this Euler theorem: $X^{\phi(n)} \equiv 1 \pmod{n}$
 - $\phi(n)$: the size of $Z_n^* = \{k: 0 < k < n \mid (k,n)=1\}$
 - $\phi(n)$ can be computed easily if knowing n factorization
 - $n = p \cdot q$, where p, q are primes $\rightarrow \phi(n) = (p-1)(q-1)$
 - First choose e then compute d s.t. $e \cdot d \equiv 1 \pmod{\phi(n)}$
or $d \equiv e^{-1} \pmod{\phi(n)}$, which will assure that
$$X^{ed} \equiv X^{k \cdot \phi(n) + 1} \equiv (X^{\phi(n)})^k \cdot X \equiv 1^k \cdot X = X \pmod{n}$$
- Note this works because we know n's factorization
 - From e we compute $d \equiv e^{-1} \pmod{\phi(n)}$ since we know $\phi(n)$, otherwise it is computational infeasible to compute d s.t. $X^{ed} \equiv 1 \pmod{n}$

RSA PKC

■ Key generation:

- Select 2 large prime numbers of about the same size, p and q
- Compute $n = pq$, and $\Phi(n) = (q-1)(p-1)$
- Select a random integer e , $1 < e < \Phi(n)$, s.t. $\gcd(e, \Phi(n)) = 1$
- Compute d , $1 < d < \Phi(n)$ s.t. $ed \equiv 1 \pmod{\Phi(n)}$
- **Public key: (e, n) and Private key: d**
 - Note: p and q must remain secret

RSA PKC (cont)

■ Encryption

- Given a message M , $0 < M < n$: $M \in Z_n - \{0\}$
- use public key (e, n) compute
 $C = M^e \text{ mod } n$, i.e. $C \in Z_n - \{0\}$

■ Decryption

- Given a ciphertext C , use private key (d) compute
 $M = C^d \text{ mod } n$

■ Why work?

- $(M^e \text{ mod } n)^d \text{ mod } n = M^{ed} \text{ mod } n = M$

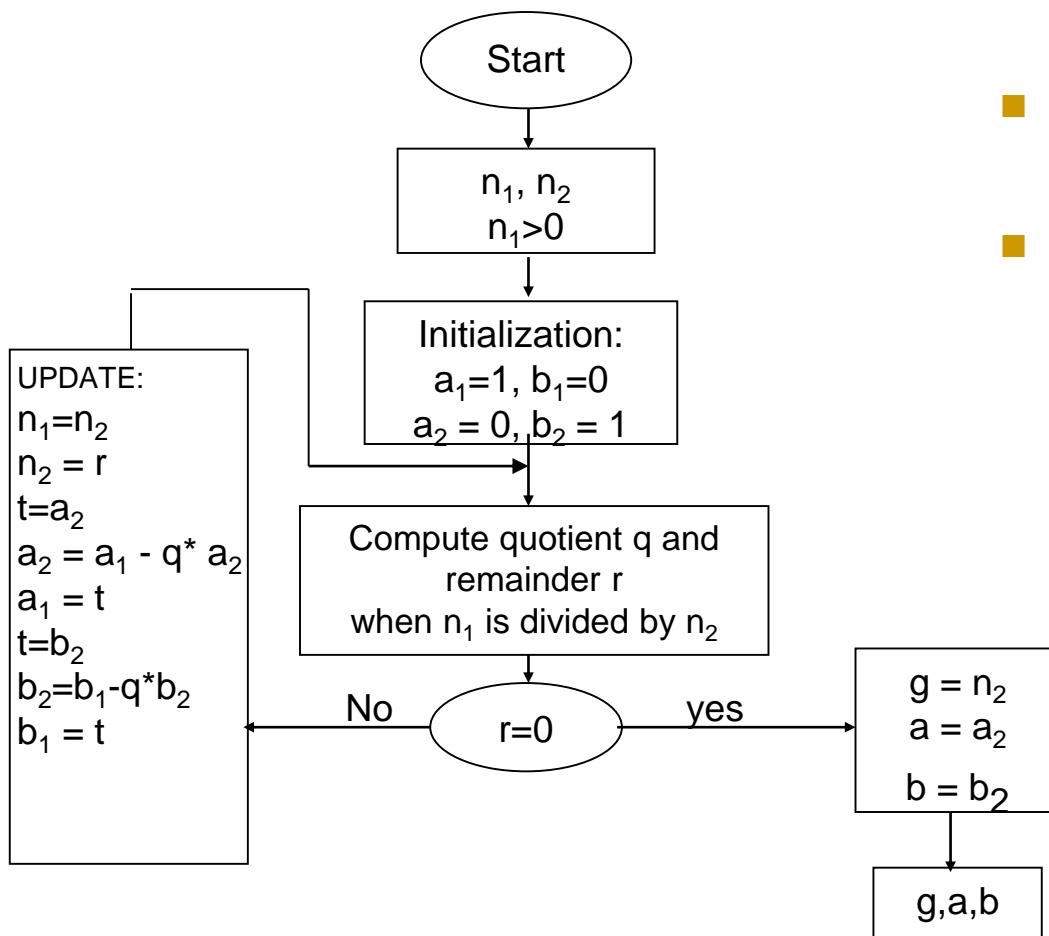
Example

- Parameters:
 - Select $p = 11$ và $q = 13$
 - $n=11*13=143$; $m= (p-1)(q-1) =10 *12=120$
 - Choose $e=37 \rightarrow \gcd(37,120)=1$
 - Using the algo gcd: $e*d = 1 \pm 120 \rightarrow d= 13$ ($e*d=481$)
- To encrypt a binary string
 - Split it into segments of u bit s.t. $2^u \leq 142 \rightarrow u = 7$. That is each segment present a number from 0 to 127
 - Compute $Y= X^e \pm 143$
E.g. For $X = (0000010) = 2$, we have
 $Y= E_Z(X) = X^{37} = 12 \pm 143 \rightarrow Y=(00001100)$
- Decryption: $X= D_Z(Y)=12^{13}=2 \pm 143$

Algorithm for computing modulo inverse

- Computing the inverse of ω by modulo m
 - Finding $x = \omega^{-1} \pmod{m}$ such that $x * \omega = 1 \pmod{m}$
 - Many applications such as in the Knapsack trapdoor
- Based on the extended GCD algorithm or the extended Euclidean algorithm (GCD: Greatest common divisor)
 - On finding the GCD of 2 numbers n_1 và n_2 , one will also compute a & b such that $GCD(n_1, n_2) = a \times n_1 + b \times n_2$.
 - If $\gcd(n_1, n_2) = 1$ then this e-GCD algorithm will find a, b to meet $a \times n_1 + b \times n_2 = 1$, i.e. n_1 is the inverse of a by modulo n_2

Homework: prove the correctness of this algorithm



- Numeric example: find the inverse of 11 by modulo 39
- Let $n_1=39$, $n_2=11$ then run the algo as in the following table:

n_1	n_2	r	q	a_1	b_1	a_2	b_2
39	11	6	3	1	0	0	1
11	6	5	1	0	1	1	-3
6	5	1	1	1	-3	-1	4

General remarks on PKC

- Since 1976, many PKC schemes had been proposed many was broken
- A PKC have two main applications
 - Hiding information (including secrete communication)
 - Authentication with digital signatures
- The two algorithms that are most successful are RSA và El-Gamal.
- In general PKC is very slow, not appropriate for on-line encryption
 - Not used for encrypting large volume of date but for special purposes.
 - PKC and SKC are used in combined:
 - Alice and Bob use a PKC system to create a shared secret key between them and then use a SKC system to encrypt the communicated data by using this secret key

RSA implementation

- n, p, q
 - The security of RSA depends on how large n is, which is often measured in the number of bits for n. Current recommendation is 1024 bits for n.
 - p and q should have the same bit length, so for 1024 bits RSA, p and q should be about 512 bits.
 - p-q should not be small
 - Way to select p and q
 - In general, select large numbers (some special forms), then test for primality
 - Many implementations use the Rabin-Miller test, (probabilistic test)

Factorization Problem

- Estimated time using the sieve algorithm

$$L(n) \approx 10^{9.7 + \frac{1}{50} \log_2 n}$$

- $\log_2 n$: the number of bits in representing n
- By 1996, for $n=200$, $L(n) \approx 55,000$ years.
- Using parallel computing, one can factorize a 129–digit number in 3 months by distributing the workload to the computers throughout the Internet at 1996-7
- Today, for applications requiring high security levels one should values of in 1024-bit or even 2048-bit.

Modulo Exponential

- Fast algorithm to compute exponential in Z_n (modulo n):
Computing X^α (modul n)
- Determine coefficients α_i in the binary representation of α :
$$\alpha = \alpha_0 2^0 + \alpha_1 2^1 + \alpha_2 2^2 + \dots + \alpha_k 2^k$$
- Loop in k rounds to compute these k modulo exponential, với $i=1,k$:

$$X^2 = X \times X$$

$$X^4 = X^2 \times X^2$$

...

$$X^{2^k} = X^{2^{k-1}} \times X^{2^{k-1}}$$

- Now compute $X^\alpha \bmod n$ by multiplying theses X^{2^i} computed in the previous steps but only with corresponding coefficients $\alpha_i = 1$:

$$(X^{2^i})^{\alpha_i} = \begin{cases} 1, & \alpha_i = 0 \\ X^{2^i}, & \alpha_i = 1 \end{cases}$$

Digital Signatures

■ Motivation

- Diffie-Hellman proposed the idea (1976)
- Simulation of the real-world into digital worlds
 - Paper contracts need signed to be valid so do electronic versions

■ The proofs conveyed in signatures

- Data integrity: information is original, not modified
- Authentication: The source of the info is correct, not impersonated

DS: how they work

- Digital Signature: a data string which associates a message with some originating entity.
- Digital Signature Scheme:
 - a signing algorithm: takes a message and a (private) signing key, outputs a signature
 - a verification algorithm: takes a (public) key verification key, a message, and a signature
- A DS is created based on a PK system
 - Alice signs message X by creating $Y=D_{z_A}(X)$, so the signed document now is $(X, Y=D_{z_A}(X))$.
 - Bob who receives (X, Y) , computes $X'=E_{z_A}(Y)$ then compare if $X=X'$ to confirm the document's validity

Non-repudiation

- We mention more on applications of DS
- Non-repudiation
 - The signer can't deny that his/her created the document
 - Only Alice knows z_A to create $(X, Y=D_{z_A}(X))$ but everyone else can verify
 - So we say the DS scheme provides non-repudiation

Public notary

- Motivation
 - Alice may lost her secret key or someone stole it → that bad guy can impersonate Alice to create documents with Alice signatures out of Alice's control
 - Alice can also deny a document truly signed by her in the past: Alice claims the document was impersonated by someone stealing her SK
- Solution: Public notary service
 - A third party – a public notary – can be hired for important documents
 - The trusted notary also signs on the same document, that is to create his signature on the concatenation of the document and Alice's signature

Proof of delivery (receipts)

- Motivation
 - The sender need proof that the receiver has already got his message
 - The receiver can't deny that once the sender got a receipt
- Solution: An adjudicated protocol
 - A → B: $Y = E_{Z_B}(D_{z_A}(X))$
 - B computes: $X' = E_{Z_A}(D_{z_B}(Y))$
 - When receiving Y, B computes and checks if $X' = X$ then signs on X' and pass to A as a receipt .
 - B → A: $Y = E_{Z_A}(D_{z_B}(X'))$
 - By computing $D_{z_A}(Y)$, A now gets $D_{z_B}(X')$, a B's signature on X
 - Only when A has Y she can consider that B has receive her doc
 - Later, B can not deny receiving X since A can prove otherwise by showing $D_{z_B}(Y)$)

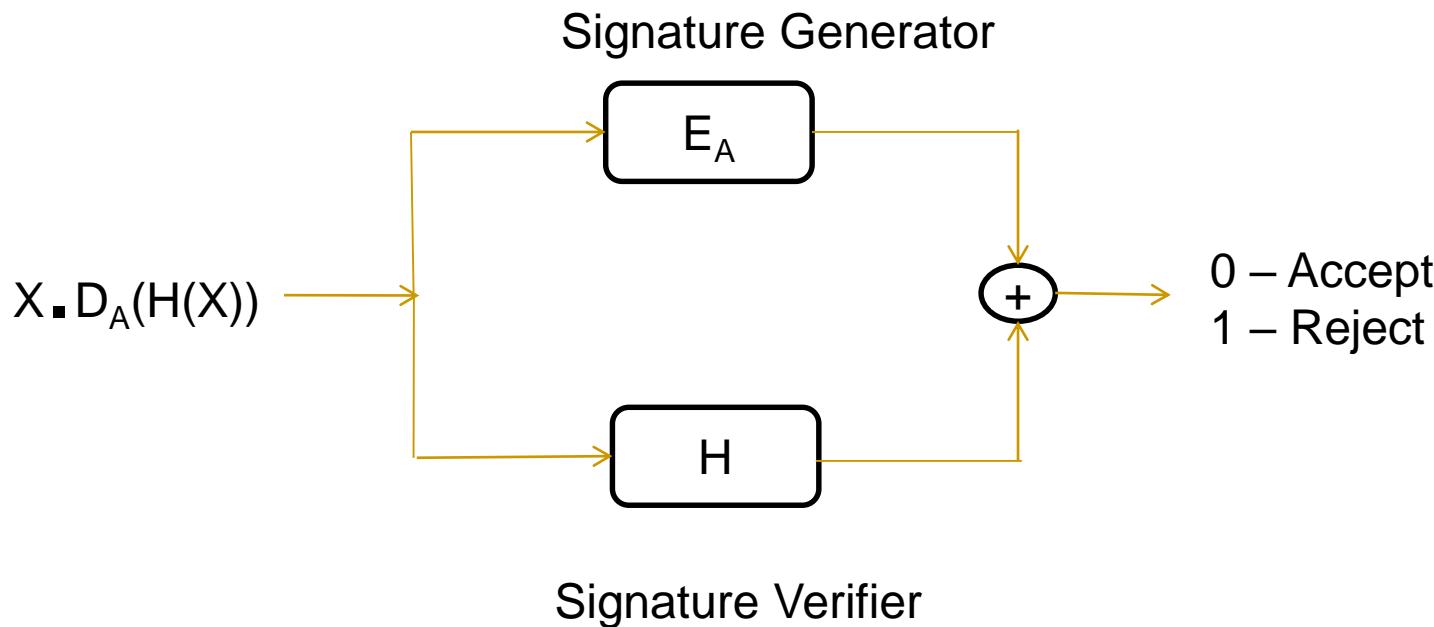
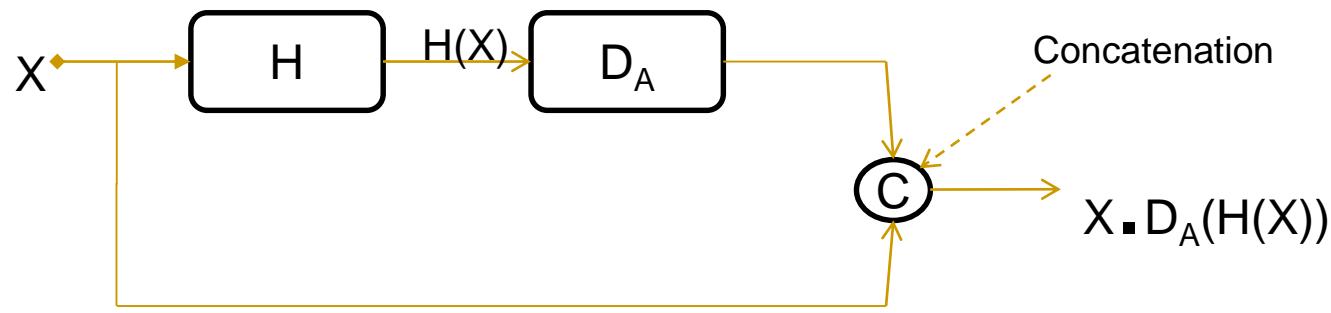
Weakness of the signature scheme mentioned so far

- When using a PKC to sign X , X can be long → splitting into blocks and signs
 $X = (X_1, X_2, X_3, \dots X_t) \rightarrow (SA(X_1), SA(X_2), SA(X_3), \dots SA(X_t))$
- This creates vulnerability to attack on manipulating blocks
 - The attacker can change order of blocks, remove/ add in a few
- Slow: PKC is already slow, now is run multiple times
- Signature is long, as long as the message itself.

Hash Functions

- A hash function H maps a message of variable length n bits to a fingerprint of fixed length m bits, with $m < n$.
 - This hash value is also called a digest (of the original message).
 - Since $n > m$, there exist many X which map to the same digest → collision.
- Applications
 - Digital signatures
 - Message authentication

DS schemes with hash functions



Main properties

Given a hash function $H: X \rightarrow Y$

- Long message \rightarrow short, fixed-length hash
- One-way property: given $y \in Y$
 - it is computationally infeasible to find a value $x \in X$ s.t. $H(x) = y$
- Collision resistance (collision-free)
 - it is computationally infeasible to find any two distinct values $x', x \in X$ s.t. $H(x') = H(x)$
 - This property prevent against signature forgery

Collisions

- Avoiding collisions is theoretically impossible
 - Dirichlet principle: $n+1$ rabbits into n cages → at least 2 rabbits go to the same cage
 - This suggest exhaustive search: try $|Y|+1$ messages then must find a collision ($H:X \rightarrow Y$)
- In practice
 - Choose $|Y|$ large enough so exhaustive search is computational infeasible.
 - $|Y|$ not too large or long signature and slow process
 - However, collision-freeness is still hard

Birthday attack

- Can hash values be of 64 bits?
 - Look good, initially, since a space of size 2^{64} is too large to do exhaustive search or compute that many hash values
 - However a birthday attack can easily break a DS with a 64-bit hash function
 - In fact, the attacker only need to create a bunch of 2^{32} messages and then launch the attack with reasonably high probability for success.

How is the attack

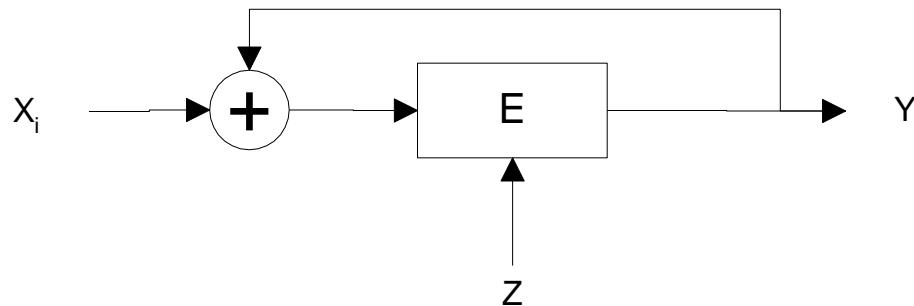
- Goal: given H , find x, x' such that $H(x)=H(x')$
- Algorithm:
 - pick a random set S of q values in X
 - for each $x \in S$, computes $h_x = H(x)$
 - if $h_x = h_{x'}$ for some $x' \neq x$ then collision found: (x, x') , else fail
- The average success probability is
 $\varepsilon = 1 - \exp(q(q-1)/2|Y|)$
 - Suppose Y has size 2^m , choose $q \approx 2^{m/2}$ then ε is almost 0.5!

Birthday paradox

- Given a group of people, the minimum number of people
 - such that two will share the same birthday with probability at least 50%
- is only 23 → why “paradox”
- Computing the chance
 - $1 - (1 - 1/365)(1 - 2/365) \dots (1 - 22/365) = 1 - 0.493 = 0.507$

Common techniques to build hash functions

- Using SKC
 - E.g. using SKC in CBC mode
- Using modulo arithmetic operations
- Specific designs
 - MD4, MD5, SHA



$$\begin{aligned} X &= X_1 \ X_2 \ X_3 \ \dots \ X_n \\ Y_i &= E_z(X_i \oplus Y_{i-1}) \\ H(X) &= Y_n \end{aligned}$$

MAC: message authentication code

- Hash function is public and the key shared between the sender and the receiver is secret
 - Sender computes $\text{mac1} = \text{MAC}(M, H, K)$ and sends it along with the message M
 - Receiver computes $\text{mac2} = \text{MAC}(M, H, K)$ and checks if $\text{mac1} = \text{mac2}$? Yes → the message is authentic; no => reject it
- The output of MAC can not be produced without knowing the secret key
 - So, this mechanism provides data integrity and ***source authentication***

Information Security

Van K Nguyen - HUT

Key Management

Overview

- Key exchange
 - Session vs. interchange keys
 - Classical, public key methods
 - Key generation
- Cryptographic key infrastructure
 - Certificates
- Key storage
 - Key escrow
 - Key revocation

Cryptographic protocols with applications

- Zero-Knowledge Protocols
- Subliminal channel
- Special signature schemes
- Electronic account-based payment systems
- Electronic anonymous cash systems
- Micropayment systems
- Secure multi-party computation
- Watermarking and DRM
- Diffie-Hellman key exchange
- Keberos
-

Notation

- $X \rightarrow Y : \{ Z \parallel W \} k_{X,Y}$
 - X sends Y the message produced by concatenating Z and W enciphered by key $k_{X,Y}$, which is shared by users X and Y
- $A \rightarrow T : \{ Z \} k_A \parallel \{ W \} k_{A,T}$
 - A sends T a message consisting of the concatenation of Z enciphered using k_A , A 's key, and W enciphered using $k_{A,T}$, the key shared by A and T
- r_1, r_2 nonces (nonrepeating random numbers)

Session key - Interchange key

- Alice wants to send a message m to Bob
 - Assume public key encryption
 - Alice know Bob's public key Z_B
- Proposed protocol
 - Alice generates a random cryptographic key k_s and uses it to encipher m
 - To be used for this message *only*
 - Called a *session key*
 - She enciphers k_s with Bob's public key Z_B
 - Z_B enciphers all session keys Alice uses to communicate with Bob
 - Called an *interchange key*
 - Alice sends Bob: $\{ m \} k_s // \{ k_s \} Z_B$

Why session key?

- Limits amount of traffic enciphered with single key
 - Standard practice, to decrease the amount of traffic an attacker can obtain
- Prevents some attacks
 - Example: Alice will send Bob message that is either “BUY” or “SELL”. Eve computes possible ciphertexts $\{ \text{“BUY”} \} Z_B$ and $\{ \text{“SELL”} \} Z_B$. Eve intercepts enciphered message, compares, and gets plaintext at once

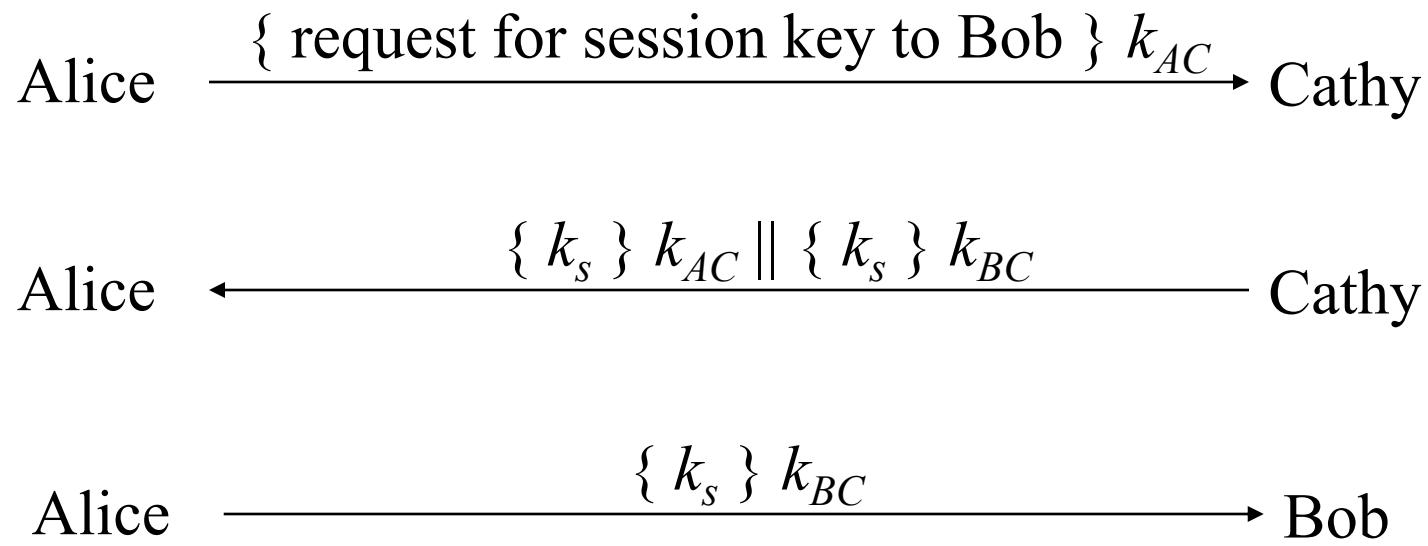
Key Exchange Algorithms

- Goal: Alice, Bob to get shared key (wo/ interchange key)
 - Key cannot be sent in clear
 - Attacker can listen in
 - Key can be sent enciphered, or derived from exchanged data plus data not known to an eavesdropper
 - Alice, Bob may use a trusted third party
 - All cryptosystems, protocols publicly known
 - Only secret data is the keys, ancillary information known only to Alice and Bob needed to derive keys
 - Anything transmitted is assumed known to attacker

Classical Key Exchange

- Bootstrap problem: how do Alice, Bob begin?
 - Alice can't send it to Bob in the clear!
- Assume trusted third party, Cathy
 - Alice and Cathy share secret key k_{AC}
 - Bob and Cathy share secret key k_{BC}
 - Use this to exchange shared key k_s

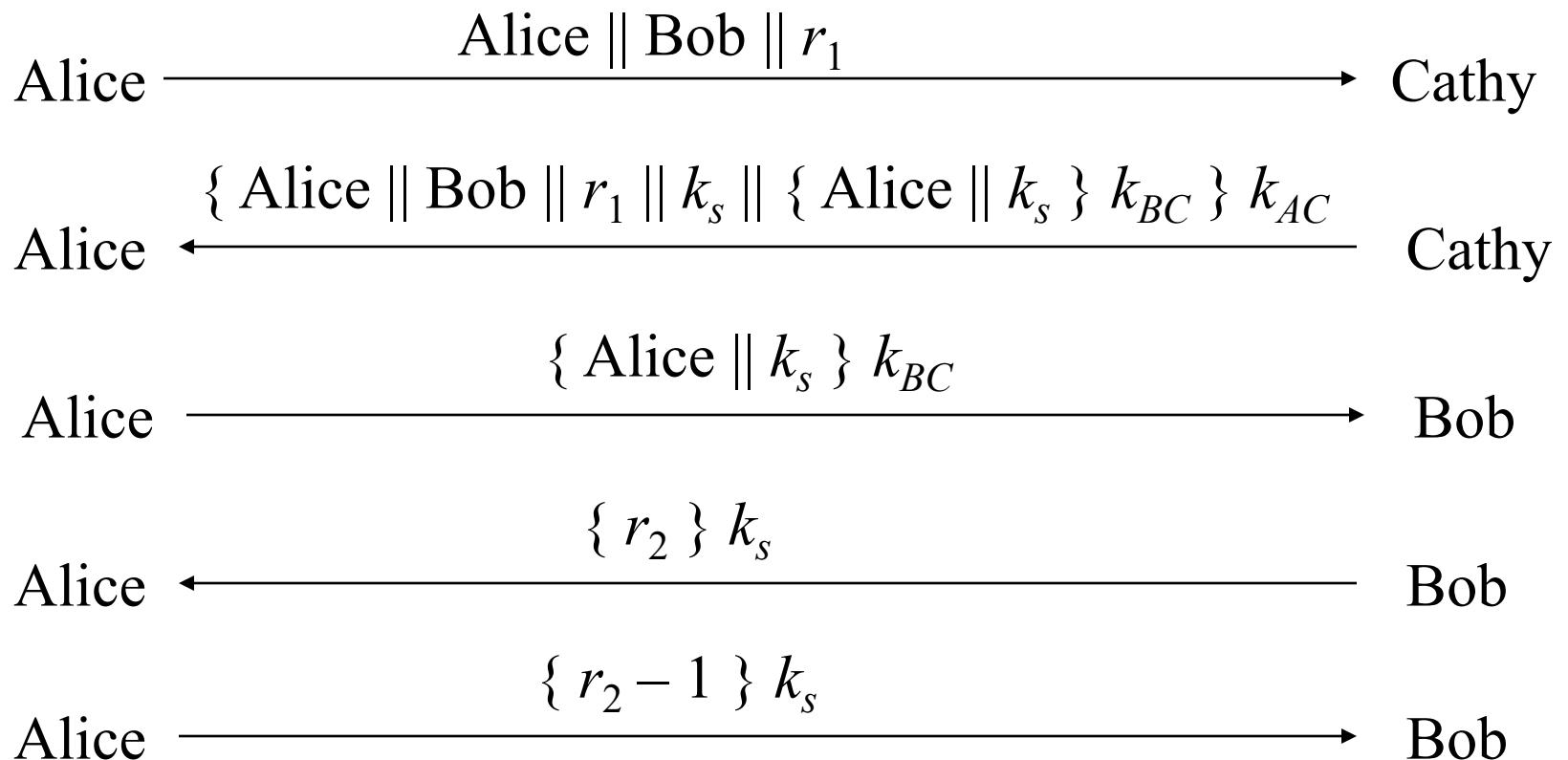
Simple Protocol



Problems

- How does Bob know he is talking to Alice?
 - Replay attack: Eve records message from Alice to Bob, later replays it; Bob may think he's talking to Alice, but he isn't
 - Session key reuse: Eve replays message from Alice to Bob, so Bob re-uses session key
 - Eve may fortunately get a session key dropped by Alice
- Protocols must provide authentication and defense against replay

Needham-Schroeder



Argument: Alice talking to Bob

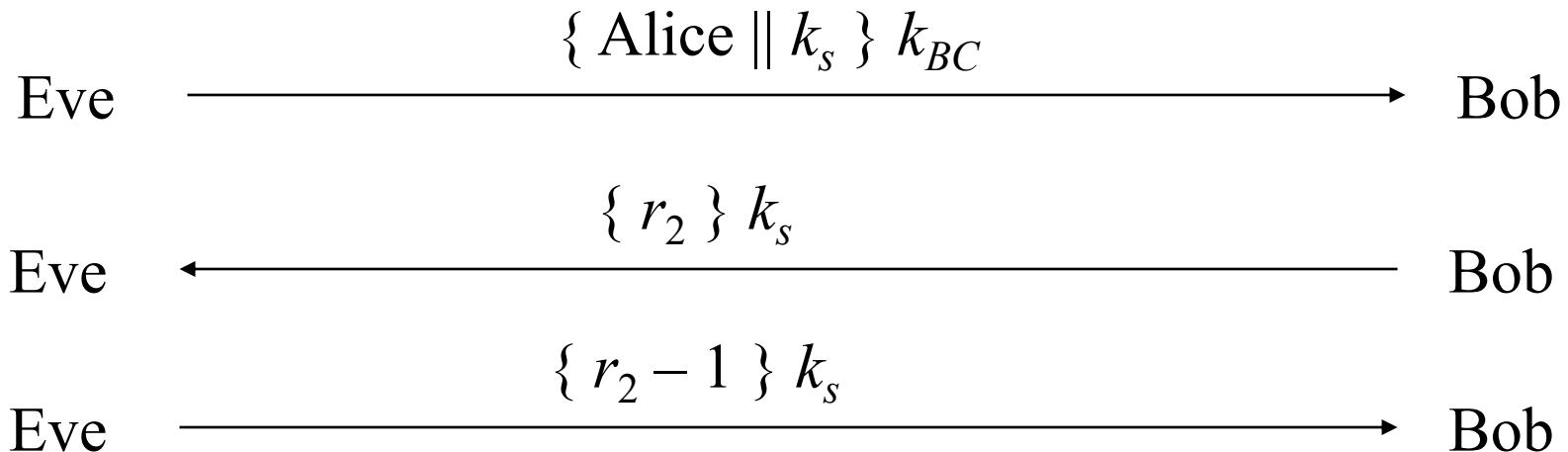
- Second message
 - Enciphered using key only she, Cathy knows
 - So Cathy enciphered it
 - Response to first message
 - As r_1 in it matches r_1 in first message
- Third message
 - Alice knows only Bob can read it
 - As only Bob can derive session key from message
 - Any messages enciphered with that key are from Bob

Argument: Bob talking to Alice

- Third message
 - Observe that: Enciphered using key only he and Cathy know
 - So Cathy enciphered it
 - Inside are the name Alice and the session key
 - Bob concludes that Cathy provided session key, saying Alice is the other party
- Fourth and Fifth messages:
 - Use session key to determine if it is replay from Eve
 - If not, Alice will respond correctly in fifth message
 - If so, Eve can't decipher r_2 and so can't respond, or responds incorrectly

Denning-Sacco Problem

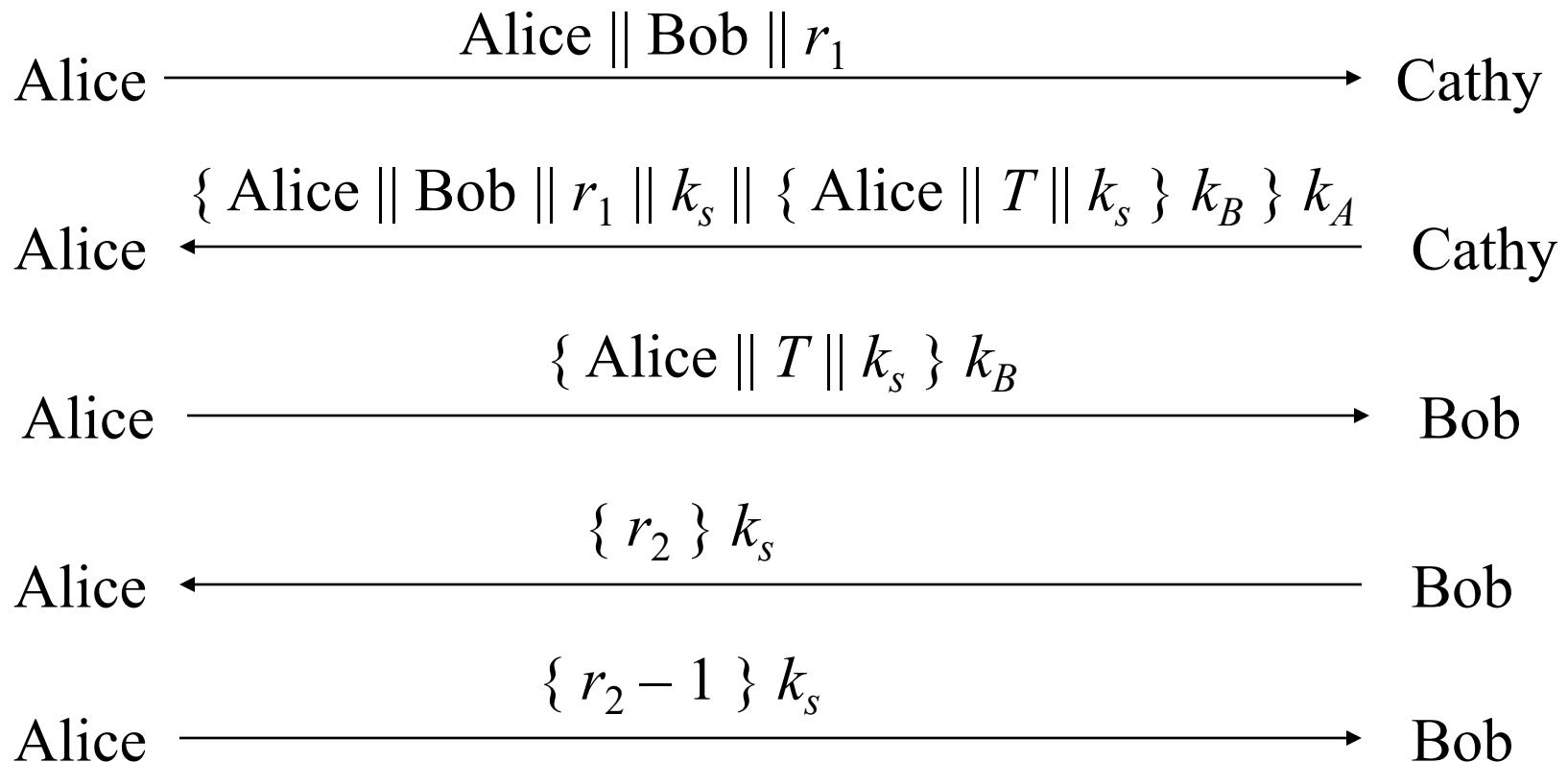
- Assumption: all keys are secret
- Question: suppose Eve can obtain session key.
How does that affect protocol?
 - In what follows, by chance Eve knows k_s



Solution

- In protocol above, Eve impersonates Alice
- Problem: replay in third step
 - First in previous slide
- Solution: use time stamp T to detect replay

Needham-Schroeder with Denning-Sacco Modification



Still weakness, anyway

- If clocks not synchronized, may either reject valid messages or accept replays
 - Parties with either slow or fast clocks vulnerable to replay
 - Resetting clock does *not* eliminate vulnerability
- Use Otway-Rees protocol (Bishop's text)

Kerberos

- Authentication system
 - Based on Needham-Schroeder with Denning-Sacco modification
 - Central server plays role of trusted third party (“Cathy”)
- Ticket
 - Issuer vouches for identity of requester of service
- Authenticator
 - Identifies sender

Idea

- User u authenticates to Kerberos server
 - Obtains ticket $T_{u,TGS}$ for ticket granting service (TGS)
- User u wants to use service s :
 - User sends authenticator A_u , ticket $T_{u,TGS}$ to TGS asking for ticket for service
 - TGS sends ticket $T_{u,s}$ to user
 - User sends A_u , $T_{u,s}$ to server as request to use s
- Details in Bishop's text

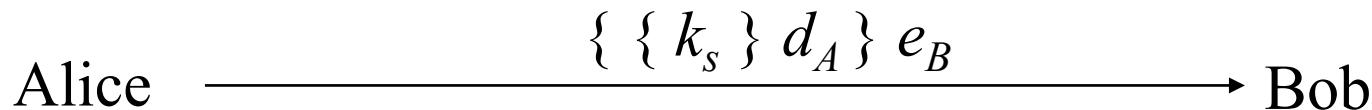
Public Key Key Exchange

- Here interchange keys known
 - e_A, e_B Alice and Bob's public keys known to all
 - d_A, d_B Alice and Bob's private keys known only to owner
- Simple protocol
 - k_s is desired session key



Problem and Solution

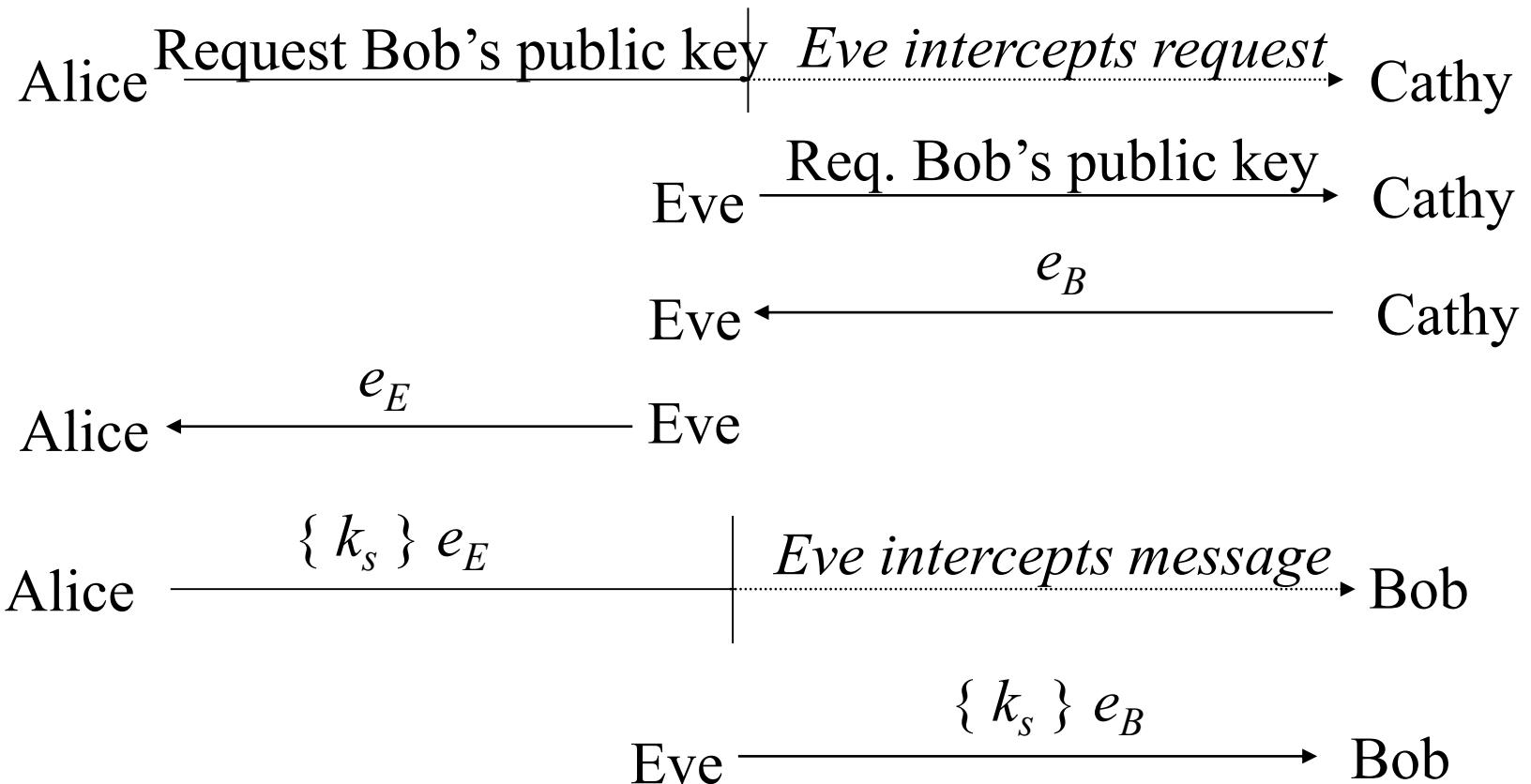
- Vulnerable to forgery or replay
 - Because e_B known to anyone, Bob has no assurance that Alice sent message
- Simple fix uses Alice's private key
 - k_s is desired session key
 - **Quiz:** Can Eve impersonate Alice and succeed to share a k_s with Bob?



Notes

- Can include message enciphered with k_s
- Assumes Bob has Alice's public key, and *vice versa*
 - If not, each must get it from public server
 - If keys not bound to identity of owner, attacker Eve can launch a *man-in-the-middle* attack (next slide; Cathy is public server providing public keys)
 - Solution to this (binding identity to keys) discussed later as public key infrastructure (PKI)

Man-in-the-Middle Attack



Key Generation

- Goal: generate keys that are difficult to guess
- Problem statement: given a set of K potential keys, choose one randomly
 - Equivalent to selecting a random number between 0 and $K-1$ inclusive
- Why is this hard: generating random numbers
 - Actually, numbers are usually *pseudo-random*, that is, generated by an algorithm

What is “Random”?

- *Sequence of cryptographically random numbers*: a sequence of numbers n_1, n_2, \dots such that for any integer $k > 0$, an observer cannot predict n_k even if all of n_1, \dots, n_{k-1} are known
 - Best: physical source of randomness
 - Random pulses
 - Electromagnetic phenomena
 - Characteristics of computing environment such as disk latency
 - Ambient background noise

What is “Pseudorandom”?

- *Sequence of cryptographically pseudorandom numbers*: sequence of numbers intended to simulate a sequence of cryptographically random numbers but generated by an algorithm
 - Very difficult to do this well
 - Linear congruential generators [$n_k = (an_{k-1} + b) \bmod n$] broken
 - Polynomial congruential generators [$n_k = (a_j n_{k-1}^j + \dots + a_1 n_{k-1} + a_0) \bmod n$] broken too
 - Here, “broken” means next number in sequence can be determined

Best Pseudorandom Numbers

- *Strong mixing function*: function of 2 or more inputs with each bit of output depending on some nonlinear function of all input bits
 - Examples: DES, MD5, SHA-1
 - Use on UNIX-based systems:
`(date; ps gaux) | md5`
where “ps gaux” lists all information about all processes on system

Cryptographic Key Infrastructure

- Goal: bind identity to key
- Classical: not possible as all keys are shared
 - Use protocols to agree on a shared key (see earlier)
- Public key: bind identity to public key
 - Crucial as people will use key to communicate with principal whose identity is bound to key
 - Erroneous binding means no secrecy between principals
 - Assume principal identified by an acceptable name

Certificates

- Create token (message) containing
 - Identity of principal (here, Alice)
 - Corresponding public key
 - Timestamp (when issued)
 - Other information (perhaps identity of signer)
- signed by trusted authority (here, Cathy)

$$C_A = \{ e_A \parallel \text{Alice} \parallel T \} d_C$$

Use

- Bob gets Alice's certificate
 - If he knows Cathy's public key, he can decipher the certificate
 - When was certificate issued?
 - Is the principal Alice?
 - Now Bob has Alice's public key
- Problem: Bob needs Cathy's public key to validate certificate
 - Problem pushed "up" a level
 - Two approaches: Merkle's tree, signature chains

Certificate Signature Chains

- Create certificate
 - Generate hash of certificate
 - Encipher hash with issuer's private key
- Validate
 - Obtain issuer's public key
 - Decipher enciphered hash
 - Recompute hash from certificate and compare
- Problem: getting issuer's public key

X.509 Chains

- Some certificate components in X.509v3:
 - Version
 - Serial number
 - Signature algorithm identifier: hash algorithm
 - Issuer's name; uniquely identifies issuer
 - Interval of validity
 - Subject's name; uniquely identifies subject
 - Subject's public key
 - Signature: enciphered hash

X.509 Certificate Validation

- Obtain issuer's public key
 - The one for the particular signature algorithm
- Decipher signature
 - Gives hash of certificate
- Recompute hash from certificate and compare
 - If they differ, there's a problem
- Check interval of validity
 - This confirms that certificate is current

Issuers

- *Certification Authority (CA)*: entity that issues certificates
 - Multiple issuers pose validation problem
 - Alice's CA is Cathy; Bob's CA is Don; how can Alice validate Bob's certificate?
 - Have Cathy and Don cross-certify
 - Each issues certificate for the other

Validation and Cross-Certifying

- Certificates:
 - Cathy<<Alice>>
 - Dan<<Bob>
 - Cathy<<Dan>>
 - Dan<<Cathy>>
- Alice validates Bob's certificate
 - Alice obtains Cathy<<Dan>>
 - Alice uses (known) public key of Cathy to validate Cathy<<Dan>>
 - Alice uses Cathy<<Dan>> to validate Dan<<Bob>>

Key Escrow

- *Key escrow system* allows authorized third party to recover key
 - Useful when keys belong to roles, such as system operator, rather than individuals
 - Business: recovery of backup keys
 - Law enforcement: recovery of keys that authorized parties require access to
- Goal: provide this without weakening cryptosystem
- Very controversial

Desirable Properties

- Escrow system should not depend on encipherment algorithm
- Privacy protection mechanisms must work from end to end and be part of user interface
- Requirements must map to key exchange protocol
- System supporting key escrow must require all parties to authenticate themselves
- If message to be observable for limited time, key escrow system must ensure keys valid for that period of time only

Components

- User security component
 - Does the encipherment, decipherment
 - Supports the key escrow component
- Key escrow component
 - Manages storage, use of data recovery keys
- Data recovery component
 - Does key recovery

Example: ESS, Clipper Chip

- Escrow Encryption Standard
 - Set of interlocking components
 - Designed to balance need for law enforcement access to enciphered traffic with citizens' right to privacy
- Clipper chip prepares per-message escrow information
 - Each chip numbered uniquely by UID
 - Special facility programs chip
- Key Escrow Decrypt Processor (KEDP)
 - Available to agencies authorized to read messages

Key Revocation

- Certificates invalidated *before* expiration
 - Usually due to compromised key
 - May be due to change in circumstance (e.g., someone leaving company)
- Problems
 - Entity revoking certificate authorized to do so
 - Revocation information circulates to everyone fast enough
 - Network delays, infrastructure problems may delay information

CRLs

- *Certificate revocation list* lists certificates that are revoked
- X.509: only certificate issuer can revoke certificate
 - Added to CRL
- PGP: signers can revoke signatures; owners can revoke certificates, or allow others to do so
 - Revocation message placed in PGP packet and signed
 - Flag marks it as revocation message

Kiểm tra giữa kỳ

1. Gọi $M = (\text{STT} \bmod 40) + 5$. Cho $p=11$, $q=17$ trong hệ RSA.

Hãy thực hiện các công việc sau:

- ❑ Xây dựng khoá công khai và bí mật của hệ (chú ý áp dụng thuật toán GCD mở rộng).
- ❑ Tính M^e của tin M
- ❑ Nếu sử dụng hệ này để làm chữ ký, xác định chữ ký cho M nói trên (chú ý dùng giải thuật nhạnh để tính lũy thừa đồng dư).
- ❑ Nếu muốn gửi một thông báo M vừa có đảm bảo xác thực vừa có tính mật, cần thực hiện cụ thể thế nào?

2. Cho biết ý nghĩa, công dụng của vector khởi đầu (initial vector) trong các chế độ mã khối.

- ❑ Cho bản rõ $M=M1||M2$. Sử dụng chế độ mã khối CBC với thuật toán DES, viết bản mật của tin trên theo $M1, M2, IV$ và hàm $\text{DES}_K()$.
- ❑ Tại sao nói chế độ ECB không nên dùng trong truyền tin bí mật, nhưng lại thích hợp để mã thông tin lưu trữ (trong các CSDL, các đĩa lưu ...).

- Sự khác biệt về ứng dụng mà chế độ CFB và OFB mang lại so với CBC?
- Trong chế độ CTR, một khóa phiên có thể duy trì bao lâu, giả thiết kích thước khóa là 64b và tốc độ truyền tin liên lạc hai bên là 100M/s?

Quản lý và trao đổi khoá

Ký hiệu

- $X \rightarrow Y : \{ Z \parallel W \} k_{X,Y}$
 - X gửi Y gói tin tạo bởi nối Z và W sau đó mã hoá với khoá $k_{X,Y}$, là khoá chung của X và Y
- $A \rightarrow T : \{ Z \} k_A \parallel \{ W \} k_{A,T}$
 - A gửi T 1 gói tin là nối của bản mã Z với khoá k_A , khoá bí mật của A , và bản mã W với khoá $k_{A,T}$, là khoá chung của A và T
- r_1, r_2 là các số ngẫu nhiên (số ngẫu nhiên không lặp lại)

Khoá phiên – khoá trao chuyêⁿ Session key - Interchange key

- Alice muốn gửi bản tin m cho Bob
 - Giả sử 2 bên sử dụng hệ mã công khai
 - Alice biết khoá công khai của Bob, Z_B
- Ví dụ về việc dung khoá phiên
 - Alice tạo 1 khoá ngẫu nhiên k_s dùng để mã hoá bản tin m
 - Khoá này chỉ dung để mã hoá bản tin
 - Được gọi là khoá phiên - *session key*
 - Alice mã hoá k_s với khoá công khai của Bob, Z_B
 - Z_B được dung để mã hoá toàn bộ khoá phiên trao đổi giữa Alice và Bob
 - Được gọi là khoá trao chuyêⁿ - *interchange key*
 - Alice gửi Bob: $\{ m \} k_s \{ k_s \} Z_B$

Session key - Interchange key

- Khoá phiên - session key
 - Gắn với 1 phiên giao dịch
 - Chỉ dùng để mã hoá thông tin, không dùng để xác thực chủ thẻ
 - Tại sao?
- Khoá trao chuyển - Interchange key
 - Gắn với 1 chủ thẻ
 - Có thể dùng để xác thực chủ thẻ

Tại sao cần dùng khoá phiên

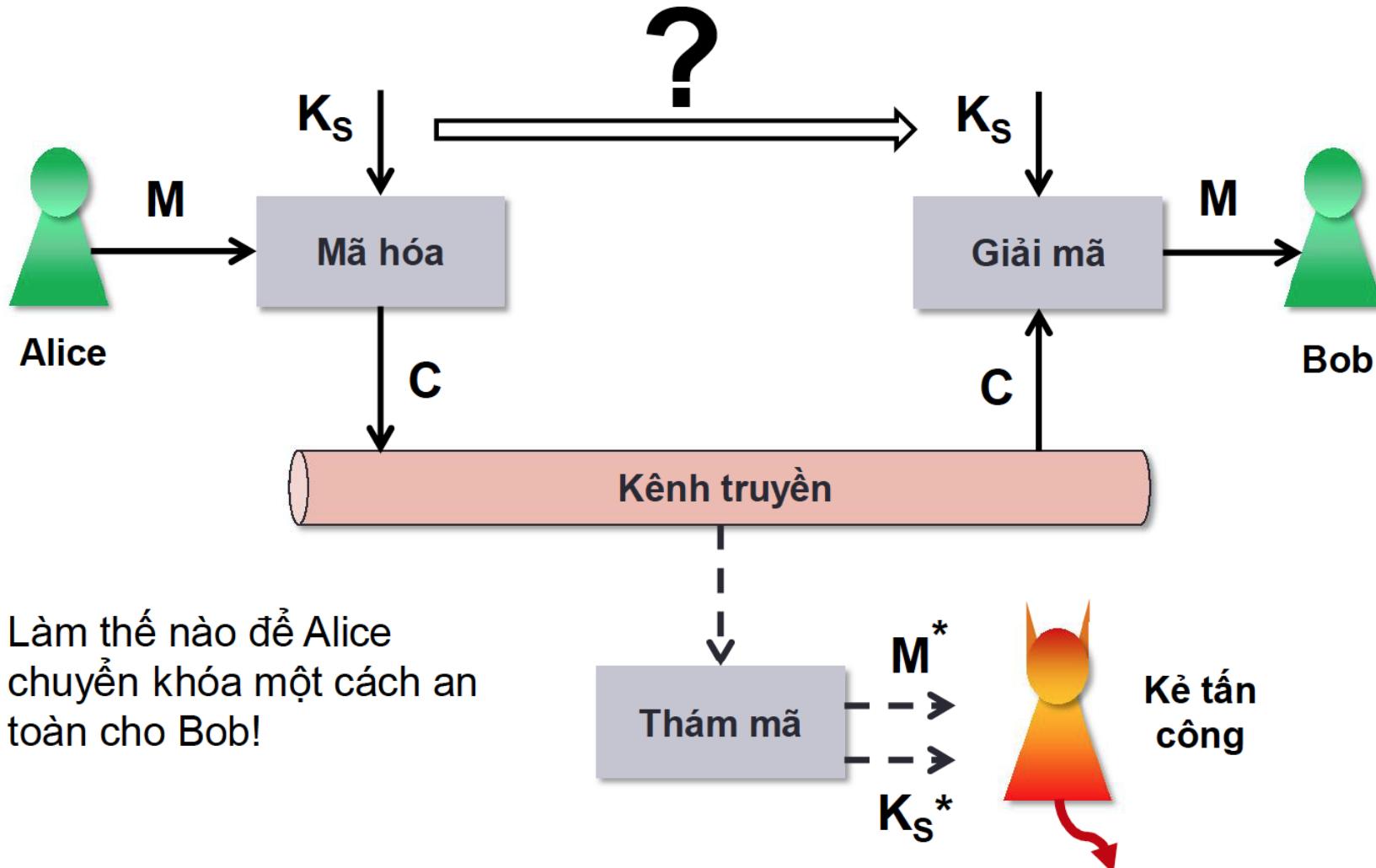
- Để hạn chế lượng thông tin được mã hoá với cùng 1 khoá
- Đảm bảo tính mới của khoá

Các thuật toán trao đổi khoá

- Mục tiêu: Alice, Bob thống nhất được khoá phiên (wo/ interchange key)
 - Khoá phiên không thể gửi dưới dạng plain text
 - Kẻ tấn công có thể đánh cắp
 - Khoá được mã hoá, hoặc được tạo ra từ thông tin trao đổi + một số thông tin bí mật giữa 2 bên (kẻ tấn công không thể biết)
 - Alice, Bob có thể sử dụng bên thứ 3 tin cây
 - Thuật toán cần đảm bảo an toàn với giả sử rằng toàn bộ thuật toán là public
 - Dữ liệu duy nhất không public là các khoá, các thông tin bí mật chỉ Alice và Bob biết
 - Giả sử rằng kẻ tấn công có thể lấy được các thông tin trên đường truyền

Các giao thức trao đổi khoá sử dụng hệ mã
đối xứng

Sơ đồ bảo mật sử dụng khoá đối xứng



Giao thức trao đổi khoá không tập trung

- Khóa chính: K_M đã được A và B chia sẻ an toàn
 - Làm thế nào vì đây chính là bài toán đang cần giải quyết?
 - Khóa chính được sử dụng để trao đổi khóa phiên K_S
- Khóa phiên K_S : sử dụng để mã hóa dữ liệu trao đổi
- Giao thức 1.1
 1. A → B: ID_A
 2. B → A: $E_{K_M}(ID_B, K_S)$
 3. (data) K_S
- Giao thức này đã đủ an toàn chưa?
 - Tấn công nghe lén
 - Tấn công thay thế
 - Tấn công giả mạo
 - Tấn công phát lại

Giao thức trao đổi khoá không tập trung

- Giao thức 1.2
 - Sử dụng các yếu tố chống tấn công phát lại (replay attack)
 - A → B: ID_A, R_1
 - B → A: $E_{K_M}(ID_B, K_S, R_1, R_2)$
 - A → B: $E_{K_S}(R_2)$
 - B: kiểm tra lại R_2
 - Hạn chế của phân phối khoá không tập trung là gì?

Giao thức trao đổi khoá tập trung

- Các thành phần tham gia
 - Alice, Bob
 - Cathy (Trọng tài): trung tâm phân phối khoá Key Distribution Center (KDC)
- Alice, bob có khoá chung với KDC trước khi tiến hành giao thức

Alice $\xrightarrow{\{ \text{request for session key to Bob} \} k_{AC}}$ Cathy

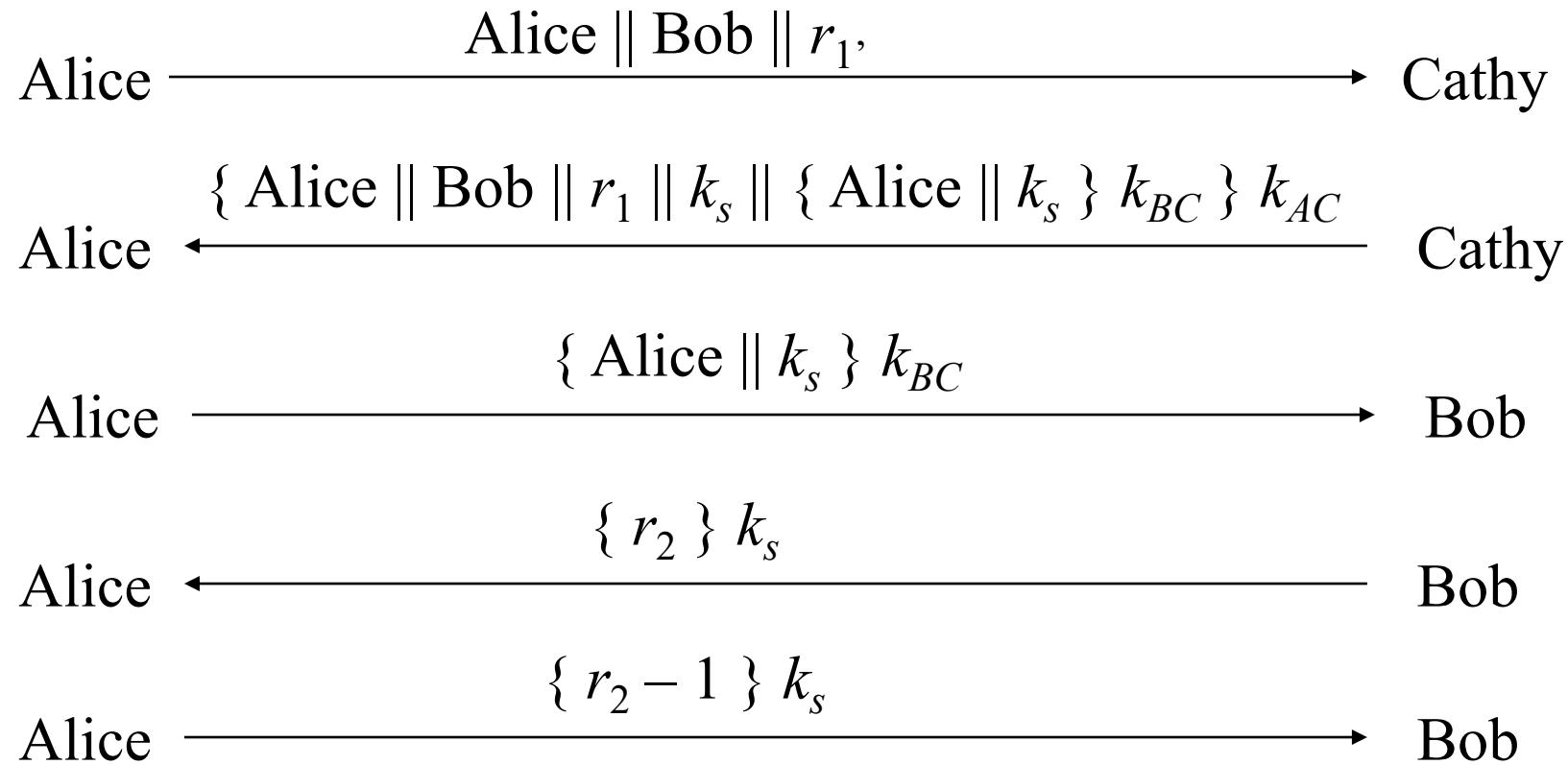
Alice $\xleftarrow{\{ k_s \} k_{AC} \parallel \{ k_s \} k_{BC}}$ Cathy

Alice $\xrightarrow{\{ k_s \} k_{BC}}$ Bob

Giao thức trao đổi khoá tập trung

- Vấn đề
 - Làm thế nào để Bob biết người đang nói chuyện với mình là Alice?
 - Replay attack: Eve có thể bắt các gói tin, sau đó gửi lại cho Bob, Bob có thể nghĩ đó là Alice, nhưng không phải
 - Sử dụng lại khoá phiên
 - Bằng cách nào đó Eve biết được khoá phiên trước đó
 - Eve sử dụng lại các gói tin ở step 3 → Bob sử dụng lại khoá phiên
- Giao thức phải có cơ chế xác thực và chống lại replay attack

Giao thức Needham-Schroeder



Giao thức Needham-Schroeder

- Ý nghĩa các bước
 - Gói tin thứ 2
 - Sử dụng khoá chỉ có Alice và cathy biết
 - Cathy và chỉ cathy mới giải mã được
 - Quan hệ với gói tin thứ 1
 - r_1 trong gói thứ 2 chính là r_1 trong gói thứ nhất
 - Gói tin thứ 3
 - Sử dụng khoá bí mật chỉ có bob và cathy biết
 - Chỉ có Bob mới đọc được
 - Chỉ có bob mới biết khoá k_s tất cả các gói tin được mã hoá bởi khoá k_s sau này là từ Bob

Giao thức Needham-Schroeder

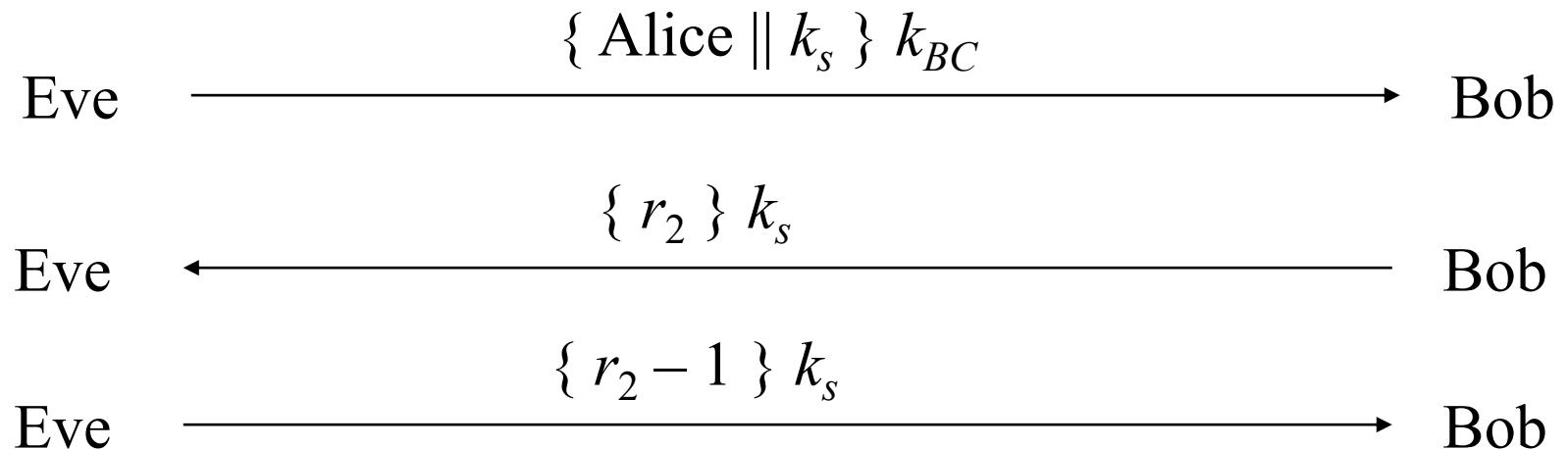
- Ý nghĩa các bước
 - Gói tin thứ 3
 - Sử dụng khoá bí mật chỉ có bob và cathy biết
 - Người mã hoá phải là Cathy
 - Phía trong có tên của Alice và khoá phiên
 - Bob kết luận rằng cathy là người cung cấp khoá, và cathy nói rằng khoá này là dùng cho phiên trao đổi với alice
 - Gói tin thứ 4 và 5:
 - Sử dụng khoá phiên để phát hiện nếu có tấn công replay attack từ eve
 - Nếu không phải là tấn công, Alice phản hồi gói tin thứ 5
 - Nếu là tấn công, Eve không thể giải mã được r_2 vì vậy không thể phản hồi đúng gói tin thứ 5

Giao thức Needham-Schroeder

- Nguy cơ tấn công vào Needham-Schroeder ?

Vấn đề Denning-Sacco

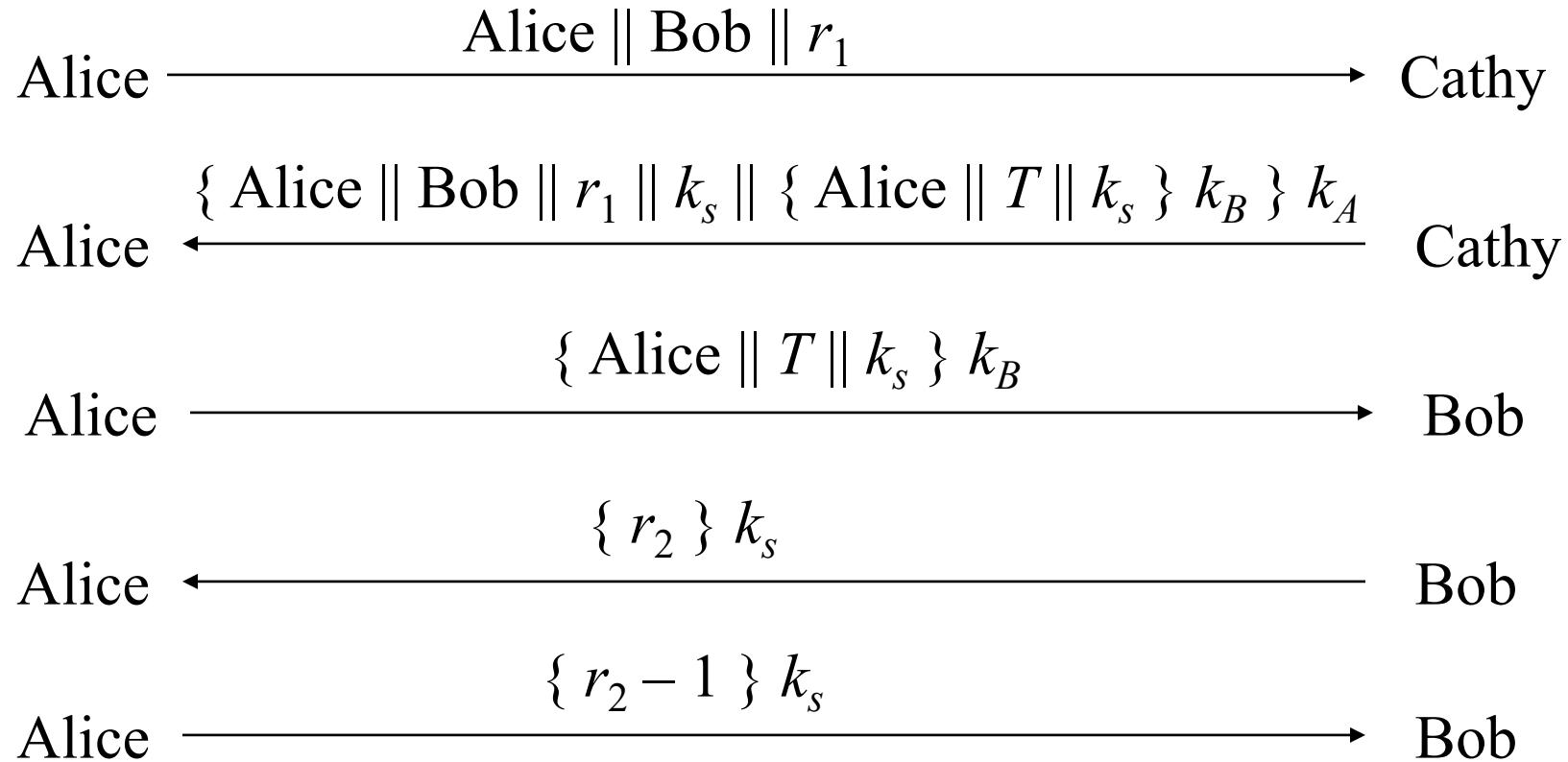
- Nếu bằng cách nào đó, sau đây Eve biết được khoá k_s . Eve tấn công như sau
 - Eve giả làm Alice, giả mạo bước thứ 3



- Giải pháp ?

Needham-Schroeder with Denning-Sacco Modification

- Thêm timestamp



Needham-Schroeder with Denning-Sacco Modification

- Vấn đề
 - Nếu đồng hồ của các bên không đồng kỳ, các bên có thể reject các gói tin hợp pháp hoặc ngược lại, chấp nhận các gói tin bất hợp pháp
- Sử dụng giao thức Use Otway-Rees
 - Tự tìm hiểu

Các giao thức trao đổi khoá sử dụng hệ mã công khai

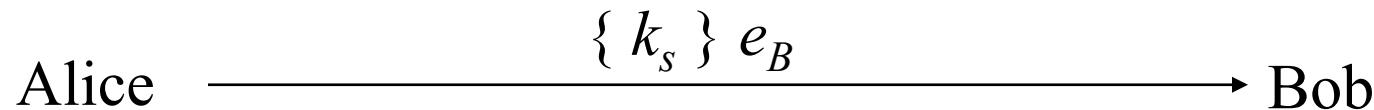
Trao đổi khoá với mã công khai

- Giả thiết
 - e_A, e_B : Khoá công khai của Alice và Bob \rightarrow công khai cho tất cả mọi người
 - d_A, d_B : Khoá bí mật của Alice and \rightarrow chỉ Alice và Bob biết

Giao thức phân phối khóa không tập trung

- Giao thức đơn giản

- k_s : khoá phiên



- Vấn đề

- Bởi vì e_B là công khai \rightarrow Bob không biết được đối phương có phải là Alice không

- Giải pháp đơn giản

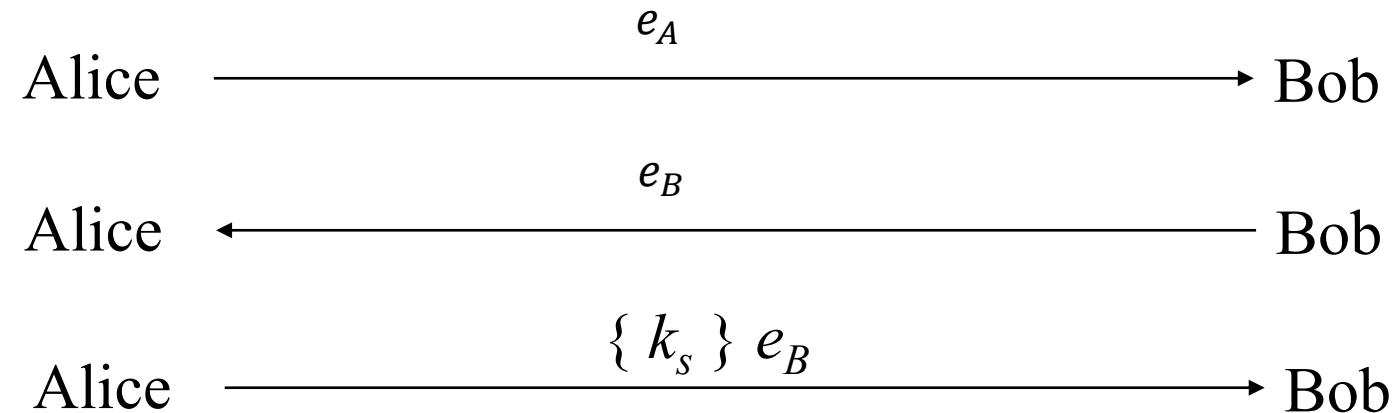
- Sử dụng mã bí mật của Alice



- Giao thức này có lỗ hổng nào không ?

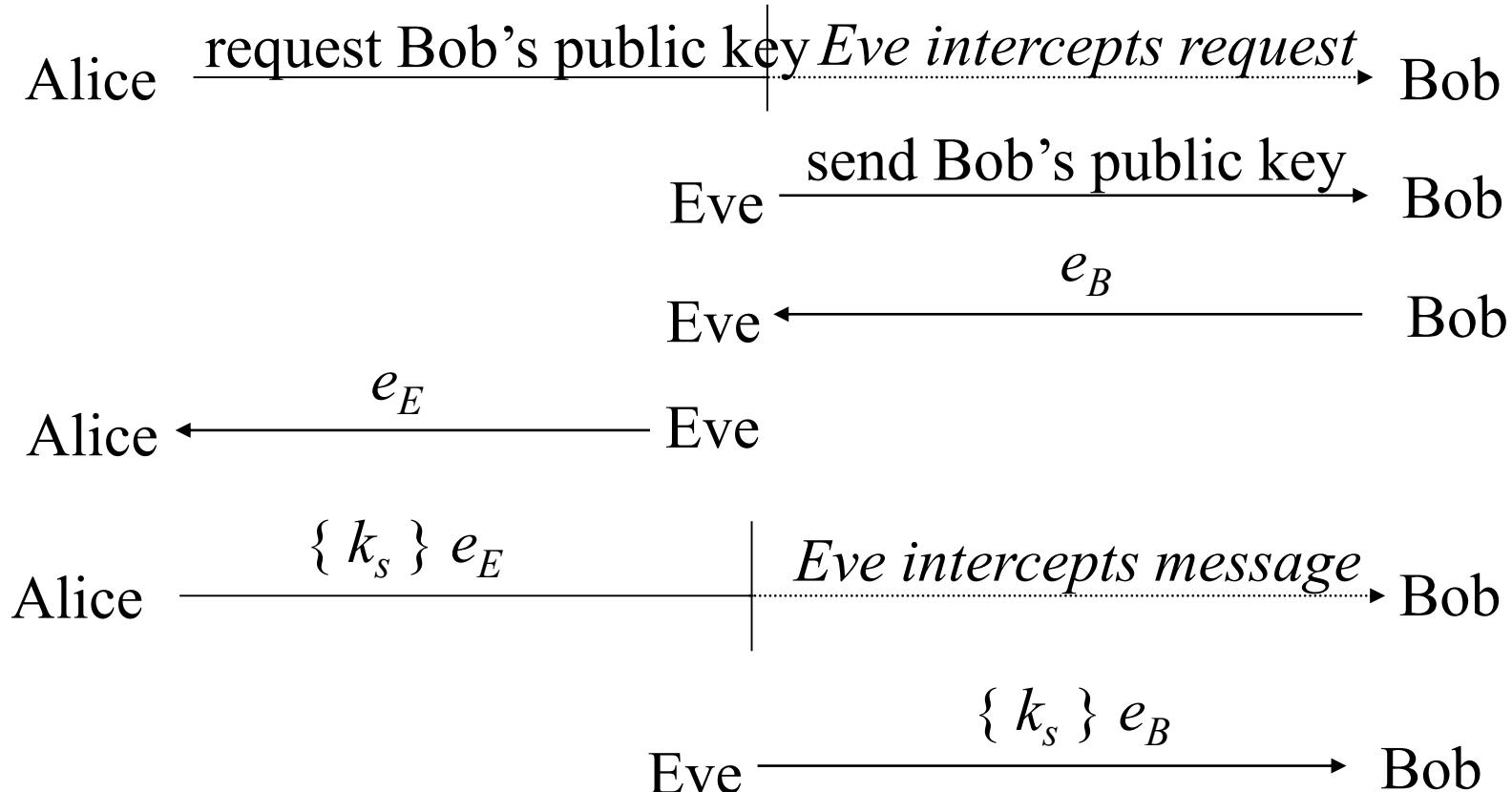
Giao thức phân phối khóa không tập trung

- Nếu Alice và Bob không biết khoá công khai của nhau
 - Trước khi giao dịch, Alice và Bob gửi khoá công khai trực tiếp cho nhau



Tấn công kẻ ngồi giữa Man-in-the-middle attack

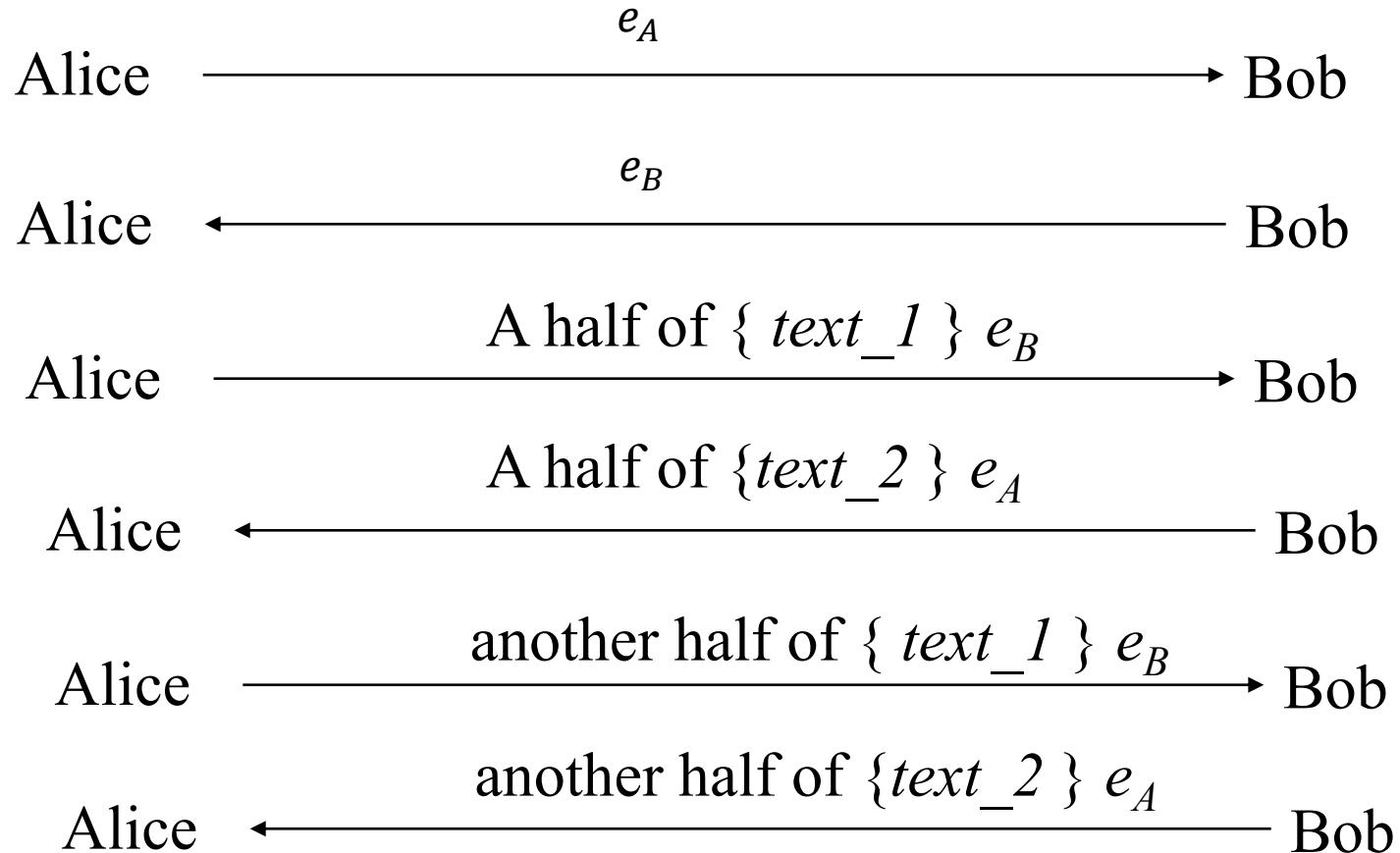
- Eve giả vờ là Bob khi nói chuyện với Alice, giả vờ là Alice khi nói chuyện với Bob



Chống tấn công kẻ ngồi giữa

- Interlock: đề xuất bởi Ron Rivest and Adi Shamir
 1. Alice gửi khoá công khai của Alice cho Bob
 2. Bob gửi khoá công khai của Bob cho Alice
 3. Alice mã hoá gói tin của mình bằng khoá công khai của Bob. Alice gửi 1 nửa bản mã cho Bob
 4. Bob mã hoá gói tin của mình bằng khoá công khai của Alice. Bob gửi 1 nửa bản mã cho Alice
 5. Alice gửi nửa còn lại của bản mã cho Bob
 6. Bob ghép 2 nửa bản mã lại, giải mã dùng khoá bí mật của Bob. Nếu bản mã được giải mã thành công Bob gửi lại nửa bản mã còn lại ở step 4 cho Alice
 7. Alice ghép 2 nửa bản mã nhận được từ Bob và giải mã
 8. Câu hỏi
 1. Tại sao interlock chống được man in the middle attack

Solution for Man-in-the-Middle Attack



Giao thức phân phối khóa tập trung

- Sử dụng bên thứ 3 tin cậy – PKA (Public Key Authority)
 - Có cặp khóa (e_{PKA}, d_{PKA})
 - Có công khai của A (e_A) và B (e_B)
 - A và B đều có khóa công khai e_{PKA} của PKA
- Giao thức 1
 1. A \rightarrow PKA: Alice || Bob
 2. PKA \rightarrow A: $\{Bob||e_B\}d_{KPA}$
 3. A \rightarrow B: $\{r_1\}e_B$
 4. B \rightarrow PKA: Alice || Bob
 5. PKA \rightarrow B: $\{Alice||e_A\}d_{KPA}$
 6. B \rightarrow A: $\{r_1\}e_A$
- Câu hỏi
 - Kiểm tra tính an toàn của giao thức 1
 - Hạn chế của giao thức này?

Giao thức phân phối khóa tập trung

- Giao thức 2: Bên thứ 3 được tin cậy – CA(Certificate Authority)
 - Có cặp khóa (e_{PKA}, d_{PKA})
 - Phát hành chứng thư số cho khóa công khai của các bên có dạng
 - $Cert_A = \{ID_A || e_A || Time_A\}d_{KPA}$
 - ID_A : định danh của thực thể A
 - e_A : khóa công khai của thực thể A đã được đăng ký tại CA
 - $Time_A$: Thời hạn sử dụng khóa công khai. Thông thường có thời điểm bắt đầu có hiệu lực và thời điểm hết hiệu lực

Giao thức phân phối khóa tập trung

- Giao thức 2: Bên thứ 3 được tin cậy – CA(Certificate Authority)
 1. $A \rightarrow CA: ID_A || e_A || Time_A$
 2. $CA \rightarrow A: Cert_A = \{ID_A || e_A || Time_A\}d_{KPA}$
 3. $B \rightarrow CA: ID_B || e_B || Time_B$
 4. $CA \rightarrow B: Cert_B = \{ID_B || e_B || Time_B\}d_{KPA}$
 5. $A \rightarrow B: Cert_A$
 6. $B \rightarrow A: Cert_B$
- Câu hỏi
 - Cải tiến giao thức trên nhằm tăng cường tính an toàn
 - Gợi ý: dùng các phương pháp kiểm tra tính toàn vẹn, ...

Giao thức phân phối khóa tập trung

- Giao thức 2
 1. $A \rightarrow PKA: Alice \parallel Bob \parallel T_1$ (T_1 : timestamp)
 2. $PKA \rightarrow A: \{Bob||e_B\}d_{KPA}$
 3. $A \rightarrow B: \{r_1\}e_B$
 4. $B \rightarrow PKA: Alice \parallel Bob \parallel T_2$ (T_2 : timestamp)
 5. $PKA \rightarrow B: \{Alice||e_A\}d_{KPA}$
 6. $B \rightarrow A: \{r_1\}e_A$
- Ý nghĩa của T_1 và T_2
 - chống tấn công phát lại
- Câu hỏi: giao thức này có hạn chế gì?

Identity Authentication

Van Nguyen – HUT

Hanoi – 2009

(with material from Bishop's text “Introduction
to Computer Security”)

Authentication

- Basics
- Passwords
- Challenge-Response
- Biometrics
- Location
- Multiple Methods

Basics

- Authentication: binding of identity to subject
 - Identity is that of external entity (my identity, Van, etc.)
 - Subject is computer entity (process, etc.)
- Note:
 - message authentication is a different topic and already mentioned in the applications of hash functions

Establishing Identity

- One or more of the following
 - What entity knows (eg. password)
 - What entity has (eg. Identity card, smart card)
 - What entity is (eg. fingerprints, retinal characteristics)
 - Where entity is (eg. In front of a particular terminal)

Authentication System

- We need a formal definition, rather abstract view, of an AS
- A 5-tuple (A, C, F, L, S)
 - A – a set: information that proves identity
 - C – a set: information stored on computer and used to validate authentication information
 - F : a set of complementation functions; $f : A \rightarrow C$
 - To compute complement information from identity information
 - L : authentication functions that prove identity
 - S : functions enabling entity to create, alter information in A or C

Example

- Password system, with passwords stored on line in clear text
 - A set of strings making up passwords
 - $C = A$
 - F singleton set of identity function { / }
 - L single equality test function { eq }
 - S function to set/change password

Passwords

- Sequence of characters
 - Examples: 10 digits, a string of letters, etc.
 - Generated randomly, by user, by computer with user input
- Sequence of words
 - Examples: pass-phrases
- Algorithms
 - Examples: challenge-response, one-time passwords

Storage

- Store as cleartext
 - If password file compromised, *all* passwords revealed
- Encipher file
 - Need to have decipherment, encipherment keys in memory
 - Reduces to previous problem → need something else
- Solution: Instead store one-way hash of password
 - Got the file, attacker must still guess passwords or invert the hash values

Example: Unix

- By definition, a 5-tuple (A , C , F , L , S)
 - A – a set: information that proves identity
 - $A = \{ \text{strings of 8 chars or less} \}$
 - C – a set: information stored on computer and used to validate authentication information
 - $C = \{ \text{hash values of password} \}$
 - F : a set of complementation functions; $f : A \rightarrow C$
 - $F = \{ \text{versions of modified DES} \}$
 - L : authentication functions that prove identity
 - $L = \{ \text{login, su, ...} \}$
 - S : functions enabling entity to create, alter information in A or C
 - $S = \{ \text{passwd, nispasswd, passwd+, ...} \}$

Example: Unix

- By definition, a 5-tuple (A , C , F , L , S)
 - A – a set: information that proves identity
 - $A = \{ \text{strings of 8 chars or less} \}$
 - C – a set: information stored on computer and used to validate authentication information
 - $C = \{ \text{hash values of password} \}$
 - F : a set of complementation functions; $f : A \rightarrow C$
 - $F = \{ \text{versions of modified DES} \}$
 - L : authentication functions that prove identity
 - $L = \{ \text{login, su, ...} \}$
 - S : functions enabling entity to create, alter information in A or C
 - $S = \{ \text{passwd, nispasswd, passwd+, ...} \}$

Attacking passwords

- Goal: find $a \in A$ such that:
 - For some $f \in F$, $f(a) = c \in C$
 - c is associated with entity
- Two ways to determine whether a meets these requirements:
 - By trying computing $f(a)$ for a set of a values until succeed
 - By trying calling $I(a)$ until succeed ($I(a)$ returns true)

Preventing Attacks

■ How to prevent this:

- Hide one of a , f , or c
 - Prevents obvious attack from above
 - Example: UNIX/Linux shadow password files
 - Hides the c 's
- Block access to all $l \in L$ or result of $l(a)$
 - Prevents attacker from knowing if guess succeeded
 - Example: preventing *any* logins to an account from a network
 - Prevents knowing results of l (or accessing l)

Dictionary Attacks

- Trial-and-error from a list of potential passwords
 - *Off-line*: know f and c 's, and repeatedly try different guesses $g \in A$ until the list is done or passwords guessed
 - Examples: *crack*, *john-the-ripper*
 - *On-line*: have access to functions in L and try guesses g until some $l(g)$ succeeds
 - Examples: trying to log in by guessing a password

Success probability over a time period

Anderson's formula:

- P probability of guessing a password in specified period of time
- G number of guesses tested in 1 time unit
- T number of time units
- N number of possible passwords ($|A|$)
- Then $P \geq TG/N$

Example

■ Goal

- Passwords drawn from a 96-char alphabet
- Can test 10^4 guesses per second
- Probability of a success to be 0.5 over a 365 day period
- What is minimum password length?

■ Solution

- $N \geq TG/P = (365 \times 24 \times 60 \times 60) \times 10^4 / 0.5 = 6.31 \times 10^{11}$
- Choose s such that $\sum_{j=0}^s 96^j \geq N$
- So $s \geq 6$, meaning passwords must be at least 6 chars long

On password selection

- Random selection
 - Any password from A equally likely to be selected
- Pronounceable passwords
- User selection of passwords

Pronounceable Passwords

- Generate phonemes randomly
 - Phoneme is unit of sound, eg. cv, vc, cvc, vcv
 - Examples: helgoret, juttelon are; przbqxdfl, zxrptglfn are not
- Problem: too few
- Solution: key crunching
 - Run long key through hash function and convert to printable sequence
 - Use this sequence as password

User Selection

- Problem: people pick easy to guess passwords
 - Based on account names, user names, computer names, place names
 - Dictionary words (also reversed, odd capitalizations, control characters, “elite-speak”, conjugations or declensions, swear words, Torah/Bible/Koran/... words)
 - Too short, digits only, letters only
 - License plates, acronyms, social security numbers
 - Personal characteristics or foibles (pet names, nicknames, job characteristics, etc.)

Picking Good Passwords

- “LIMm*2^Ap”
 - Names of members of 2 families
- “OoHeO/FSK”
 - Second letter of each word of length 4 or more in third line of third verse of Star-Spangled Banner, followed by “/”, followed by author’s initials
- What’s good here may be bad there
 - “DMC/MHmh” bad at Dartmouth (“Dartmouth Medical Center/Mary Hitchcock memorial hospital”), ok here
- Why are these now bad passwords? ☹

Proactive Password Checking

- Analyze proposed password for “goodness”
 - Always invoked
 - Can detect, reject bad passwords for an appropriate definition of “bad”
 - Discriminate on per-user, per-site basis
 - Needs to do pattern matching on words
 - Needs to execute subprograms and use results
 - Spell checker, for example
 - Easy to set up and integrate into password selection system

Salting

- Goal: slow dictionary attacks
- Method: perturb hash function so that:
 - Parameter controls *which* hash function is used
 - Parameter differs for each password
 - So given n password hashes, and therefore n salts, need to hash guess n

Examples

■ Vanilla UNIX method

- Use DES to encipher 0 message with password as key; iterate 25 times
- Perturb E table in DES in one of 4096 ways
 - 12 bit salt flips entries 1–11 with entries 25–36

■ Alternate methods

- Use salt as first part of input to hash function

Unix actually is ...

- UNIX system standard hash function
 - Hashes password into 11 char string using one of 4096 hash functions
- As authentication system:
 - $A = \{ \text{strings of 8 chars or less} \}$
 - $C = \{ 2 \text{ char hash id} \parallel 11 \text{ char hash} \}$
 - $F = \{ 4096 \text{ versions of modified DES} \}$
 - $L = \{ \text{\textit{login}, \textit{su}, ...} \}$
 - $S = \{ \text{\textit{passwd}, \textit{nispasswd}, \textit{passwd+}, ...} \}$

Guessing Through L

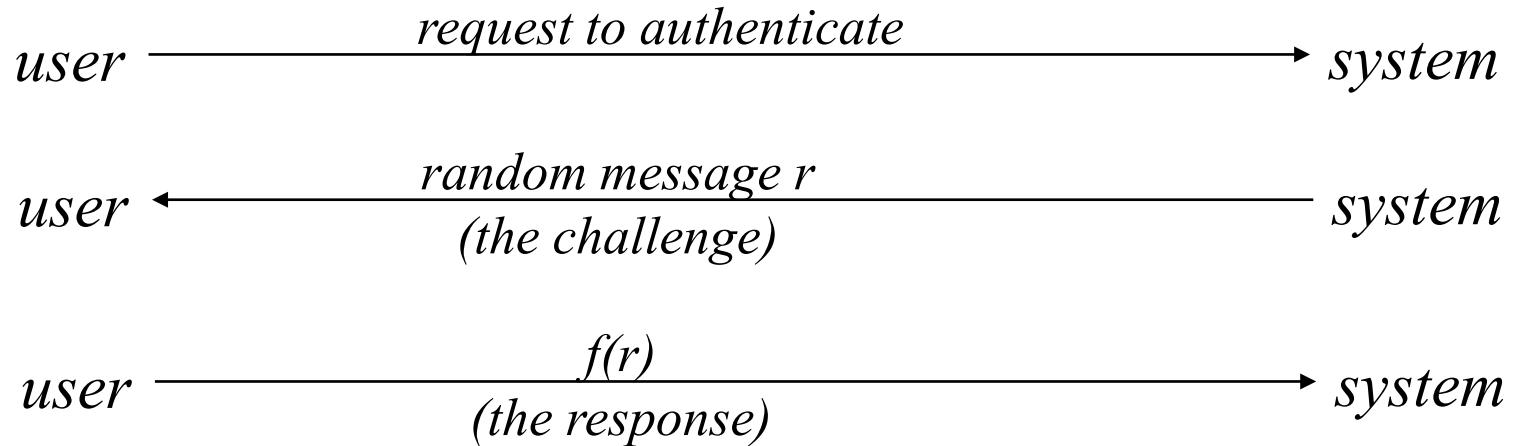
- Cannot prevent these
 - Otherwise, legitimate users cannot log in
- Make them slow
 - Backoff
 - Disconnection
 - Disabling
 - Be very careful with administrative accounts!
 - Jailing
 - Allow in, but restrict activities

Password Aging

- Force users to change passwords after some time has expired
 - How do you force users not to re-use passwords?
 - Record previous passwords
 - Block changes for a period of time
 - Give users time to think of good passwords
 - Don't force them to change before they can log in
 - Warn them of expiration days in advance

Challenge-Response

- User, system share a secret function f (in practice, f is a known function with unknown parameters, such as a cryptographic key)



Pass Algorithms

- Challenge-response with the function f itself a secret
 - Challenge is a random string of characters
 - Response is some function of that string
 - Usually used in conjunction with fixed, reusable password

login

username:

password:

CAPTCHA:

asbury law

Type the two words:

 **reCAPTCHA™**
stop spam.
read books.

One-Time Passwords

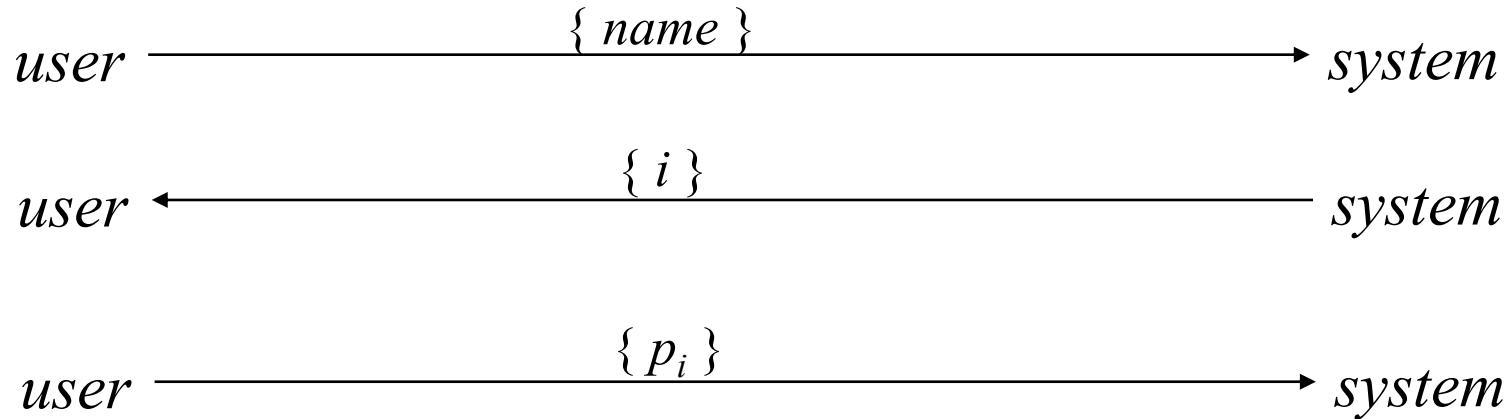
- Password that can be used exactly *once*
 - After use, it is immediately invalidated
- Challenge-response mechanism
 - Challenge is number of authentications; response is password for that particular number
- Problems
 - Synchronization of user, system
 - Generation of good random passwords
 - Password distribution problem

S/Key

- One-time password scheme based on idea of Lamport
- h one-way hash function (MD5 or SHA-1, for example)
- User chooses initial seed k
- System calculates:
$$h(k) = k_1, h(k_1) = k_2, \dots, h(k_{n-1}) = k_n$$
- Passwords are reverse order:
$$p_1 = k_n, p_2 = k_{n-1}, \dots, p_{n-1} = k_2, p_n = k_1$$

S/Key Protocol

System stores maximum number of authentications n , number of next authentication i , last correctly supplied password p_{i-1} .



System computes $h(p_i) = h(k_{n-i+1}) = k_{n-i+2} = p_{i-1}$. If match with what is stored, system replaces p_{i-1} with p_i and increments i .

Hardware Support

- Token-based
 - Used to compute response to challenge
 - May encipher or hash challenge
 - May require PIN from user
- Temporally-based
 - Every minute (or so) different number shown
 - Computer knows what number to expect when
 - User enters number and fixed password

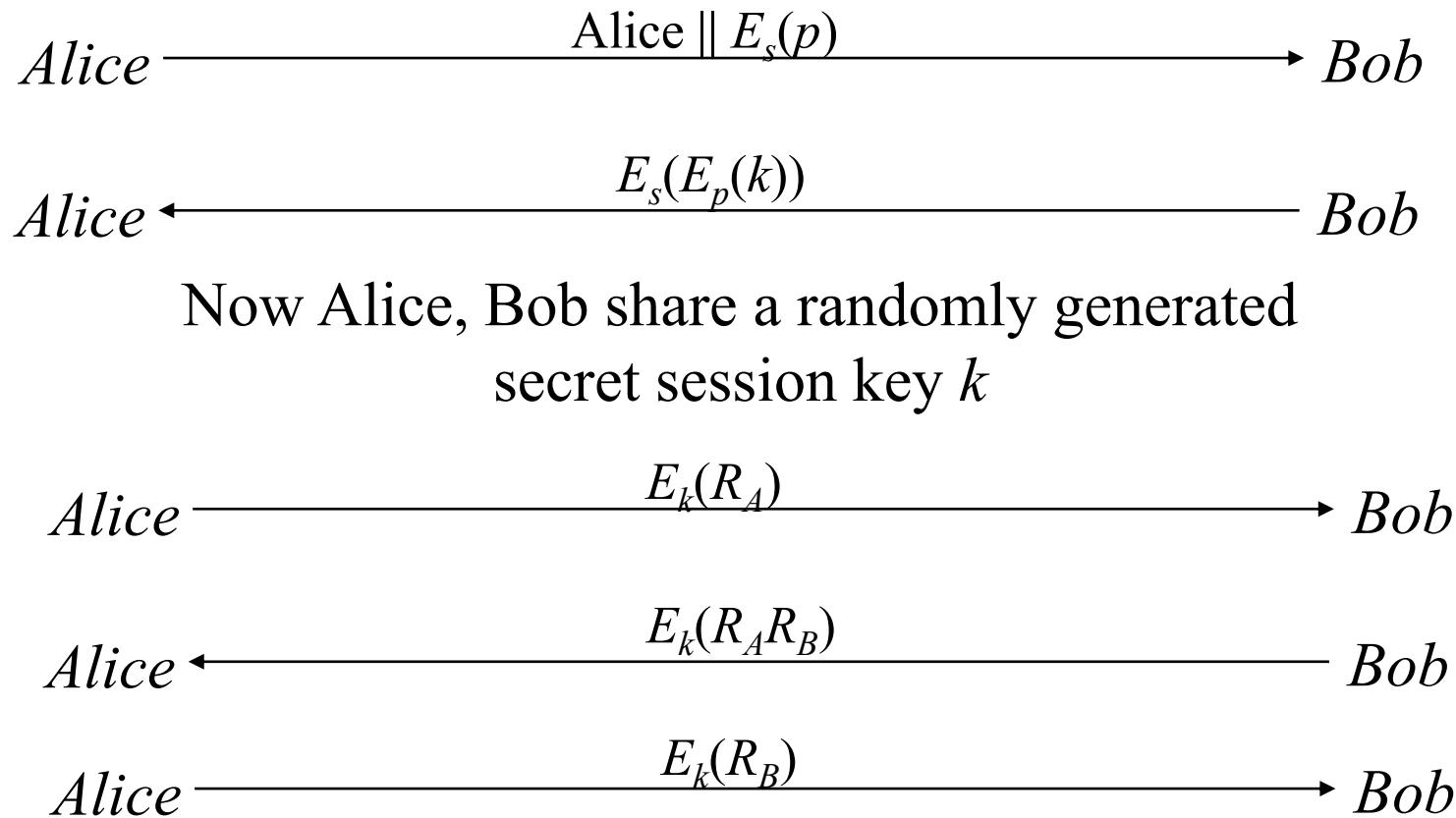
C-R and Dictionary Attacks

- Same as for fixed passwords
 - Attacker knows challenge r and response $f(r)$; if f encryption function, can try different keys
 - May only need to know *form* of response; attacker can tell if guess correct by looking to see if deciphered object is of right form
 - Example: Kerberos Version 4 used DES, but keys had 20 bits of randomness; Purdue attackers guessed keys quickly because deciphered tickets had a fixed set of bits in some locations

Encrypted Key Exchange

- Defeats off-line dictionary attacks
- Idea: random challenges enciphered, so attacker cannot verify correct decipherment of challenge
- Assume Alice, Bob share secret password s
- In what follows, Alice needs to generate a random public key p and a corresponding private key q
- Also, k is a randomly generated session key, and R_A and R_B are random challenges

EKE Protocol



Biometrics

- Automated measurement of biological, behavioral features that identify a person
 - Fingerprints: optical or electrical techniques
 - Maps fingerprint into a graph, then compares with database
 - Measurements imprecise, so approximate matching algorithms used
 - Voices: speaker verification or recognition
 - Verification: uses statistical techniques to test hypothesis that speaker is who is claimed (speaker dependent)
 - Recognition: checks content of answers (speaker independent)

Other Characteristics

- Can use several other characteristics
 - Eyes: patterns in irises unique
 - Measure patterns, determine if differences are random; or correlate images using statistical tests
 - Faces: image, or specific characteristics like distance from nose to chin
 - Lighting, view of face, other noise can hinder this
 - Keystroke dynamics: believed to be unique
 - Keystroke intervals, pressure, duration of stroke, where key is struck
 - Statistical tests used

Cautions

- These can be fooled!
 - Assumes biometric device accurate *in the environment it is being used in!*
 - Transmission of data to validator is tamperproof, correct

Location

- If you know where user is, validate identity by seeing if person is where the user is
 - Requires special-purpose hardware to locate user
 - GPS (global positioning system) device gives location signature of entity
 - Host uses LSS (location signature sensor) to get signature for entity

Multiple Methods

- Example: “where you are” also requires entity to have LSS and GPS, so also “what you have”
- Can assign different methods to different tasks
 - As users perform more and more sensitive tasks, must authenticate in more and more ways (presumably, more stringently) File describes authentication required
 - Also includes controls on access (time of day, etc.), resources, and requests to change passwords
 - Pluggable Authentication Modules

PAM

- Idea: when program needs to authenticate, it checks central repository for methods to use
- Library call: *pam_authenticate*
 - Accesses file with name of program in */etc/pam_d*
- Modules do authentication checking
 - *sufficient*: succeed if module succeeds
 - *required*: fail if module fails, but all required modules executed before reporting failure
 - *requisite*: like *required*, but don't check all modules
 - *optional*: invoke only if all previous modules fail

Example PAM File

```
auth sufficient /usr/lib/pam_ftp.so
auth required      /usr/lib/pam_unix_auth.so use_first_pass
auth required      /usr/lib/pam_listfile.so onerr=succeed \
                    item=user sense=deny file=/etc/ftpusers
```

For ftp:

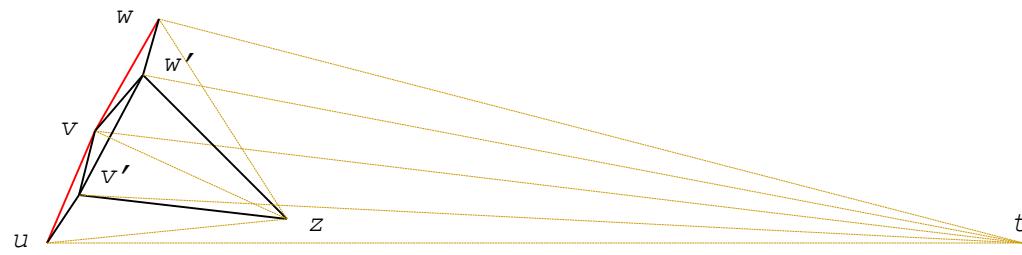
1. If user “anonymous”, return okay; if not, set PAM_AUTHOK to password, PAM_RUSER to name, and fail
2. Now check that password in PAM_AUTHOK belongs to that of user in PAM_RUSER; if not, fail
3. Now see if user in PAM_RUSER named in /etc/ftpusers; if so, fail; if error or not found, succeed

Key Points

- Authentication is not cryptography
 - You have to consider system components
- Passwords are here to stay
 - They provide a basis for most forms of authentication
- Protocols are important
 - They can make masquerading harder
- Authentication methods can be combined
 - Example: PAM

Kerberos

- A computer network authentication protocol
 - which allows nodes communicating over a non-secure network to prove their identity to one another in a secure manner.
- Details:
 - Self-study materials from Internet



Information Security

Van K Nguyen - HUT

Access Control

Topics

- Overview
- Access Control Matrix model
- Discretionary Access Control (DAC)
- Mandatory Access Control (MAC) and an example model
- Role Based Access Control (RBAC)
- Access Control in Unix

What is AC

- Quote from Ross Anderson (text “Security Engineering”)
 - Its function is to control which principals (persons, processes, machines, ...) have access to which resources in the system -- which files they can read, which programs they can execute, and how they share data with other principals, and so on.

Access Control is Pervasive

- Application
 - business applications
- Middleware
 - DBMS
- Operating System
 - controlling access to files, ports
- Hardware
 - memory protection, privilege levels

Access Control Matrix – A general model for protection systems

- Lampson'1971
 - “Protection”
- Refined by Graham and Denning'1972
 - “Protection---Principles and Practice”
- Harrison, Ruzzo, and Ullman'1976
 - “Protection in Operating Systems”

Overview

- Protection state of system
 - Describes current settings, values of system relevant to protection
- Access control matrix
 - Describes protection state precisely
 - Matrix describing rights of subjects
 - State transitions change elements of matrix

Access Matrix

- A set of subjects S
- A set of objects O
- A set of rights R
- An access control matrix
 - one row for each subject
 - one column for each subject/object
 - elements are right of subject on another subject or object

Description

		objects (entities)					
		o_1	\dots	o_m	s_1	\dots	s_n
subjects	s_1						
	s_2						
	\dots						
	s_n						

- Subjects $S = \{ s_1, \dots, s_n \}$
- Objects $O = \{ o_1, \dots, o_m \}$
- Rights $R = \{ r_1, \dots, r_k \}$
- Entries $A[s_i, o_j] \subseteq R$
- $A[s_i, o_j] = \{ r_x, \dots, r_y \}$
means subject s_i has rights r_x, \dots, r_y over object o_j

Example 1

- Processes p, q
- Files f, g
- Rights r, w, x, a, o

	f	g	p	q
p	rwo	r	$rwxo$	w
q	a	ro	r	$rwxo$

Example 2

- Procedures *inc_ctr*, *dec_ctr*, *manage*
- Variable *counter*
- Rights +, -, call

	<i>counter</i>	<i>inc_ctr</i>	<i>dec_ctr</i>	<i>manage</i>
<i>inc_ctr</i>	+			
<i>dec_ctr</i>	-			
<i>manage</i>		<i>call</i>	<i>call</i>	<i>call</i>

Implementation

- Storing the access matrix
 - by rows: capability lists
 - by column: access control lists
 - through indirection:
 - e.g., key and lock list
 - e.g., groups, roles, multiple level of indirections, multiple locks
- How to do indirection correctly and conveniently is the key to management of access control.

Implementation

**Access Control List (column)
(ACL)**

Capability List (row)

Joe: File 1/Read, File 1/Write, File 1/Own, File 2/Read

Sam: File 2/Read, File 2/Write, File 2/Own

Access Control Triples	<u>Subject</u>	<u>Access</u>	<u>Object</u>
	Joe	Read	File 1
	Joe	Write	File 1
	Joe	Own	File 1
	Joe	Read	File 2
	Sam	Read	File 2
	Sam	Write	File 2
	Sam	Own	File 2

Access control lists

U: r,w, own
V: w
S: r

Object F

S: r,w, own
T: r,w
U: r

Object G

- ACL is a list of permissions attached to an object
 - Who can modify the object's ACL?
 - What changes are allowed?
 - How are contradictory permissions handled?
 - How is revocation handled?

Owners and Groups

- Who can modify the object's ACL?
 - One way is by introducing owners of objects
- With ACLs we can define any combination of access, but that makes them difficult to manage
 - Group allow relatively fine-grained access control while making ACLs easier to manage
- Owners and groups can change

Capability lists

- One way to partition the matrix is by rows.
 - All access rights of one user together, stored in a data structure called a **capability list**
 - Lists all the access rights or capabilities that a user has.
 - E.g. Fred --> /dev/console(RW)--> fred/prog.c(RW)--> fred/letter(RW) --> /usr/ucb/vi(X) Jane --> /dev/console(RW)--> fred/prog.c(R)--> fred/letter() --> /usr/ucb/vi(X)

Capability lists

- All access to objects is done through capabilities
 - Every program holds a set of capabilities
 - Each program holds a small number of capabilities
 - The only way a program can obtain capabilities is to have them granted as a result of some communication
 - The set of capabilities held by each program must be as small as possible (*principle of least privilege*)
- Example: EROS Operating System
 - <http://www.eros-os.org/eros.html>

Harrison-Ruzzo-Ullman model

- **Discretionary Access Control**
 - Rights defined on specific (subject, object), decided by individual owners (as oppose to **Mandatory Access Control**, decided by system policies)
- HRU work
 - Formulating access matrices, towards Operating Systems
 - Provide a model that is sufficiently powerful to encode several access control approaches, and precise enough so that security properties can be analyzed
 - Introduce the “safety problem”
 - Show that the safety problem
 - is decidable in certain cases
 - is undecidable in general
 - is undecidable in monotonic case

Primitive Operations

- **create subject s; create object o**
 - Creates new row, column in ACM; creates new column in ACM
- **destroy subject s; destroy object o**
 - Deletes row, column from ACM; deletes column from ACM
- **enter r into A[s, o]**
 - Adds r rights for subject s over object o
- **delete r from A[s, o]**
 - Removes r rights from subject s over object o

Creating File

- Process p creates file f with r and w permission

```
command create•file( $p$ ,  $f$ )
    create object  $f$ ;
    enter own into  $A[p, f]$ ;
    enter  $r$  into  $A[p, f]$ ;
    enter  $w$  into  $A[p, f]$ ;
end
```

Mono-Operational Commands

- Make process p the owner of file g

```
command make-owner( $p$ ,  $g$ )
    enter own into  $A[p, g]$ ;
end
```

- Mono-operational command
 - Single primitive operation in this command

Conditional Commands

- Let p give q r rights over f , if p owns f

```
command grant•read•file•1(p, f, q)
    if own in A[p, f]
    then
        enter r into A[q, f];
    end
```

- Mono-conditional command
 - Single condition in this command

Discretionary Access Control (DAC)

- No precise definition
- Widely used in modern operating systems
- Often has the notion of owner of an object
- The owner controls other users' accesses to the object
- Allows access rights to be propagated to other subjects

Drawbacks in DAC

- DAC cannot protect against
 - Trojan horse
 - Malware
 - Software bugs
 - Malicious local users
- Cannot control information flow

Mandatory Access Control (MAC)

Mandatory Access Control

- # *Objects*: security classification
 - e.g., grades=(confidential, {student-info})
- # *Subjects*: security clearances
 - e.g., Joe=(confidential, {student-info})
- # *Access rules*: defined by comparing the security classification of the requested objects with the security clearance of the subject
 - e.g., subject can read object only if label(subject) dominates label(object)

Mandatory Access Control

- # If *access control rules* are satisfied, access is permitted

e.g., Joe wants to read grades.

$\text{label}(\text{Joe}) = (\text{confidential}, \{\text{student-info}\})$

$\text{label}(\text{grades}) = (\text{confidential}, \{\text{student-info}\})$

Joe is permitted to read grades

- # *Granularity* of access rights!

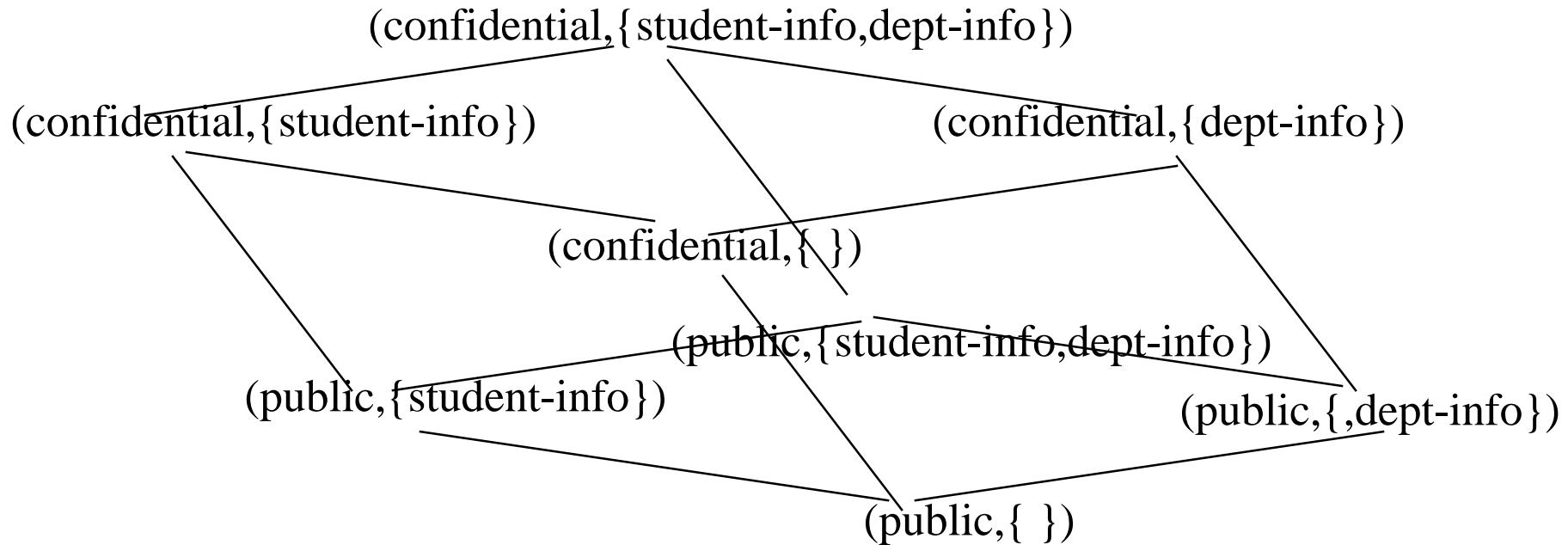
Mandatory Access Control

Security Classes (labels): (A,C)

A – total order authority level

C – set of categories

e.g., A = confidential > public , C = { student-info, dept-info }



Mandatory Access Control

- # *Dominance* (\geq): label $l=(A,C)$ dominates $l'=(A',C')$ iff $A \geq A'$ and $C \supseteq C'$

e.g., (confidential, {student-info}) \geq (public, {student-info})

BUT NOT

(confidential, {student-info}) \geq (public, {student-info, department-info})

Bell- LaPadula (BLP) Model

- Confidentiality protection
- Lattice-based access control
 - Subjects
 - Objects
 - Security labels
- Supports decentralized administration

BLP Reference Monitor

- All accesses are controlled by the reference monitor
- Cannot be bypassed
- Access is allowed iff the resulting system state satisfies all security properties
- *Trusted subjects*: subjects trusted not to compromise security

BLP Axioms 1.

Simple-security property: a subject s is allowed to read an object o only if the security label of s dominates the security label of o

- ❑ No read up
- ❑ Applies to *all subjects*

Subject s can read object o iff $L(o) \leq L(s)$ and s has permission to read o

- Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)

BLP Axioms 2.

**-property*: a subject s is allowed to write an object o only if the security label of o dominates the security label of s

- No write down
- Applies to *un-trusted subjects* only
 - Subject s can write object o iff $L(s) \leq L(o)$ and s has permission to write o
 - Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)

Example

<i>security level</i>	<i>subject</i>	<i>object</i>
Top Secret	Tamara	Personnel Files
Secret	Samuel	E-Mail Files
Confidential	Claire	Activity Logs
Unclassified	Ulaley	Telephone Lists

- Tamara can read all files
- Claire cannot read Personnel or E-Mail Files
- Ulaley can only read Telephone Lists

Levels and Lattices

- Security level is (*clearance, category set*)
 - (Top Secret, { NUC, EUR, ASI })
 - (Confidential, { EUR, ASI })
 - (Secret, { NUC, ASI })
- $(A, C) \text{ dom } (A', C')$ iff $A' \leq A$ and $C' \subseteq C$
 - (Top Secret, {NUC, ASI}) *dom* (Secret, {NUC})
 - (Secret, {NUC, EUR}) *dom* (Confidential,{NUC, EUR})
 - (Top Secret, {NUC}) $\neg\text{dom}$ (Confidential, {EUR})

MAC Overview

- Advantages:
 - Very secure
 - Centralized enforcement
- Disadvantages:
 - May be too restrictive
 - Need additional mechanisms to implement multi-level security system
 - Security administration is difficult

Role-Based Access Control (RBAC)

RBAC Motivation

- Multi-user systems
- Multi-application systems
- Permissions are associated with roles
- Role-permission assignments are persistent v.s. user-permission assignments
- Intuitive: competency, authority and responsibility

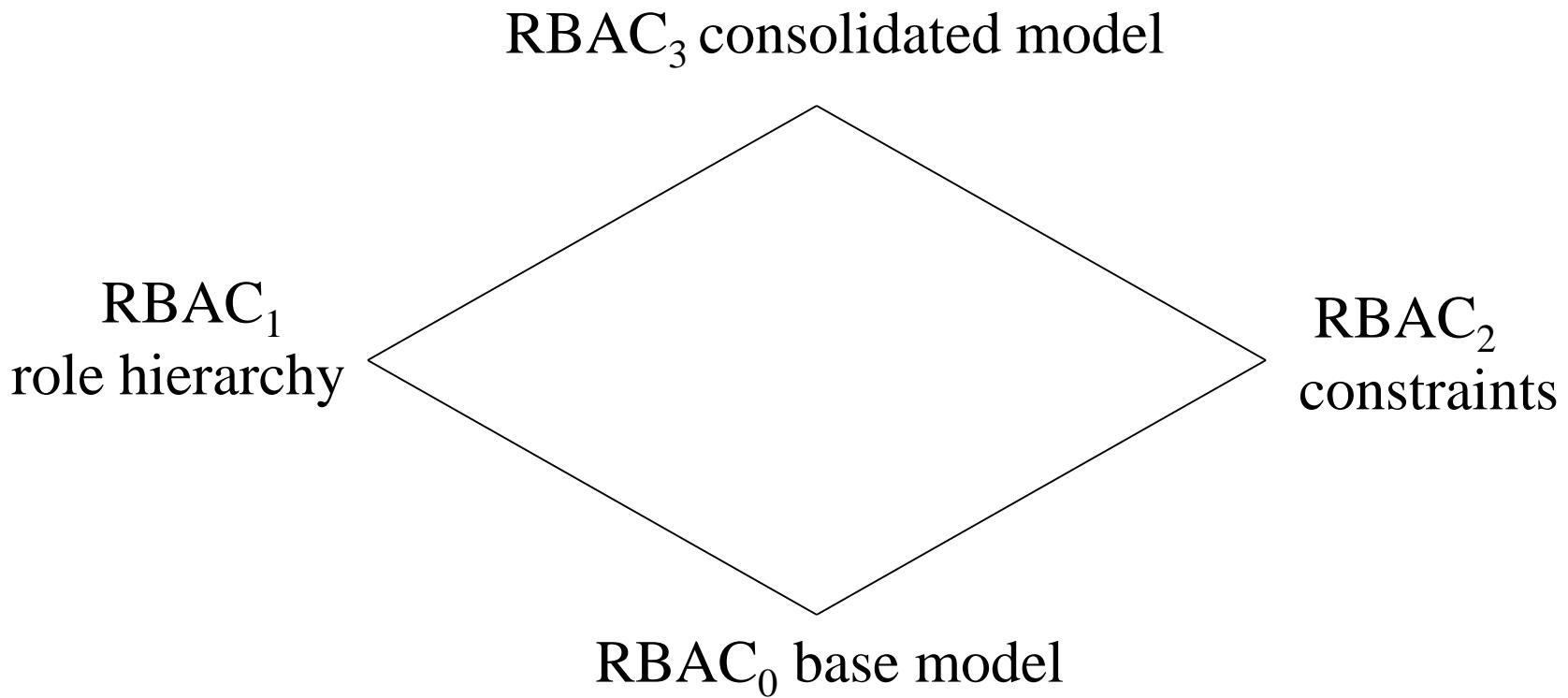
Motivation

- Express organizational policies
 - Separation of duties
 - Delegation of authority
- Flexible: easy to modify to meet new security requirements
- Supports
 - Least-privilege
 - Separation of duties
 - Data abstraction

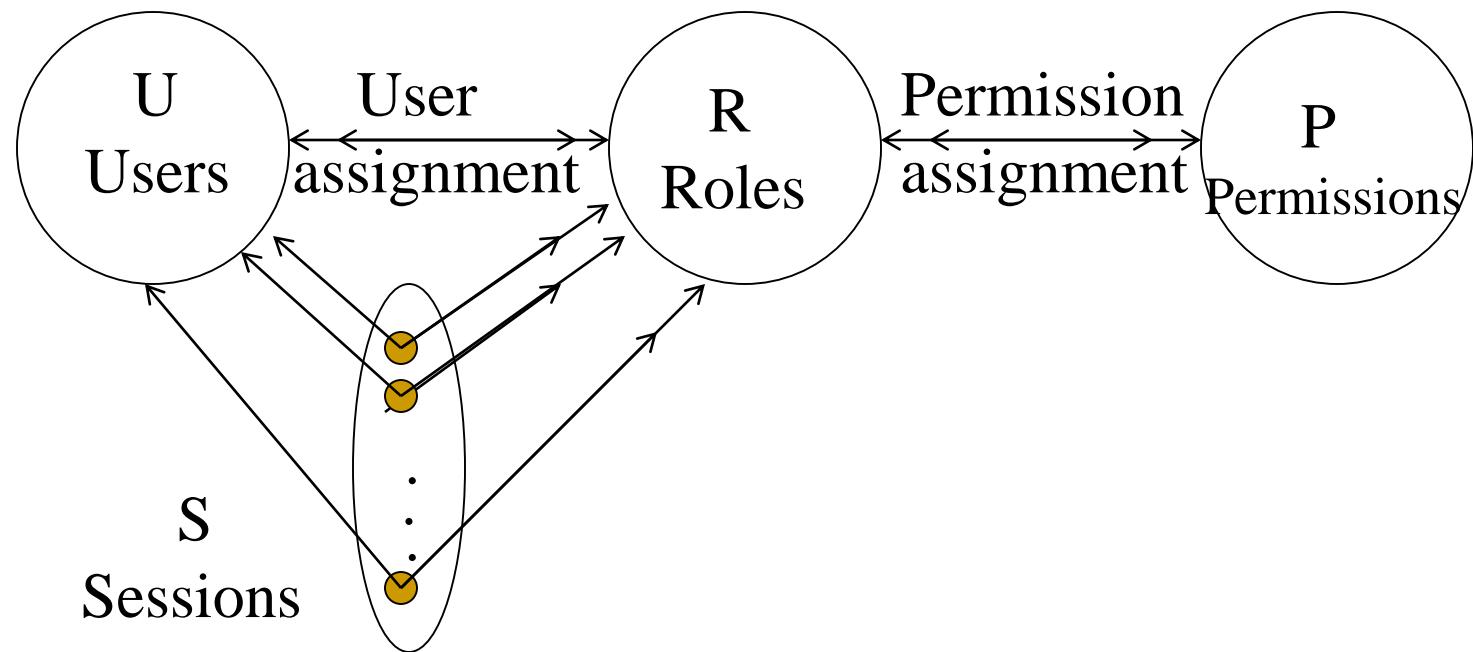
Roles

- **User group:** collection of user with possibly different permissions
- **Role:** mediator between collection of users and collection of permissions
- **RBAC** independent from DAC and MAC (they may coexist)
- RBAC is **policy neutral:** configuration of RBAC determines the policy to be enforced

RBAC



RBAC₀



RBAC₀

- User: human beings
- Role: job function (title)
- Permission: approval of a mode of access
 - Always positive
 - Abstract representation
 - Can apply to single object or to many

RBAC₀

- UA: user assignments
 - Many-to-many
- PA: Permission assignment
 - Many-to-many
- Session: mapping of a user to possibly many roles
 - Multiple roles can be activated simultaneously
 - Permissions: union of permissions from all roles
 - Each session is associated with a single user
 - User may have multiple sessions at the same time

RBAC₀ Components

- **Users, Roles, Permissions, Sessions**
- $PA \subseteq P \times R$ (many-to-many)
- $UA \subseteq U \times R$ (many-to-many)
- user: $S \rightarrow U$, mapping each session s_i to a single user $\text{user}(s_i)$
- roles: $S \rightarrow 2^R$, mapping each session s_i to a set of roles:
 - $\text{roles}(s_i) \subseteq \{r \mid (\text{user}(s_i), r) \in UA\}$ and s_i has permissions
 $\cup_{r \in \text{roles}(s_i)} \{p \mid (p, r) \in PA\}$

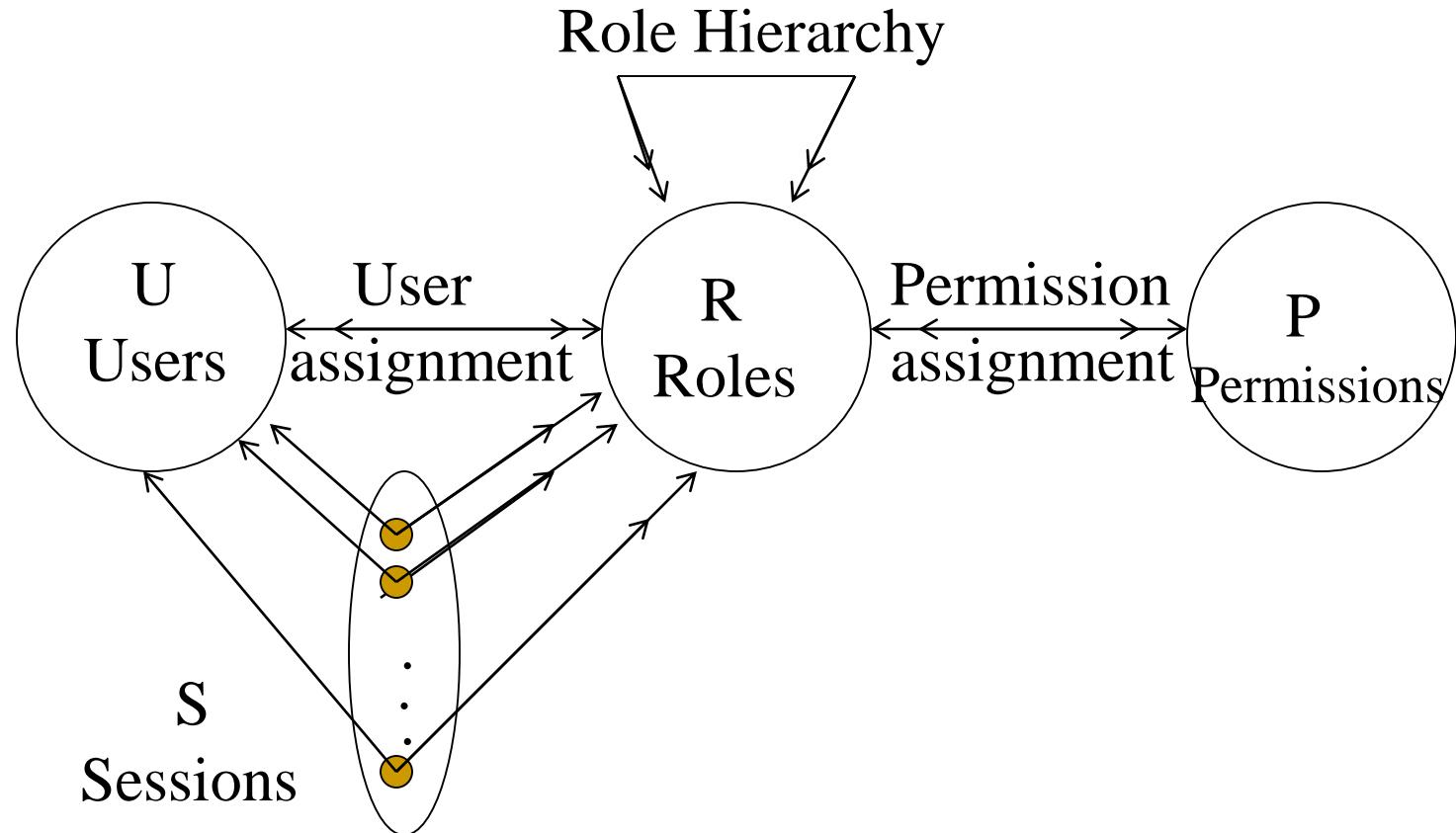
RBAC₀

- Permissions apply to data and resource objects only
- Permissions do NOT apply to RBAC components
- Administrative permissions: modify U,R,S,P
- Session: under the control of user to
 - Activate any subset of permitted roles
 - Change roles within a session

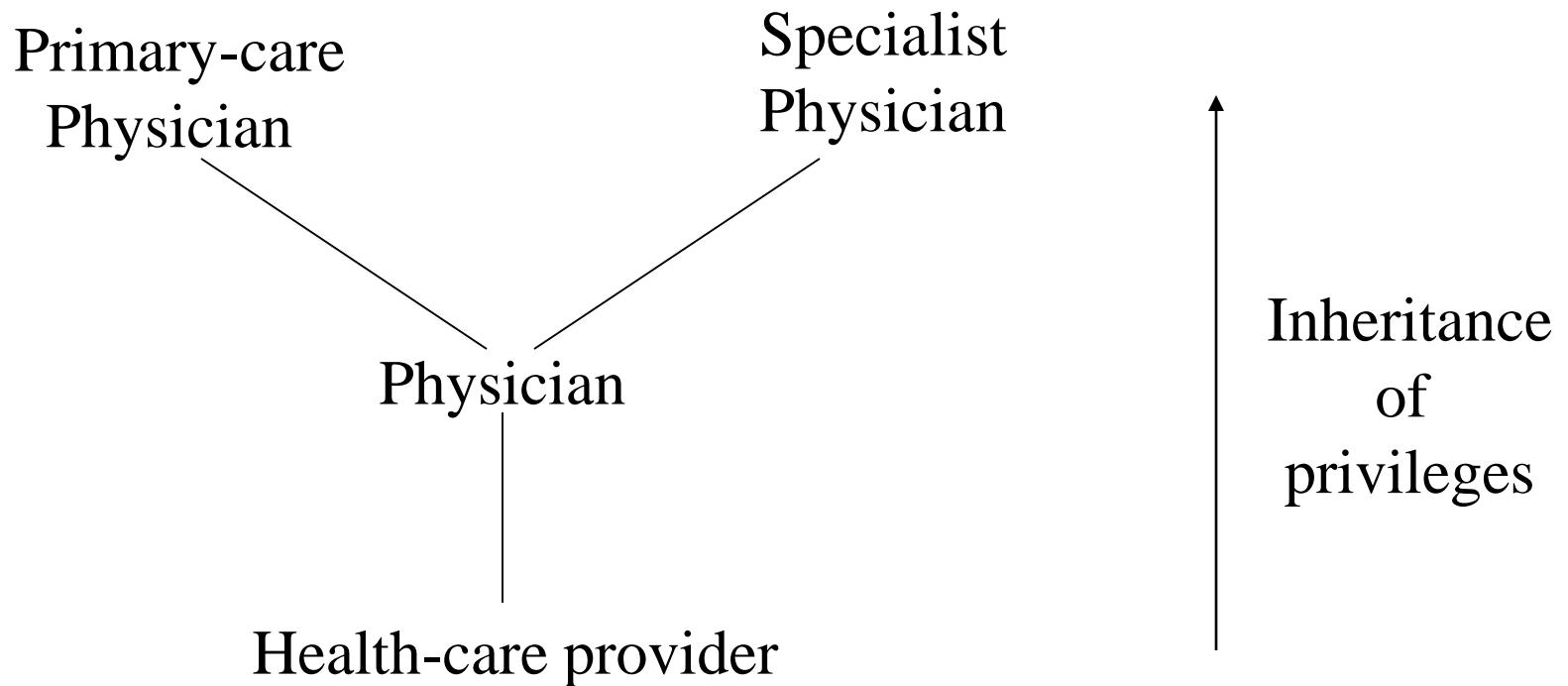
RBAC₁

- Structuring roles
- Inheritance of permission from junior role (bottom) to senior role (top)
- Partial order
 - Reflexive
 - Transitive
 - Anti-symmetric

RBAC₁



Role Hierarchy



RBAC₁ Components

- # Same as RBAC₀: **Users, Roles, Permissions, Sessions**, $PA \subseteq P \times R$, $UA \subseteq U \times R$, $\text{user}: S \rightarrow U$, mapping each session s_i to a single user $\text{user}(s_i)$
- # $RH \subseteq R \times R$, partial order (\geq dominance)
- # roles: $S \rightarrow 2^R$, mapping each session s_i to a set of roles
 - # $\text{roles}(s_i) \subseteq \{r \mid (\exists r' \geq r) [(user(s_i), r') \in UA]\}$ and s_i has permissions $\cup_{r \in \text{roles}(s_i)} \{p \mid (\exists r'' \leq r) [(p, r'') \in PA]\}$

Access Control in Unix

General Concepts

■ Users, Groups, Processes, Files

- Each user has a unique UID
- Each group has a unique GID
- Each process has a unique PID
- Users belong to multiple groups GID
- Objects whose access is controlled
 - Files
 - Directories

■ Organization of Objects

- Files are arranged in a hierarchy
- Files exist in directories
- Directories are one type of files
- In UNIX, access on directories are not inherited

Basic Permissions Bits on Files

- Permission:
 - Read: control reading the content of a file
 - Write: controls changing the content of a file
 - Execute: controls loading the file then execute
- Many operations can be performed only by the owner of the file
- Where are Permission Bits Kept?
 - Each file/directory has associated an i-node.
 - The file type, permissions, owner UID and owner GID are save on disk in the inode of a file or directory

Permission Bits on Directories

- Read: for showing file names in a directory
- Execution: for traversing a directory
 - does a lookup, allows one to find inode # from file name
 - ‘chdir’ to a directory requires execution
- Write + execution: for creating/deleting files in the directory
 - requires no permission on the file
- Accessing a file a path name: need execution permission to all directories along the path

The Three Sets of Permission Bits

- Permission example

drwxr-xr-x

- First: directory or not
 - Next three: owner permission
 - if the user is the owner of a file then the r/w/x bits for owner apply
 - Next three: group permission
 - if the user belongs to the group the file belongs to then the r/w/x bits for group apply
 - Next three: others permission
 - Apply when not the owner or belong to the group

- Where are Permission Bits Kept?

- Each file/directory has associated an inode.
 - The file type, permissions, owner UID and owner GID are save on disk in the inode of a file or directory

Users vs. Subjects

- Permission bits talk about what users can access a file
 - ➔ but it is subjects (processes) to perform actions on files
 - When a subject accesses a file, the system check which user it is acting on behalf of
- Problem: what if an executable need stronger permission than the subject calling it
 - The **passwd** program needs to update a system-wide password file, which ordinary users should not be able to modify, but only root can modify
 - But remember, it needs to be run by ordinary users

Real User ID vs. Effective User ID

- Each process has three user IDs
 - real user ID (ruid): owner of the process
 - effective user ID (euid): used in most access control decisions, often the same as ruid unless there is a change
 - saved user ID (suid): keeps the previous euid if it was a change
- and three group IDs
 - real group ID
 - effective group ID
 - saved group ID

The setuid flag

■ When used for a file

- allows certain processes to have more than ordinary privileges while still being executable by ordinary users
- When set, the effective uid of the calling process takes the value of the owner of the file

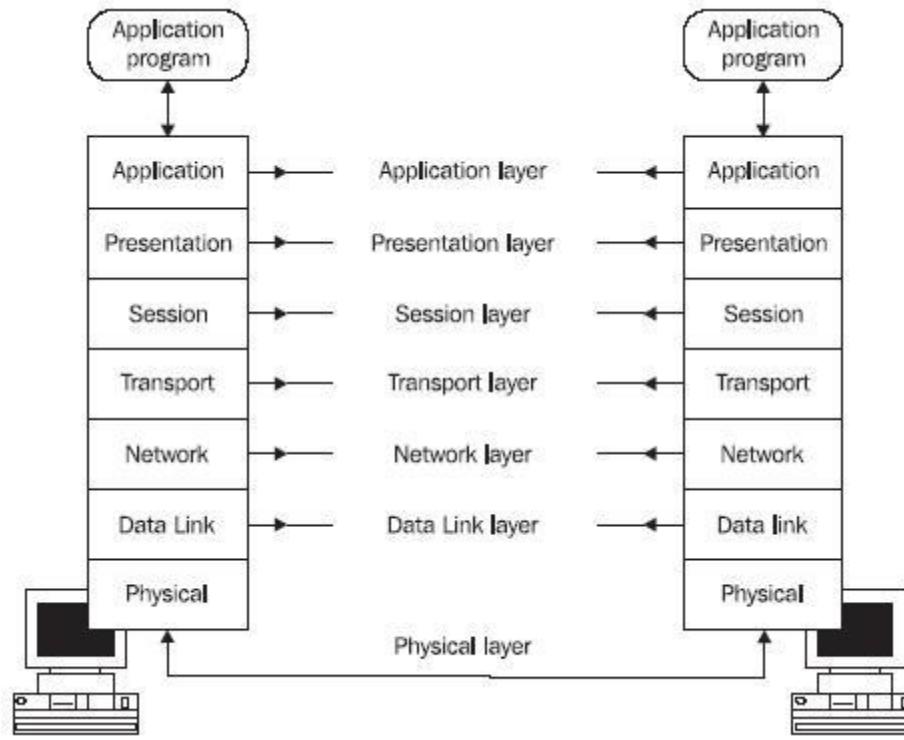
How the process user IDs work

- When a process is created by *fork*
 - it inherits all three UIDs from its parent process
- When a process executes a file by *exec*
 - if (the setuid bit of the file is off)
 - it keeps its three user IDs
 - otherwise // the setuid is set
 - euid of the process = ruid of the file
 - suid = previous euid
- How to solve the passwd problem and the likes?
 - Passwd is owned by root and setuid is set
 - When a process executes it, then effective user becomes root, so the program runs as root on behalf of the user (only within the passwd work)
- Can be a security flaw if the mechanism for temporary higher privilege is abused

Information Security

Van K Nguyen - HUT

Network Security

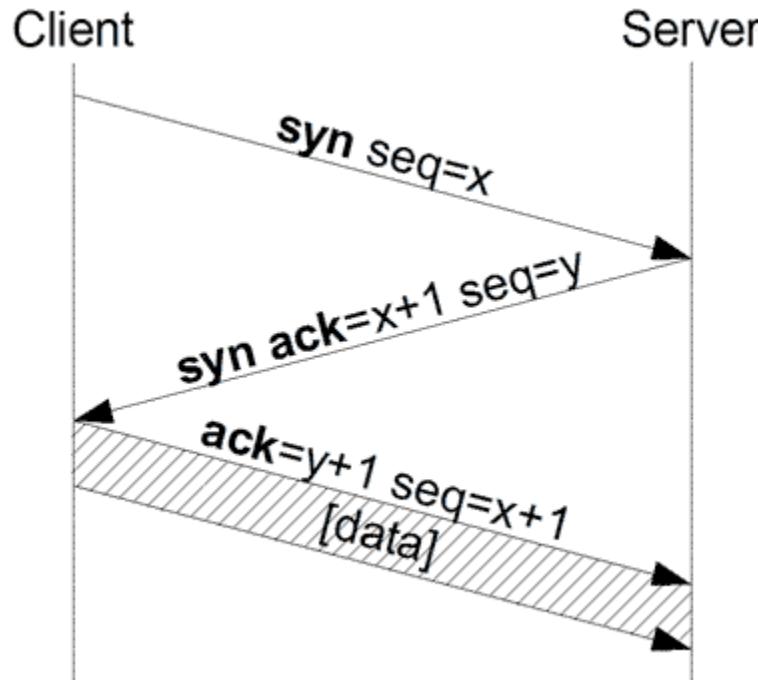


Attacks against TCP

Transmission Control Protocol - TCP

- Connection oriented protocol for a user process
 - **Reliable**, full-duplex channel: acknowledgements, retransmissions, timeouts
 - The packets are delivered in the same order
- Congestion control mechanisms

TCP 3-way handshake



- The sequence number x and y are random values that the other side need to **ack** by increment ($x+1$ or $y+1$)
- The connection only fully opened when server-side received client's ack

SYN Attack

- An attacker sends flood of SYNs with source address spoofed packets to a target.
- If the limit is reached, target machine will refuse any incoming connections till the timeout expires
 - The server send the SYN-ACK to the falsified IP address, and thus never receive the ACK
 - Server wait for ACK for some time, as simple network congestion could also be the cause of the missing ACK.
- Spoofed address chosen to be a non-existent one
 - If the spoofed address belongs to a machine, then what ?

Why it works?

- There is no authentication of the source of the packets
- Addresses can be easily spoofed
- Server needs to allocate a lot of resources while client doesn't

Some measurements to the SYN attack

■ Configuration Optimization

□ At the server

- Reduce the timeout to 10 seconds
- Increase the size of the queue
- Disable non-essential services, reducing the number of ports to be attacked

□ At all routers in the Internet

- Block packets to the outside that have source addresses from outside the internal network

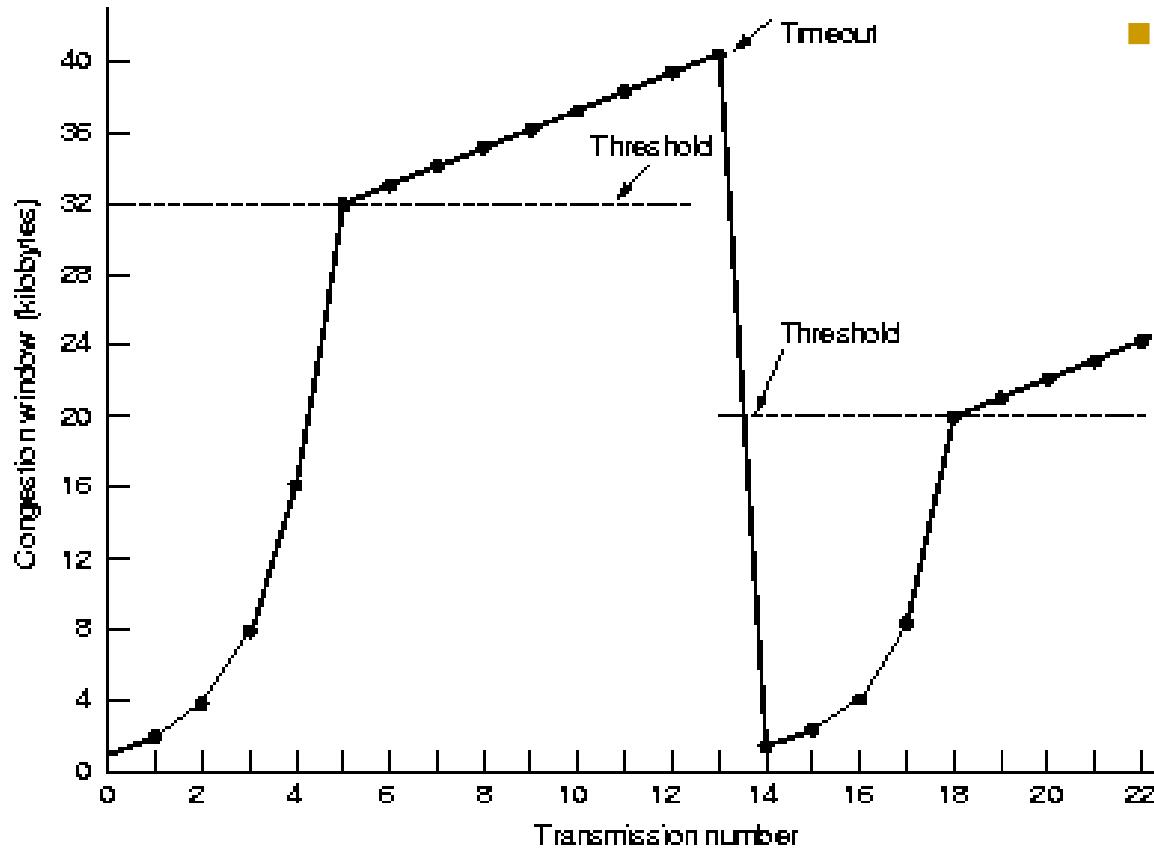
Some measurements to the SYN attack

- Using firewall as relay/gateway
 - Firewall acts in between, receive then forward the SYN packet to server
 - Firewall send “fake” ACK to server, then wait a little timeout then send RST to server if no real ACK coming.
- Active Monitoring
 - Monitor the TCP traffic within a local area network and figure out which ones are illegitimate connections.
 - Send RST for the illegitimate connections to close them

TCP Congestion Control

- Source determines how much bandwidth is available for it to send, it starts slow and increases the window of send packet based on ACKS.
- ACKS are also used to control the transmission of packets.
- Uses Additive Increase Multiplicative Decrease (AIMD)
- Uses Retransmission Timeout (RTO) to avoid congestion

TCP Congestion Control



- All the attacker needs to do is generate a TCP flow to force the targeted TCP connection to repeatedly enter a retransmission timeout state

IPsec: secure communication for the IP layer

Intro

- **Internet Protocol Security (IPsec)** is a protocol suite for securing Internet Protocol (IP) communications by authenticating and encrypting each IP packet of a data stream.
 - Authentication/integrity
 - Confidentiality
 - Protection against replayed packets
- Transparent to applications
 - below transport layer (TCP, UDP)
- IETF IPSEC Working Group
 - Documented in RFCs and Internet drafts

Basics on IPSec

- Protocols
 - Internet key exchange (IKE): set up a **security association** (SA) with encryption and authentication keys to be used.
 - Authentication Header (AH): provides integrity and authentication without confidentiality
 - Encapsulating Security Payload (ESP): provides confidentiality and can also provide integrity and authentication
- Both AH/ESP can operate on two different modes
 - Transport-mode: encapsulates an upper-layer protocol (e.g. TCP or UDP) and prepends an IP header in clear
 - Tunnel-mode: encapsulates an entire IP datagram into new packet adding a new IP header

Transport mode

- ESP in Transport Mode
 - encrypts and optionally authenticates the IP payload (data), but not the IP header.
- AH in Transport Mode
 - authenticates the IP payload and selected portions of the IP header

Tunnel Mode

■ ESP in Tunnel Mode

- encrypts and optionally authenticates the entire inner IP packet, including the inner IP header.

■ AH in Tunnel Mode

- authenticates the entire inner IP packet and selected portions of the outer IP header.

Security Associations

- SA- the basis for building security functions into IP.
- A security association is simply the bundle of algorithm selection and parameters (such as keys) that is being used to encrypt and authenticate a particular flow in one direction.
 - SPI (Security Parameter Index) + IP destination address uniquely identifies a particular Security Association.
- Therefore, in normal bi-directional traffic, the flows are secured by a pair of security associations.
 - SAs are unidirectional, sender supplies SPI to receiver.

Authentication Header

- Provides support for data integrity and authentication (MAC) of IP packets, using HMAC based on MD5 or SHA1.
- Defends against replay attacks (sequence number)

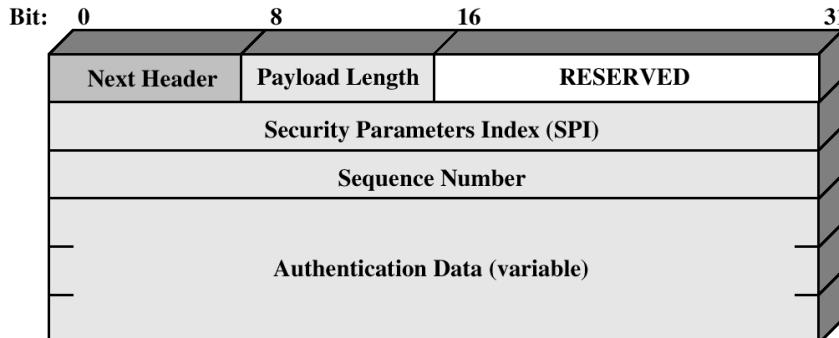
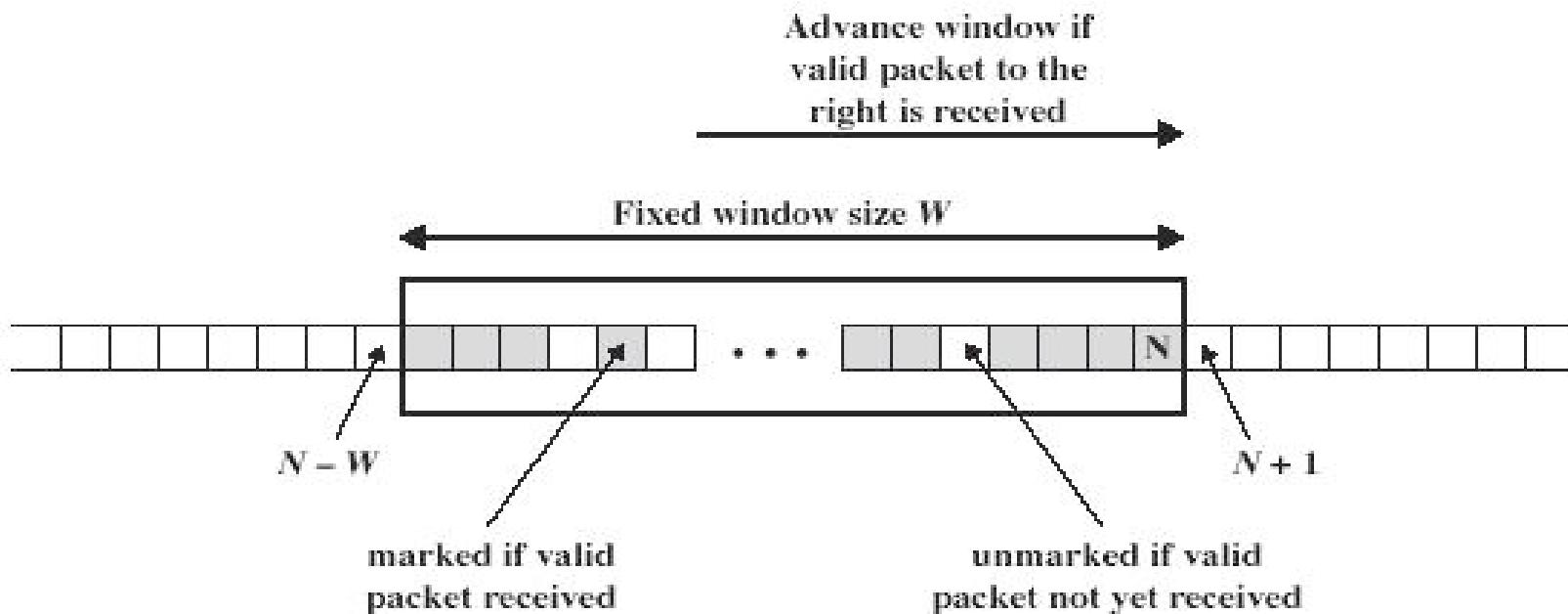


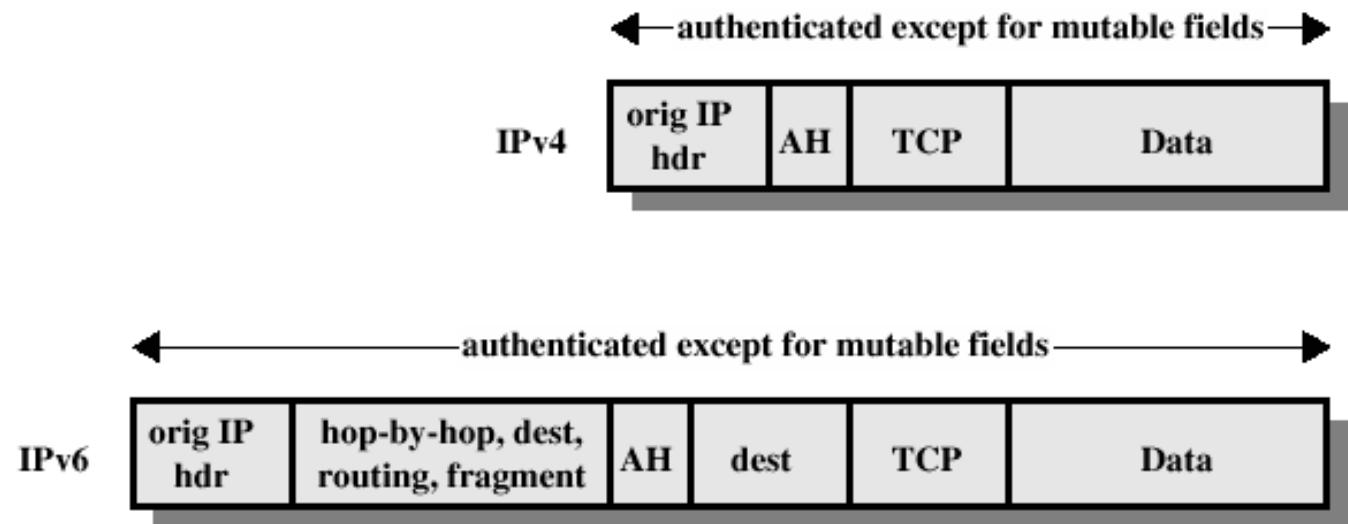
Figure 6.3 IPSec Authentication Header

AH: Preventing Replay

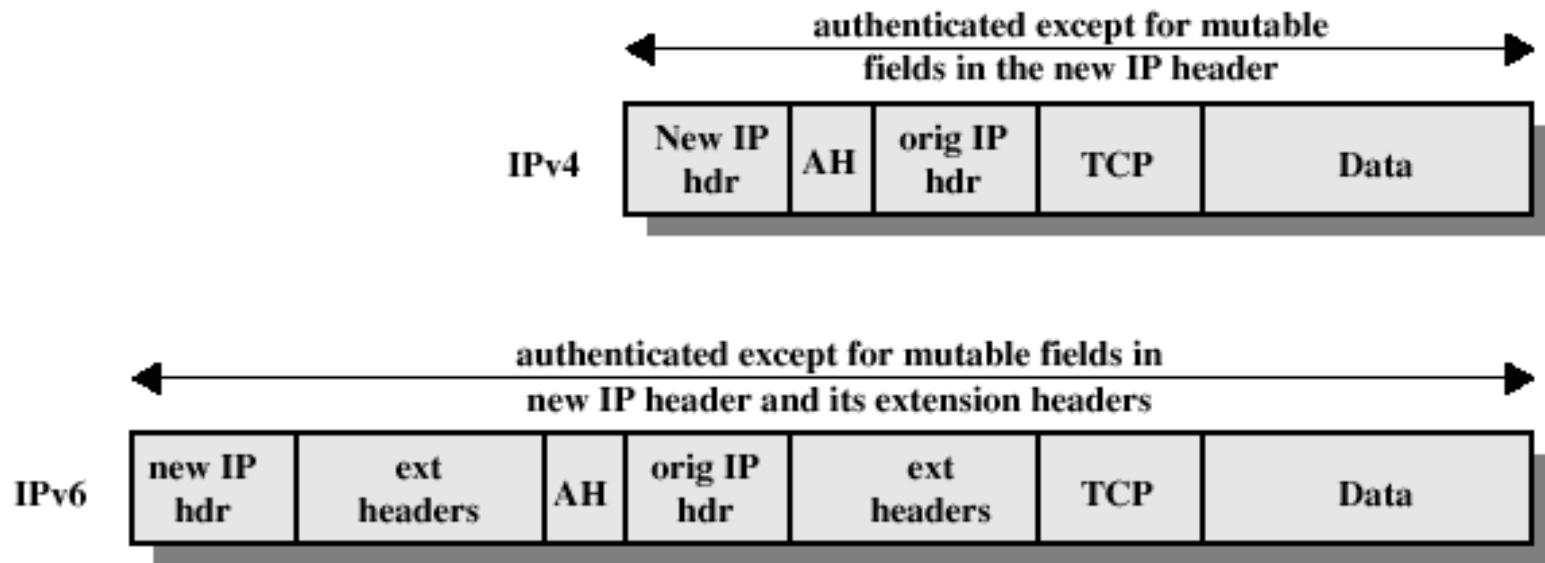
- When a SA is established, sender initializes sequence counter to 0.
- Every time a packet is sent the counter is incremented and is set in the sequence number in the AH header.
- When sequence number $2^{32} - 1$ is reached, a new SA should be negotiated.



AH Authentication: Transport Mode



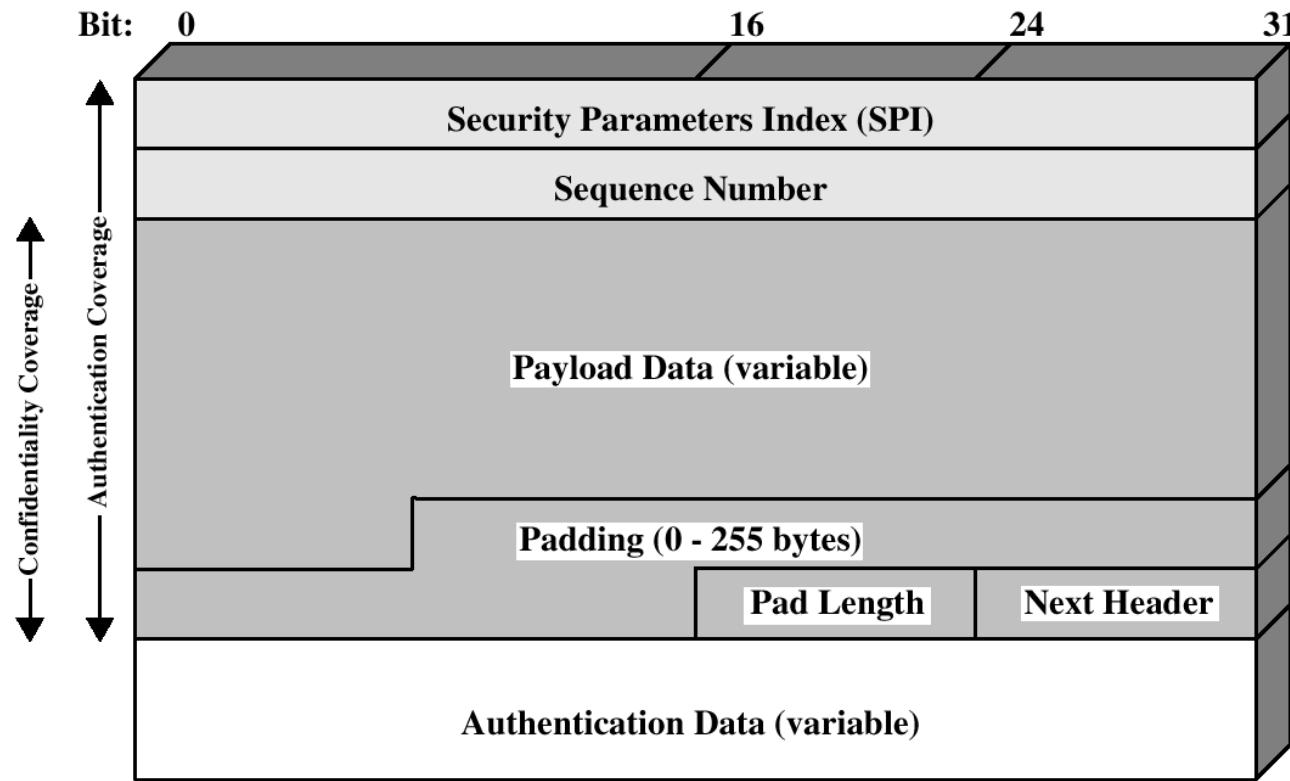
AH Authentication: Tunnel Mode



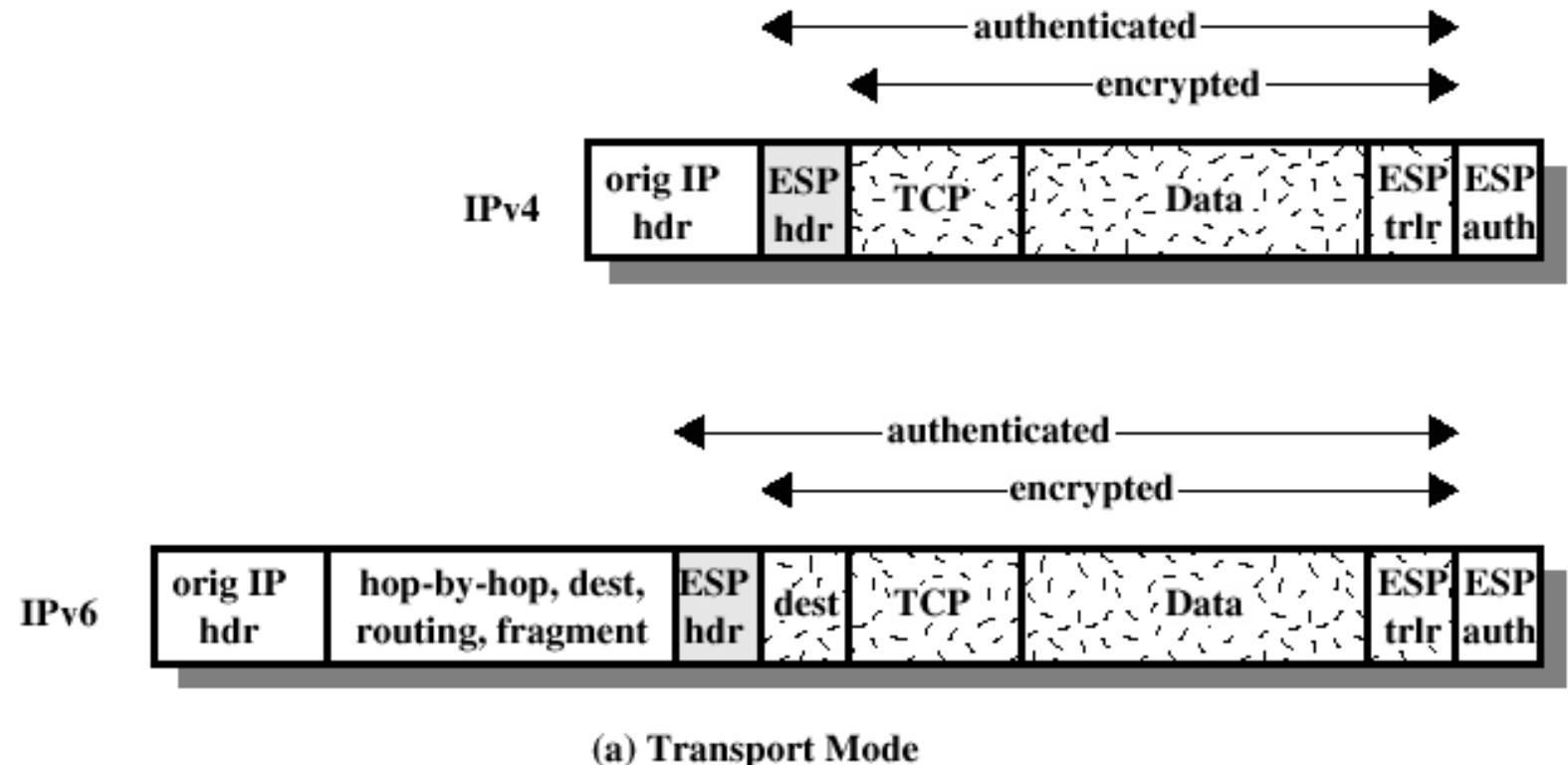
- The new IP header contains different IP addresses than the ultimate destination and source

Encapsulating Security Payload

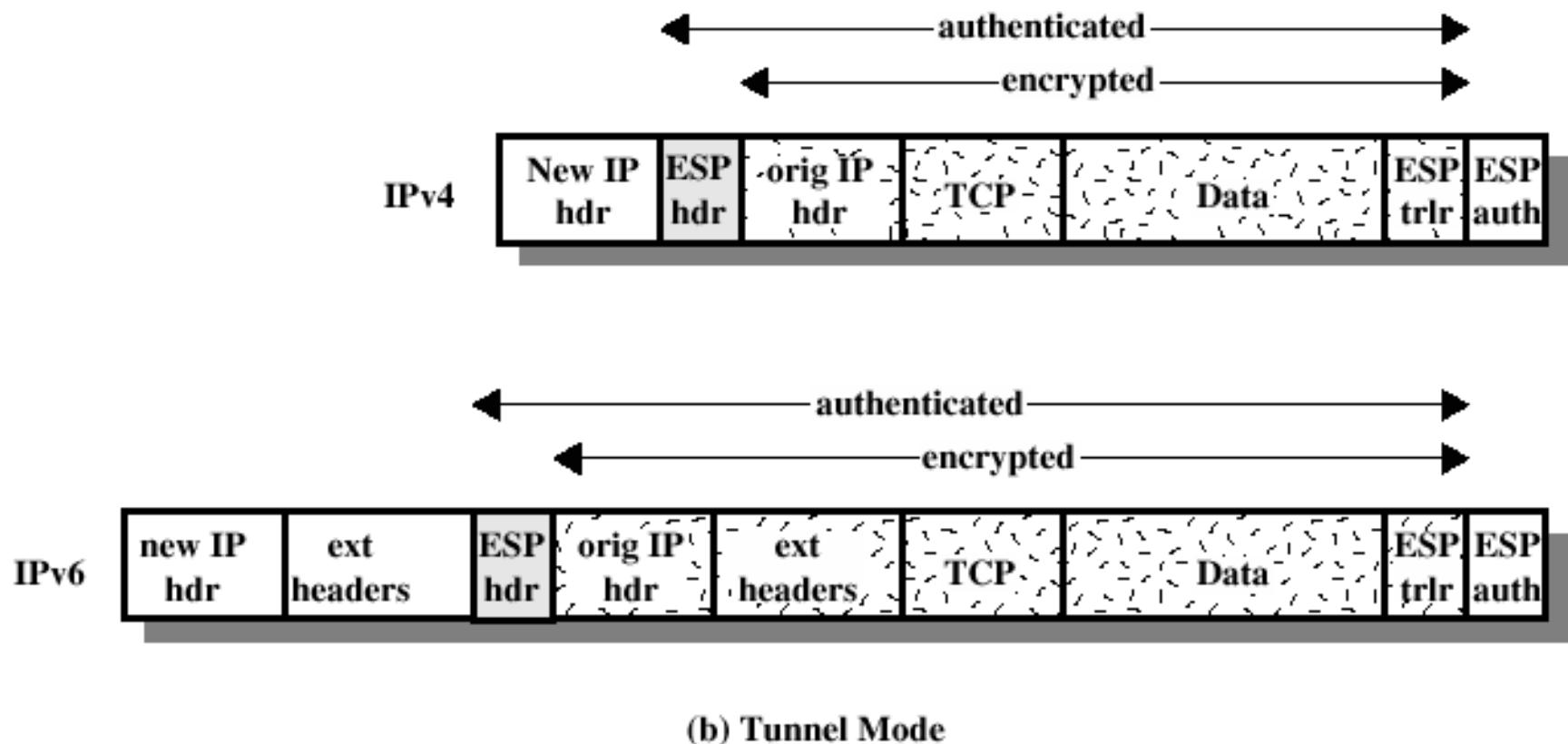
- ESP provides confidentiality services, optionally can provide the same services as AH
- Encryption: 3DES, Blowfish, CAST, IDEA, 3IDEA



ESP Encryption and Authentication: Transport Mode



ESP Encryption and Authentication: Tunnel Mode



TLS/SSL: SECURE END-TO-END COMMUNICATION

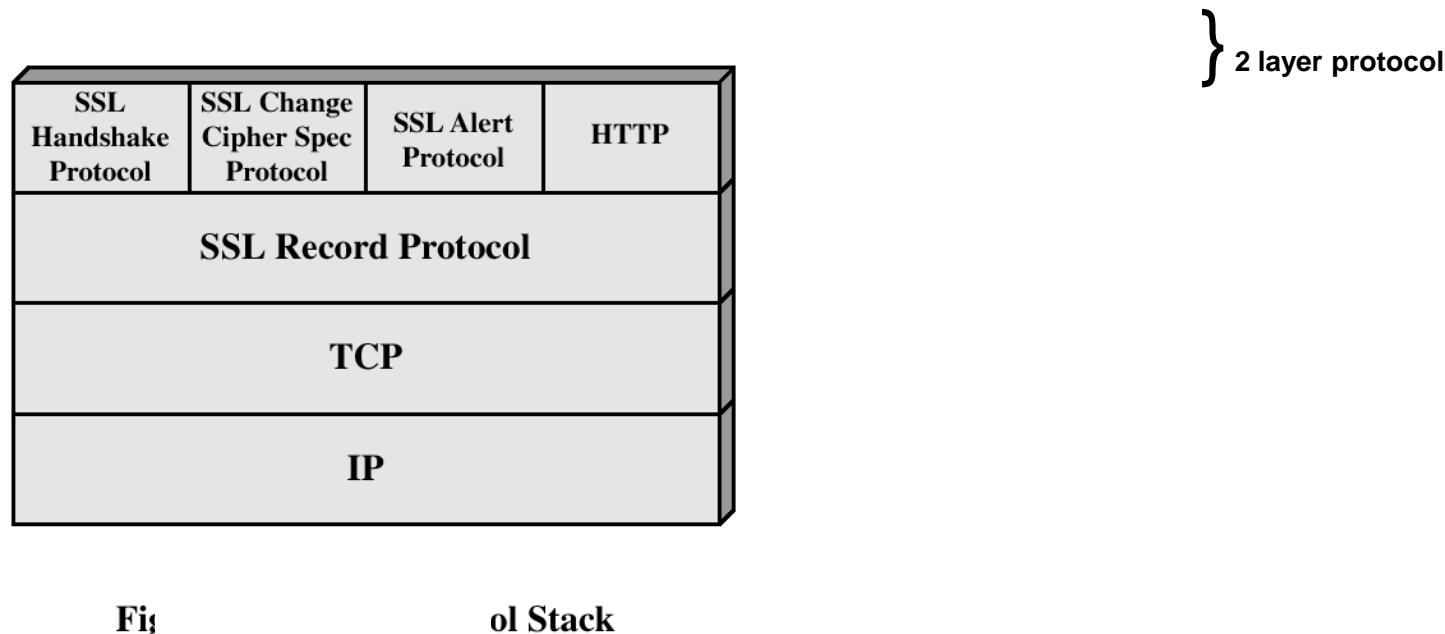
History

- Netscape Communications developed the first three versions of Secure Socket Layer (SSL) with significant assistance from the Web community.
 - Although SSL's development was open, and Netscape encouraged others in the industry to participate, the protocol technically belonged to Netscape.
- Beginning in May **1996**, however, SSL **development became the responsibility** of the Internet Engineering Task Force (IETF).
- The IETF renamed SSL to *Transport Layer Security (TLS)*.
 - *The final version* of the first official TLS specification was released in January **1999**.
- Despite the change of names, TLS is nothing more than a new version of SSL.
 - In fact, there are far fewer differences between TLS **1.0** and SSL 3.0 than there are between SSL 3.0 and SSL 2.0.

TLS/SSL basics

- Protocol suite that allows to establish an end-to-end secure channel:
 - Confidentiality: by encryption using DES, 3DES, RC2, RC4, IDEA.
 - Integrity: by computing a MAC and send it with the message; MD5, SHA1.
 - Key exchange: by public key encryption
- Defines how the characteristics of the channel are negotiated
 - key establishment, encryption cipher, authentication mechanism
- Requires reliable end-to-end protocol, so it runs on top of TCP
- Typically, used by other session protocols (HTTPS ...)
- Several implementations:
 - e.g. SSLeay, open source implementation (www.openssl.org)

TLS: Protocol Architecture



Session and Connection

■ Session

- association between a client and a server
- created by the Handshake Protocol
- defines secure cryptographic parameters that can be shared by multiple connections.

■ Connection

- end-to-end reliable secure communication
- every connection is associated with a session

Session

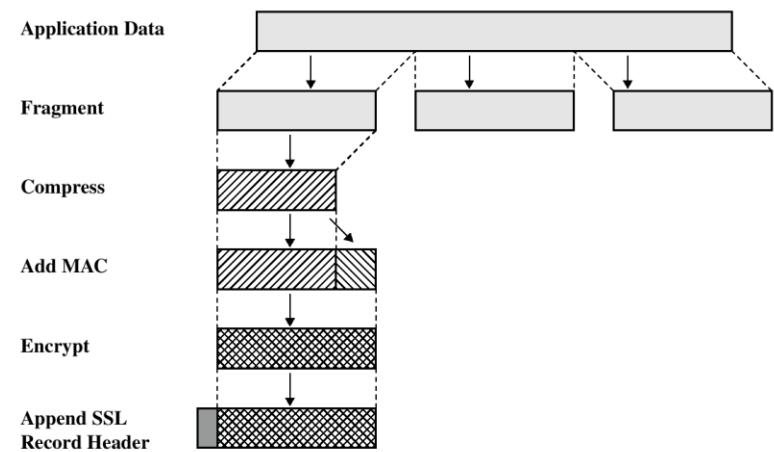
- Session identifier: generated by the server to identify an active or resumable session.
- Peer certificate: X 509v3 certificate.
- Compression method: algorithm used to compress the data before encryption.
- Cipher spec: encryption and hash algorithm, including hash size.
- Master secret: 48 byte secret shared between the client and server.
- Is resumable: indicates if the session can be used to initiate new connections.

Connection

- Server and client random: chosen for each connection.
- Server write MAC secret: shared key used to compute MAC on data sent by the server.
- Client write MAC secret: same as above for the client
- Server write key: shared key used by encryption when server sends data.
- Client write key: same as above for the client.
- Initialization vector: initialization vectors required by encryption.
- Sequence numbers: both server and client maintains such a counter to prevent replay, cycle is $2^{64} - 1$.

TLS: SSL Record Protocol

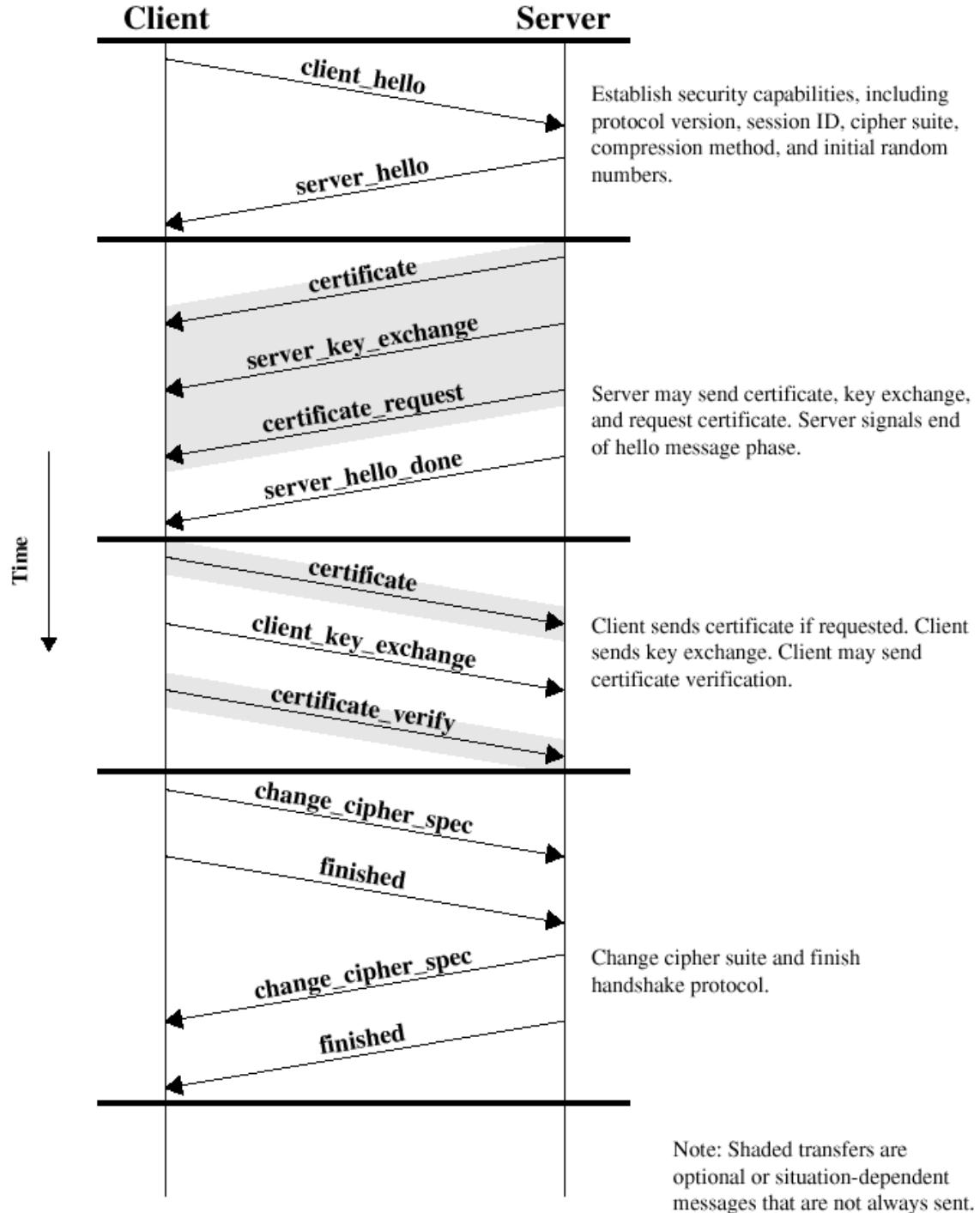
- Provides confidentiality and message integrity using shared keys established by the Handshake Protocol



TLS: Handshake Protocol

- Negotiate Cipher-Suite Algorithms to use
 - Symmetric cipher
 - Key exchange method
 - Message digest function
- Establish the shared master secret
- Optionally authenticate server and/or client

Handshake: At a glance



Handshake: Hellos messages

- Client_hello_message has parameters:
 - Version
 - Random: timestamp + 28-bytes random
 - Session ID
 - CipherSuite: cipher algorithms supported by the client, first is key exchange
 - Compression method
- Server responds with the same
- Client may request use of cached session
 - Server chooses whether to accept or not

Handshake: Key Exchange

- Supported key exchange methods:
 - RSA: shared key encrypted with RSA public key
 - Fixed Diffie-Hellman
 - public parameters provided in a certificate
 - Ephemeral Diffie-Hellman
 - Diffie-Hellman with temporary secret key, messages signed using RSA or DSS
 - Anonymous Diffie-Hellman
 - vulnerable to man-in-the-middle

TLS: Authentication

- Verify identities of participants
 - Client authentication is optional
 - Certificate is used to associate identity with public key and other attributes

TLS: Change Cipher Spec/Finished

- Change Cipher Spec completes the setup of the connections.
- Announce switch to negotiated algorithms and values
- The client sends a message under the new algorithms, allows verification of that the handshake was successful

TLS vs. IPSEC

- Security goals are similar
- IPsec more flexible in services it provides, decouples authentication from encryption
- Different granularity: IPsec operates between hosts, TLS between processes

1. IC có quy luật liên hệ gì với văn bản mà từ đó nó được tính ra. Hãy thử so sánh giữa IC của bản mã sinh ra từ cùng một bản tin qua các hệ mã sau đây (có giải thích):

- ❑ Additive cipher
- ❑ One-time-pad cipher
- ❑ Vigenere cipher with key “loveisblue”
- ❑ Vigenere cipher with key “blue”
- ❑ DES

2. Giả sử ta dùng mật mã khối DES với khóa K và vec-tơ khởi đầu IV để bảo mật các tệp dữ liệu nhị phân trong máy tính. Như vậy một tệp dữ liệu gồm 2 khối 64 bit (X, Y) sẽ mã hóa thành $(DES_K(X), DES_K(Y))$ trong chế độ ECB.

Tệp dữ liệu A có biểu diễn nhị phân (X, Y, Z) , X là các khối 64 bits. Cho biết dạng biểu diễn của A khi ta sử dụng chế độ mã hóa:

- a) ECB b) CBC c) CTR

3. Một sinh viên có “cải tiến” sửa đổi TT trao chuyển khoá Needham-Schroeder như sau:

- 1) $A \rightarrow S: A, B, R_A$
- 2) $S \rightarrow A: E_{K_{AS}}(R_A, B, K, E_{K_{BS}}((K, B)))$
- 3) $A \rightarrow B: E_{K_{BS}}(K, A)$
- 4) $B \rightarrow A: E_K(R_B)$
- 5) $A \rightarrow B: E_K(2^*R_B)$

Cho biết những chỗ sai/không hợp lý

- 1. Tại sao nói độ an toàn của một hệ chữ ký điện tử phụ thuộc vào việc lựa chọn hàm băm?
 - Giả sử hàm băm H với đầu ra 64bit được Alice chọn xây dựng chữ ký điện tử. Eve là một kẻ gian muốn lừa dối Alice bằng cách tạo ra hai văn bản X và Y có nội dung đối nghịch nhau nhưng chữ ký của Alice lên cả 2 sẽ giống nhau. Eve có khả năng làm điều này hay không nếu có thời gian chuẩn bị khoảng 1 ngày (từ lúc biết đặc tả của H đến lúc gặp Alice), trong đó Eve có khả năng tính toán mỗi phép băm H mất 10^{-6} s ?
 - 2. Phát biểu định lý Euler và ý nghĩa ứng dụng của nó với việc xây dựng hệ khóa công khai. Hãy cho biết giá trị và nêu cách tính giá trị của:
a) $\varphi(131)$ b) $\varphi(133)$ c) $\varphi(30)$
 - 3. Một sinh viên có “cải tiến” sửa đổi TT trao chuyển khoá Needham-Schroeder như sau:
 - 1) $A \rightarrow S: A, B, R_A$
 - 2) $S \rightarrow A: E_{K_{AS}}(R_A, B, K, E_{K_{BS}}((K,B))$
 - 3) $A \rightarrow B: E_{K_{BS}}(K,A)$
 - 4) $B \rightarrow A: E_K(R_B)$
 - 5) $A \rightarrow B: E_K(2^*R_B)$
- Cho biết những chỗ sai/không hợp lý

Information Security

Van K Nguyen - HUT

Program (Software) Security

MALICIOUS PROGRAMS

- **Malware: software designed to infiltrate or damage a computer system without the owner's informed consent**
- **Spyware: software designed to intercept or take partial control over the user's interaction with the computer, without the user's informed consent**
 - secretly monitors the user's behavior
 - collect various types of personal information

Trapdoor/backdoor

- Secret entry point into a system
 - Special login into system (circumvents normal security procedures.)
- Presents a security risk
- Can be for good purpose as for Troubleshooting or maintenance
- Can be bad in wrong hand - Malicious intent

Logic bomb

- Embedded in legitimate programs
- Activated when specified conditions met
 - E.g., presence/absence of some file; Particular date/time or particular user
- When triggered, typically damages system:
Modify/delete files/disks

Trojan Horse

- Program with an covert effect besides the expected
 - Appears normal/expected
 - Covert effect violates security policy
- User tricked into executing a trojan horse
 - Look normal but behind the scene, covert effect performed with user's authorization

Virus

- Self-replicating code
 - Like replicating Trojan horse
 - Alters normal code with “infected” version
- Generally tries to remain undetected
- Operates when infected code executed
 - If *spread condition* then
 - For *target files*
 - if *not infected* then *alter to include virus*
 - Perform malicious action
 - Execute normal program

Virus types

- Problem: How to ensure virus “carrier” executed?
 - Place in boot sector of disk OR in executales which are likely to be used
- Boot Sector
 - Run on any boot
 - Propagate by altering boot disk creation
- Executable
 - Malicious code placed at beginning of legitimate program
 - Runs when application run
 - Application then runs normally

Virus Types

- Terminate but Stay Resident (TSRs)
 - Stays active in memory after application completes
 - Allows infection of previously unknown files
 - Trap calls that execute a program
- Stealth
 - Conceal Infection
 - Trap read and disinfect
 - Let execute call infected file
 - Encrypt virus
 - Prevents “signature” to detect virus
 - Polymorphism
 - Change virus code to prevent signature

Macro Virus

- Infected “executable” isn’t machine code
 - Relies on something “executed” inside application data → Macros
- Properties specific to these viruses
 - Architecture-independent
 - Application-dependent

Worms

- Runs independently
 - Does not require a host program
 - Propagates a fully working version of itself to other machines
- Carries a payload performing hidden tasks
 - Backdoors, spam relays, DDoS agents; ...
- Phases
 - Probing → Exploitation → Replication → Payload

Cost of Worm Attacks

- Morris worm, 1988
 - Infected approximately 6,000 machines
 - 10% of computers connected to the Internet
 - cost ~ \$10 million in downtime and cleanup
- Code Red worm, July 16 2001
 - Direct descendant of Morris' worm
 - Infected more than 500,000 servers
 - Caused ~ \$2.6 Billion in damages,
- Love Bug worm: May 3, 2000, \$8.75 billion

Statistics: Computer Economics Inc., Carlsbad, California

Morris Worm

- Released November 1988
 - Program spread through Digital, Sun workstations
 - Exploited Unix security vulnerabilities
- Consequences
 - No immediate damage from program itself
 - Replication and threat of damage
 - Load on network, systems used in attack
 - Many systems shut down to prevent further attack

Morris Worm

- Two parts
 - Program to spread worm
 - look for other machines that could be infected
 - try to find ways of infiltrating these machines
 - Vector program (99 lines of C)
 - compiled and run on the infected machines
 - transferred main program to continue attack
- Security vulnerabilities
 - fingerd – Unix finger daemon
 - sendmail - mail distribution program
 - Trusted logins (.rhosts)
 - Weak passwords

Morris Worm: Spread Mechanisms

■ Sendmail

- Exploit debug option in sendmail to allow shell access

■ Fingerd

- Exploit a buffer overflow in the gets function
- Apparently, this was the most successful attack

■ Rsh

- Exploit trusted hosts
- Password cracking

sendmail

- Worm used debug feature
 - Opens TCP connection to machine's SMTP port
 - Invokes debug mode
 - places 40-line C program in a temporary file
 - Compiles and executes this program
 - Opens socket to machine that sent script
 - Retrieves worm main program, compiles it and runs

Finger

- *An utility that allows users to obtain information about other users.*
 - the full name or login name of a user
 - whether or not a user is currently logged in,
 - telephone numbers, maybe, and other info
- *fingerd: a daemon, or background process, to service remote requests using the finger protocol*
- The bug exploited to break *fingerd: overrunning the buffer for input*
 - Gets, a standard C library, *takes input to a buffer without doing any bounds checking*

fingerd

- Array bounds attack
 - Fingerd expects an input string
 - Worm writes long string to internal 512-byte buffer
- Attack string
 - Includes machine instructions
 - Overwrites return address
 - Invokes a remote shell
 - Executes privileged commands

Remote shell

- Unix trust information
 - /etc/host.equiv – system wide trusted hosts file
 - /.rhosts and ~/.rhosts – users' trusted hosts file
- Worm exploited trust information
 - Examining files that listed trusted machines
 - Assume reciprocal trust
 - If X trusts Y, then maybe Y trusts X
- Password cracking
 - Worm was running as daemon (not root) so needed to break into accounts to use .rhosts feature
 - Read /etc/passwd, used ~400 common password strings & local dictionary to do a dictionary attack

The worm itself

- Program is shown as 'sh' when ps
 - Clobbers argv array so a 'ps' will not show its name
 - Opens its files, then unlinks (deletes) them so can't be found
 - Since files are open, worm can still access their contents
- Tries to infect as many other hosts as possible
- When worm successfully connects, forks a child to continue the infection while the parent keeps trying new hosts
- find targets using several mechanisms: 'netstat -r -n', /etc/hosts,
...
■ Worm did not:
 - Delete system's files, modify existing files, install trojan horses, record or transmit decrypted passwords, capture superuser privileges

Detecting Morris Internet Worm

■ Files

- Strange files appeared in infected systems
- Strange log messages for certain programs

■ System load

- Infection generates a number of processes
- Password cracking uses lots of resources
- Systems were reinfected => number of processes grew and systems became overloaded
 - Apparently not intended by worm's creator

Thousands of systems were shut down

Buffer Overflow

- Buffer overflow occurs when a program or process tries to store more data in a buffer than the buffer can hold
- Very dangerous because the extra information may:
 - Affect user's data
 - Affect user's code
 - Affect system's data
 - Affect system's code

Why Does Buffer Overflow Happen?

- No check on boundaries
 - Programming languages give user too much control
 - Programming languages have unsafe functions
 - Users do not write safe code
- C and C++, are more vulnerable because they provide no built-in protection against accessing or overwriting data in any part of memory

Why Buffer Overflow Matter

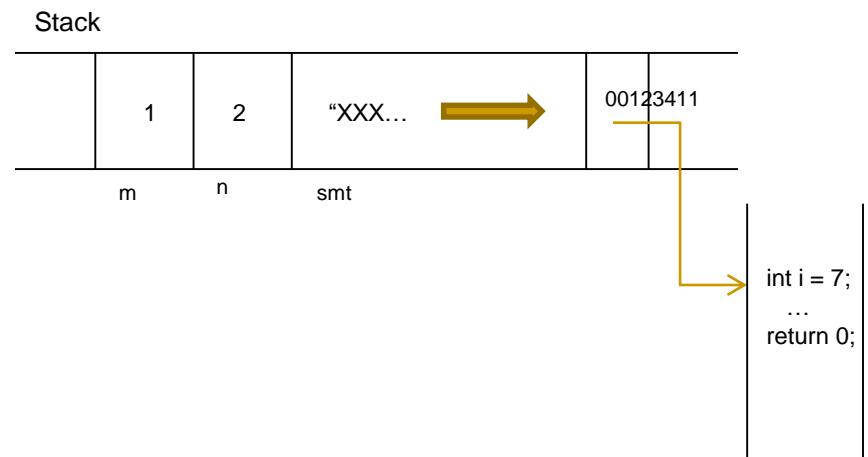
- Overwrites:
 - other buffers
 - variables
 - program flow data
- Results in:
 - erratic program behavior
 - a memory access exception
 - program termination
 - incorrect results
 - **breach of system security**

Basic Example

- A program has defined two data items which are adjacent in memory
 - an 8-byte-long string buffer, A, and a two-byte integer, B.
 - Initially, A contains nothing but zero bytes, and B contains the number 3
- Now, the program attempts to store the character string "excessive" in the A buffer, followed by a zero byte to mark the end of the string
 - By not checking the length of the string, it overwrites the value of B

Stack-based exploitation

- A malicious user may exploit stack-based buffer overflows to manipulate the program in one of several ways:
 - By overwriting a local variable that is near the buffer in memory on the stack to change the behaviour of the program which may benefit the attacker.
 - By overwriting the return address in a stack frame.
Once the function returns, execution will resume at the return address as specified by the attacker, usually a user input filled buffer.
 - By overwriting a function pointer,^[1] or exception handler, which is subsequently executed.



WEB SECURITY

SQL injection

- **SQL injection** is a code injection technique that exploits a security vulnerability occurring in the database layer of an application.
- The vulnerability is present when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and thereby unexpectedly executed.
- It is an instance of a more general class of vulnerabilities that can occur whenever one programming or scripting language is embedded inside another.
- SQL injection attacks are also known as SQL insertion attacks.

Example

- Consider: `SELECT * FROM users WHERE name = 'a' OR 't'='t';`
- Set **username** as: `a` or `'t'='t'`
- Then get: `SELECT * FROM users WHERE name = 'a' OR 't'='t';`

Another example

- Use:

a';DROP TABLE users; SELECT * FROM data
WHERE 't' = 't'

So:

SELECT * FROM users WHERE name =
'a';DROP TABLE users; SELECT * FROM
DATA WHERE 't' = 't';

Cross Site Scripting (XSS)

■ Recall the basics

- scripts embedded in web pages run in browsers
- scripts can access cookies
 - get private information
- and manipulate DOM objects
 - controls what users see
- scripts controlled by the same-origin policy

■ Why would XSS occur

- Web applications often take user inputs and use them as part of webpage

Why XSS

- Name originated from the fact that a malicious web site could load another web site into another frame or window, then use Javascript to read/write data on the other web site
- The definition changed to mean the injection of HTML/Javascript into a web page

Example: Exploiting Social Network

- 1. Bad guy posts a message
- 2. When good guy reads the message, bad guy steals the cookie that contains information about authentication



Computer Security

Tuan M. Nguyen & Van K. Nguyen

Database Security

Content

- Introduction to Databases
- Security Requirements
- Database Security Levels
- Reliability and Integrity
- Attacks against database

Introduction to Databases

- Database – collection of *data* and set of *rules* that organize the data by specifying certain relationships among the data.
- Database administrator (DBA)
- Database management system (DBMS) – database manager

Introduction to Databases

- Records – contain related group of data
- Fields – elementary data items
- Schema – logical structure of database
- Subschema – view into database

Name	First	Address	City	State	Zip	Airport
ADAMS	Charles	212 Market St.	Columbus	OH	43210	CMH
BENCHLY	Zeke	501 Union St.	Chicago	IL	60603	ORD
CARTER	Marlene	411 Elm St.	Columbus	OH	43210	CMH

Introduction to Databases

■ Queries

- Commands that retrieve, modify, add, or delete fields and records of the database
- Most query languages are based on SQL, a structured query language.
- `SELECT NAME = 'ADAMS'`
- `SELECT (ZIP = '43210') ^ (NAME = 'ADAMS')`

Introduction to Databases

- Advantages of Using Databases
 - Shared access
 - Minimal redundancy
 - Data consistency
 - Data integrity
 - Controlled access

Security Requirements

- Physical database integrity
- Logical database integrity
- Element integrity
- Auditability
- Access control
- User authentication
- Availability

Security Requirements

■ Integrity of the Database

- Users must be able to trust the accuracy of the data values
- Updates are performed by authorized individuals
- Integrity is the responsibility of the DBMS, the OS, and the computing system manager
- Must be able to reconstruct the database at the point of a failure

Security Requirements

■ Element Integrity

- Correctness or accuracy of elements
- Field checks
- Access control
- Maintain a change log – list every change made to the database

Security Requirements

■ Auditability & Access Control

- Desirable to generate an audit record of all access to the database (reads/writes)
- **Pass-through problem** – accessing a record or element without transferring the data received to the user (no reads/writes)
- Databases separated logically by user access privileges

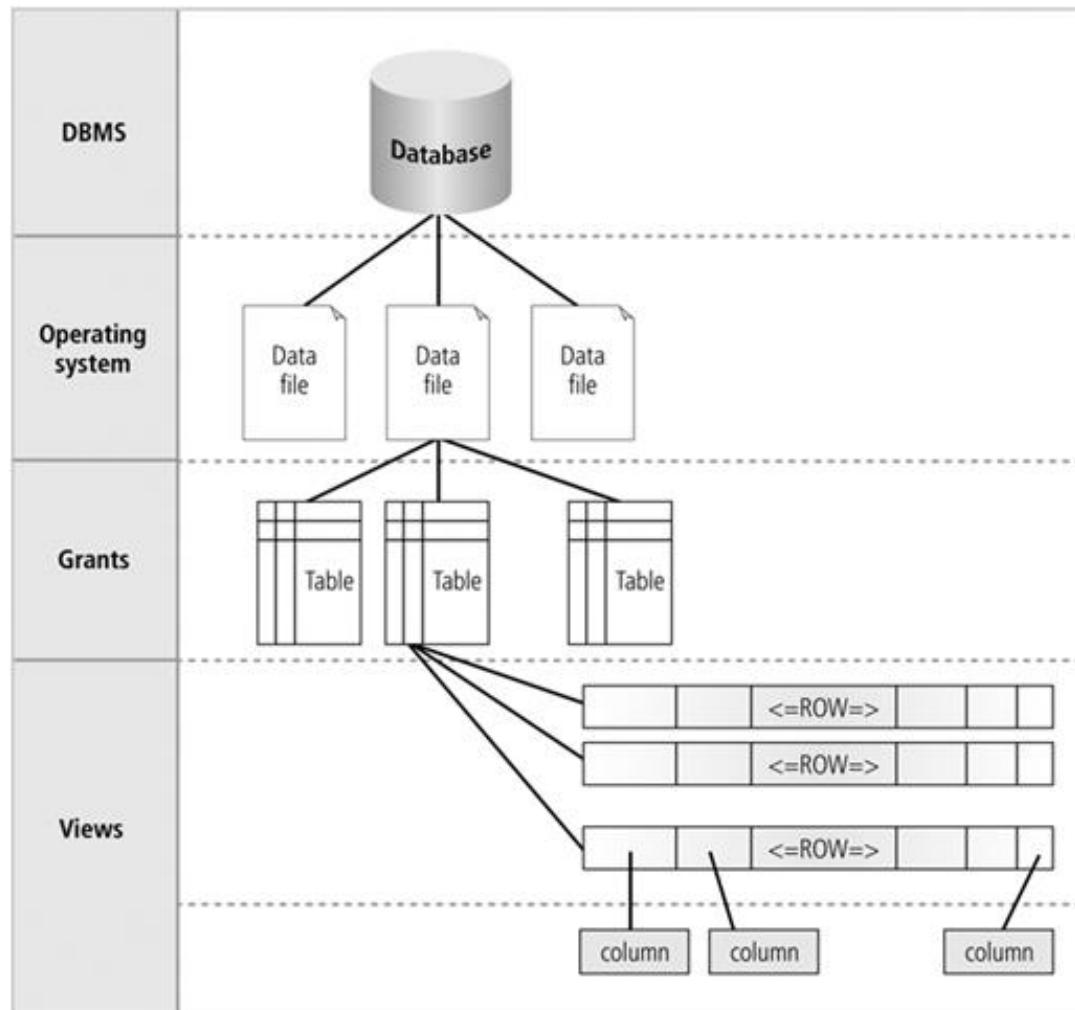
Security Requirements

- Other Security Requirements
 - User Authentication
 - Confidentiality
 - Availability

Database Security Levels

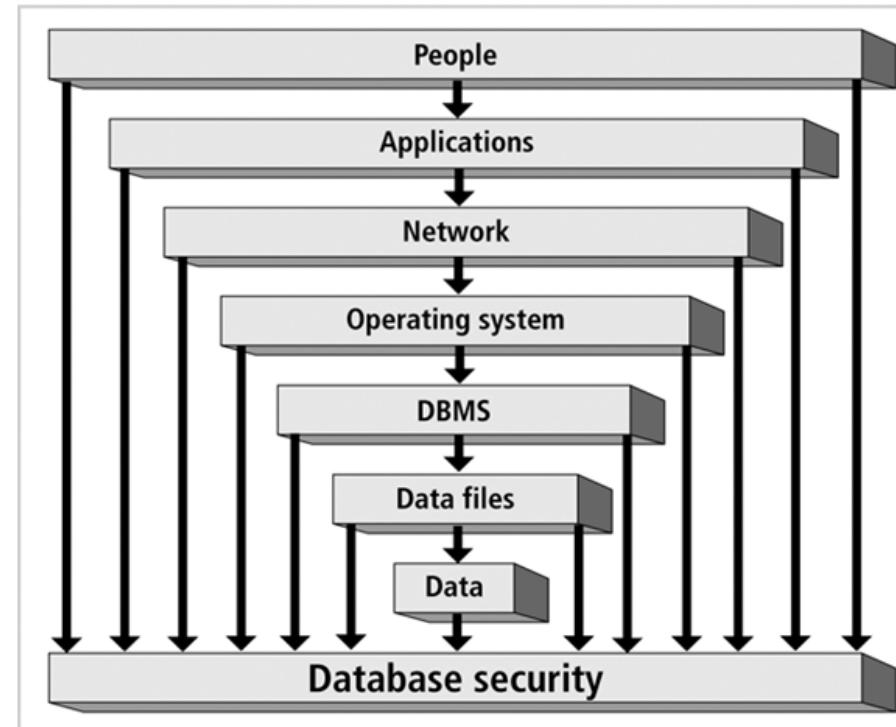
- Relational database: collection of related data files
- Data file: collection of related tables
- Table: collection of related rows (records)
- Row: collection of related columns (fields)

Database Security Levels



Menaces to Databases

- Security access point: place where database security must be protected and applied

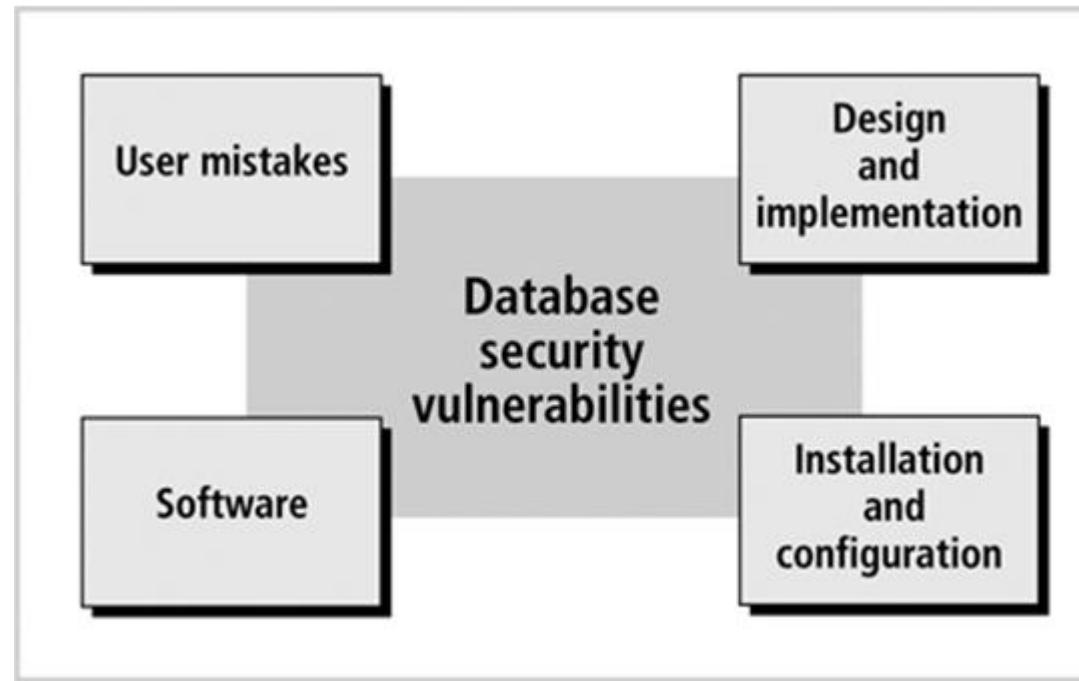


Menaces to Databases

- Security gaps: points at which security is missing
- Vulnerabilities: kinks in the system that can become threats
- Threat: security risk that can become a system breach

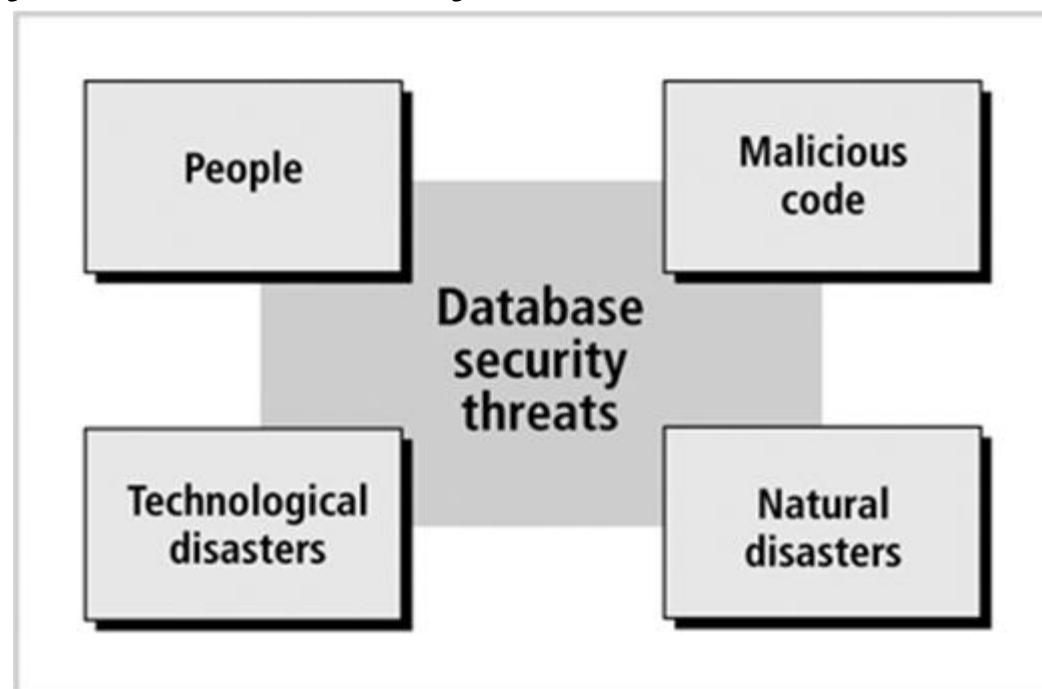
Menaces to Databases

- Security vulnerability: a weakness in any information system component



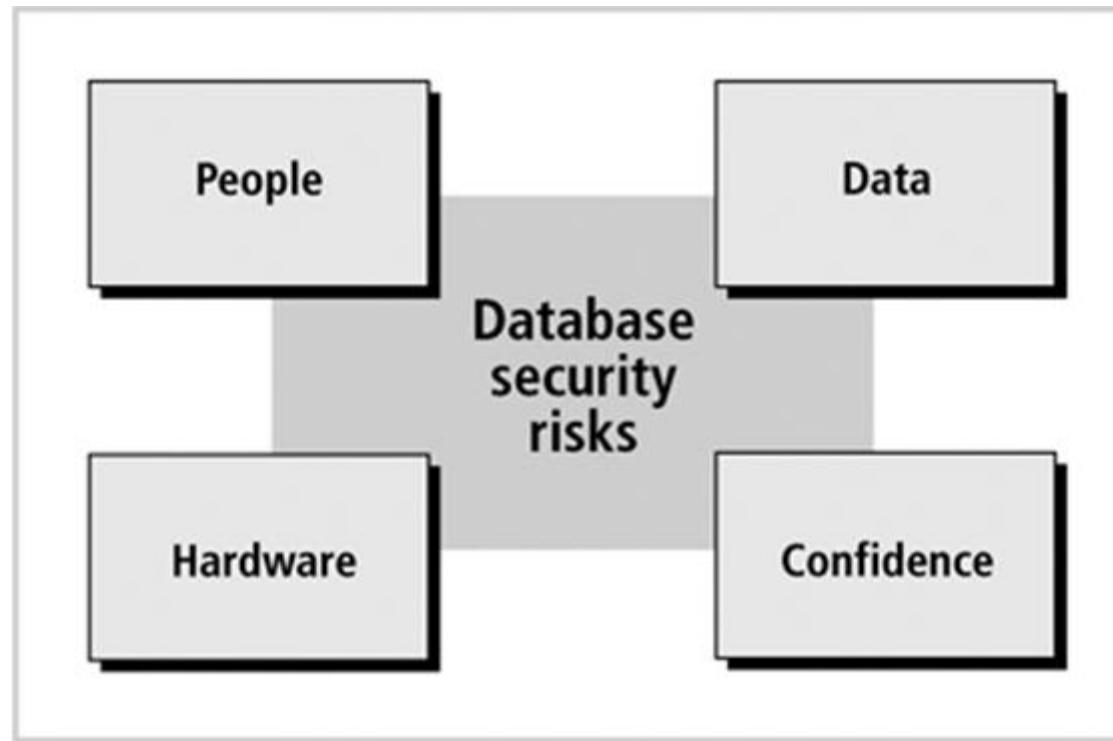
Menaces to Databases

- Security threat: a security violation or attack that can happen any time because of a security vulnerability



Menaces to Databases

- Security risk: a known security gap intentionally left open



Reliability and Integrity

- Database concerns
 - Database integrity
 - Element integrity
 - Element accuracy
- Some protection from OS
 - File access
 - Data integrity checks

Reliability and Integrity

- Two-Phase Update
 - Failure of computing system in middle of modifying data
 - Intent Phase – gather resources needed for update; write **commit flag** to the database
 - Update Phase – make permanent changes

Reliability and Integrity

- Redundancy / Internal Consistency
 - Error detection / Correction codes (parity bits, Hamming codes, CRCs)
 - Shadow fields
 - Log of user accesses and changes

Reliability and Integrity

■ Concurrency/Consistency

- Access by two users sharing the same database must be constrained (lock)
- Monitors –check entered values to ensure consistency with rest of DB
- State Constraints – describes condition of database (unique employee #)
- Transition Constraints – conditions before changes are applied to DB

Reliability and Integrity

- Access control
 - Physical access
 - Access control techniques
 - Discretionary Access Control (DAC)
 - Mandatory Access Control (MAC)
 - Role Based Access Control (RBAC)
- Identification and authentication
- Authorization
- Accountability

Attacks against database

■ Inference Attack

- ❑ Data mining technique performed by analyzing data in order to illegitimately gain knowledge about a subject or database Identification and authentication

■ SQL injection

- ❑ Code injection technique that exploits a security vulnerability occurring in the database layer of an application.

Attacks against database

■ Inference Attack

- Direct Attack: tries to determine values of sensitive fields by seeking them directly with queries that yield few records
 - List NAME where SEX=M ^ DRUGS=1
 - List NAME where (SEX=M ^ DRUGS=1) v (SEX#M ^ SEX#F) v (DORM=AYRES)

Attacks against database

■ Inference Attack

- Sum
 - Show STUDENT-AID WHERE SEX=F ^ DORM=Grey
- Count
 - Show Count, STUDENT-AID WHERE SEX=M ^ DORM=Holmes
 - List NAME where (SEX=M ^ DORM=Holmes)
- Median
- Tracker Attacks – using additional queries that produce small results

Attacks against database

■ Inference Attack - Controls

- Suppression – don't provide sensitive data
- Concealing – don't provide actual values
- Limited Response Suppression
- Combined Results
 - Sums
 - Ranges
 - Rounding
- Query Analysis – “should the result be provided”

Attacks against database

■ SQL Injection

- Incorrectly filtered escape characters
 - `SELECT * FROM users WHERE name = 'a' OR 't'='t';`
- Incorrect type handling
 - `statement := "SELECT * FROM data WHERE id = " +
a_variable + ";"`
 - `SELECT * FROM DATA WHERE id=1;DROP TABLE
users;`
- Vulnerabilities inside the database server

Attacks against database

- SQL Injection - preventing
 - Parameterized statements
 - Enforcement at the database level
 - Enforcement at the coding level
 - *escape* dangerous characters
 - Check user's inputs

Summary

- Database security: degree to which data is fully protected from tampering or unauthorized acts
- Enforce security at all database levels
- Data requires highest level of protection: data access point must be small

