




CHƯƠNG IV. ĐẢM BẢO CHẤT LƯỢNG (5 LT + 2 BT)

- I. Nhất quán và sao lưu (Consistency & Replication)
 - II. Dung lỗi (Fault tolerance)
 - III. An toàn an ninh (Security)
- 



I. SAO LƯU VÀ TÍNH NHẤT QUÁN (REPLICATION & CONSISTENCY)



I. Sao lưu và tính nhất quán

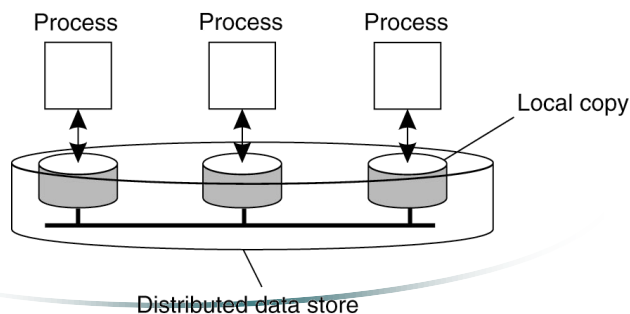
1. Đặt vấn đề

- Sao lưu dữ liệu (Data replication): kỹ thuật phổ biến trong các HPT
 - Tăng độ tin cậy (Reliability)
 - Nếu 1 bản sao (replica) trở nên không sẵn dùng hay bị hỏng, có thể dùng bản khác
 - Chống lại hiện tượng ngắt dữ liệu
 - Hiệu năng (Performance)
 - Mở rộng kích thước HPT (vd: replicated web servers)
 - Mở rộng phạm vi địa lý của HPT (vd: web proxies)
- → Cần duy trì tính nhất quán (consistency) của các dữ liệu sao lưu
 - Nếu 1 bản sao thay đổi, các bản sao khác trở nên không nhất quán

I. Sao lưu và tính nhất quán

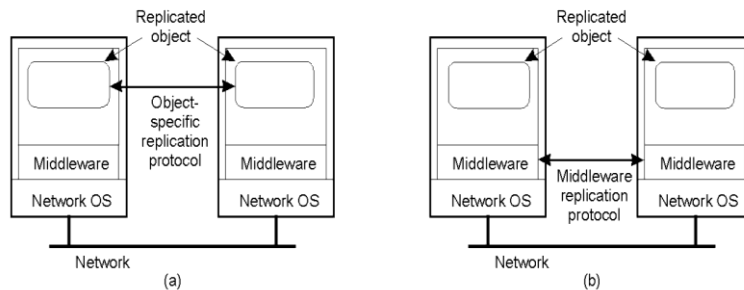
1. Đặt vấn đề

- Tính nhất quán rất quan trọng với các thao tác đọc/ghi trên bộ nhớ dùng chung
- Mô hình nhất quán là hợp đồng giữa các tiến trình và kho dữ liệu
- Ràng buộc về tính nhất quán/ ngữ nghĩa nhất quán
 - Nếu các tiến trình làm đúng luật đã giao hẹn, kho dữ liệu sẽ hoạt động chính xác



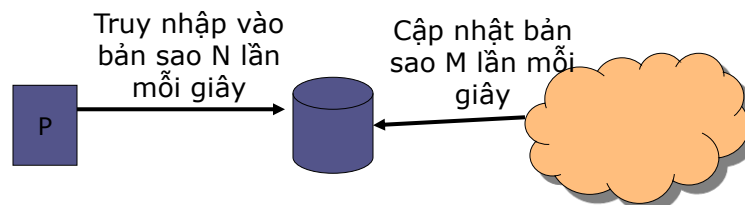
2. Sao lưu đối tượng (Object Replication)

- Cách 1: ứng dụng chịu trách nhiệm sao lưu
- Cách 2: hệ thống hoặc phần dẻo chịu trách nhiệm sao lưu



Chi phí

- Sao lưu là kỹ thuật dùng để mở rộng phạm vi, nên không phải lúc nào cũng dùng được

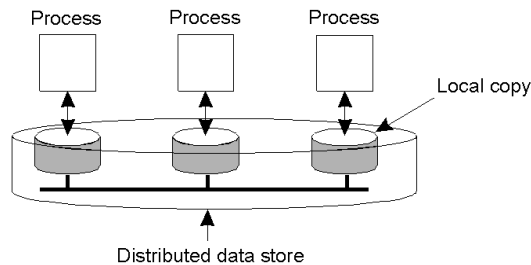


- Điều gì xảy ra nếu $N \ll M$?



3. Các mô hình nhất quán lấy dữ liệu làm trung tâm (Data-Centric Consistency Models)

- Cho phép các tiến trình trong hệ thống thấy kho dữ liệu nhất quán.
- Mục đích của các mô hình này là khi có yêu cầu đọc dữ liệu, kết quả trả về phải là dữ liệu được ghi sau cùng (last write)
 - Khác nhau về khái niệm và cách xác định dữ liệu được ghi sau cùng



a. Nhất quán chặt chẽ (Strict Consistency)

- Kết quả trả về cho bất cứ yêu cầu đọc nào cũng là dữ liệu được ghi mới nhất.
 - Ngầm giả thiết có 1 đồng hồ chung để đồng bộ giữa các tiến trình
 - Tất cả các tiến trình đều quan sát được thao tác ghi dữ liệu sẽ
 - Khó thực hiện trong thực tế, do thời gian trễ mạng là khác nhau

1. Nhất quán và sao lưu
1. Đặt vấn đề
2. Sao lưu đối tượng
- 3. Các mô hình nhất quán lấy dữ liệu làm trung tâm**

b. Nhất quán tuần tự (Sequential Consistency)

- Giả thiết rằng tất cả các thao tác được thực hiện theo trình tự nhất định và mỗi tiến trình sẽ thực hiện các thao tác theo đúng thứ tự chương trình
 - Được phép thay đổi thứ tự thực hiện các thao tác nếu không ảnh hưởng đến thứ tự đã định
 - Tất cả tiến trình đều phải nhất trí với sự thay đổi này
 - Mỗi tiến trình phải bảo lưu thứ tự thực hiện của nó
 - Không tiến trình nào biết dữ liệu được ghi mới nhất (most recent write)
- Yếu hơn nhất quán chặt chẽ

P1:	W(x)a	
P2:	W(x)b	
P3:	R(x)b	R(x)a
P4:	R(x)b	R(x)a

(a)

P1:	W(x)a	
P2:	W(x)b	
P3:	R(x)b	R(x)a
P4:	R(x)a	R(x)b

(b)

c. Nhất quán tuyến tính (Linearizability)

- Giả thiết nhất quán tuần tự và
 - Nếu $TS(x) < TS(y)$ thì thực hiện $OP(x)$ trước $OP(y)$
 - Mạnh hơn nhất quán tuần tự
 - Khác biệt giữa nhất quán tuần tự và nhất quán tuyến tính
 - reads/writes vs transactions
- Ví dụ:

Process P1	Process P2	Process P3
x = 1; print (y, z);	y = 1; print (x, z);	z = 1; print (x, y);

Ví dụ: Nhất quán tuần tự và nhất quán tuyến tính

- 4 dãy lệnh hợp lệ tương ứng với các tiến trình miêu tả trong slide trước

	$x = 1;$ $\text{print}((y, z);$ $y = 1;$ $\text{print}(x, z);$ $z = 1;$ $\text{print}(x, y);$	$x = 1;$ $y = 1;$ $\text{print}(x, z);$ $\text{print}(y, z);$ $z = 1;$ $\text{print}(x, y);$	$y = 1;$ $z = 1;$ $\text{print}(x, y);$ $\text{print}(x, z);$ $x = 1;$ $\text{print}(y, z);$	$y = 1;$ $x = 1;$ $z = 1;$ $\text{print}(x, z);$ $\text{print}(y, z);$ $\text{print}(x, y);$
time	Prints: 001011	Prints: 101011	Prints: 010111	Prints: 111111
	Signature: 001011 (a)	Signature: 101011 (b)	Signature: 110101 (c)	Signature: 111111 (d)

d. Nhất quán nhân quả (Causal consistency)

1. Nhất quán và sao lưu
2. Đặt vấn đề
3. Sao lưu đối tượng
3. Các mô hình nhất quán lấy dữ liệu làm trung tâm

- Tất cả các tiến trình có cùng thứ tự thực hiện thao tác ghi sẽ nhìn thấy thao tác ghi liên quan
 - Các máy khác nhau có thể nhìn thấy các thao tác ghi đồng thời ở thứ tự khác nhau

P1:	W(x)a		
P2:		R(x)a	W(x)b
P3:			R(x)b R(x)a
P4:			R(x)a R(x)b

(a)

P1:	W(x)a		
P2:		W(x)b	
P3:			R(x)b R(x)a
P4:			R(x)a R(x)b

(b)

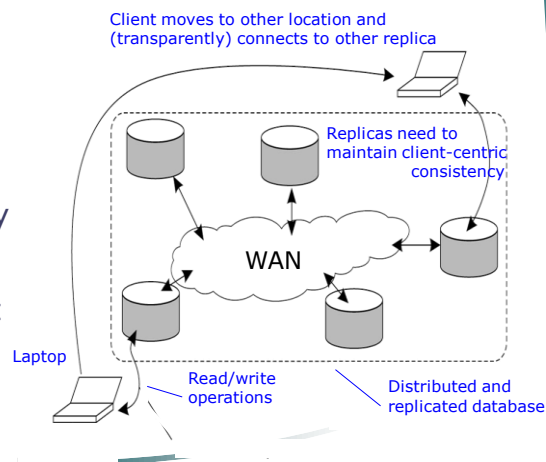
Đặc điểm chung

- I. Nhất quán và sao lưu
1. Đặt vấn đề
 2. Sao lưu đối tượng
 3. Các mô hình nhất quán lấy dữ liệu làm trung tâm
 4. Các mô hình nhất quán lấy người dùng làm trung tâm

- Hạn chế sự không nhất quán về dữ liệu theo cách ít tổn chi phí nhất
- Giảm thiểu các ràng buộc về tính nhất quán (weak consistency)
 - Dữ liệu không cần nhất quán mọi nơi, mọi lúc (nhất quán tức thời hay eventual consistency).
- Áp dụng cho các kho dữ liệu phân tán có đặc điểm như sau:
 - Ít khi cần cập nhật dữ liệu đồng thời giữa các bản sao
 - Hầu hết các thao tác là đọc dữ liệu

Vấn đề tồn tại với mô hình nhất quán tức thời

- Mô hình nhất quán tức thời có tốt cho các khách hàng là các thiết bị di động hay không?
 - Không tốt lắm: có dữ liệu bị mất (go "backwards", etc.).
- Mô hình nhất quán lấy người dùng làm trung tâm có thể giải quyết vấn đề này: tính nhất quán được đảm bảo cho từng khách hàng.



Nhất quán cho từng người dùng

- Giả thiết:
 - Kho dữ liệu phân tán vật lý trên nhiều máy
 - Khi 1 tiến trình truy cập vào kho dữ liệu, nó kết nối đến phiên bản tại chỗ hoặc phiên bản trên máy gần nhất. Tất cả các thao tác đều được thực hiện trên phiên bản này
- $x_i[t]$: phiên bản của x tại vị trí L_i vào thời điểm t .
- $WS(x_i[t])$: chuỗi thao tác kể từ khi kết nối đến L_i ; chuỗi này tạo ra $x_i[t]$.
- $WS(x_i[t_1]; x_j[t_2])$: các thao tác trong chuỗi $WS(x_i[t])$, được thực hiện tại L_j vào thời điểm t_2 (sau t).

a. Monotonic Reads

- Kho dữ liệu thỏa mãn các điều kiện sau đây được coi là đọc dữ liệu một cách nhất quán:
 - Nếu 1 tiến trình đọc giá trị của một mục dữ liệu x , tất cả các thao tác đọc x sau đó sẽ luôn trả về giá trị giống giá trị đã đọc hoặc giá trị mới nhất
- Ví dụ: 1 khách hàng mở hộp thư điện tử tại San Francisco, sau đó bay sang New York. Có khi nào anh ta thấy phiên bản cũ hơn của mailbox hay không?
 - Nếu 1 tiến trình đọc được giá trị của x tại thời điểm t , sau đó nó sẽ không bao giờ nhìn thấy giá trị nào cũ hơn của x .

Đâu là mô hình đọc nhất quán?

- Một tiến trình thực hiện các thao tác đọc dữ liệu trên 2 bản sao tại 2 vị trí khác nhau của cùng 1 kho dữ liệu

L1:	WS(x_1)	$R(x_1)$
L2:	WS($x_1; x_2$)	$R(x_2)$

L1:	WS(x_1)	$R(x_1)$
L2:	WS(x_2)	$R(x_2)$

b. Monotonic Writes

- Kho dữ liệu thỏa mãn các điều kiện sau đây được coi là ghi dữ liệu một cách nhất quán:
 - 1 TT phải hoàn tất việc thực hiện 1 thao tác ghi dữ liệu x trước khi thực hiện bất kỳ 1 thao tác ghi dữ liệu nào đó trên x
- Đâu là mô hình ghi nhất quán ?

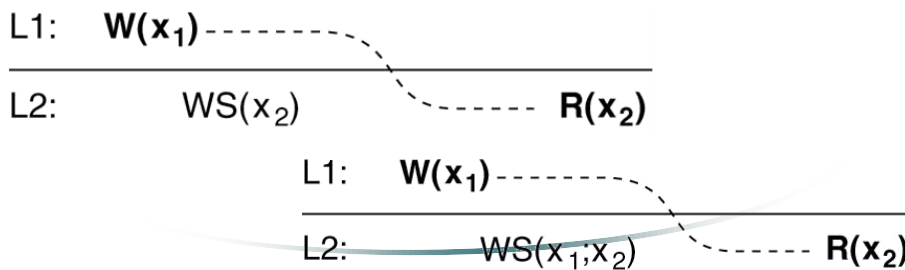
L1:	$W(x_1)$
L2:	WS(x_1)

L1:	$W(x_1)$
L2:	$W(x_2)$



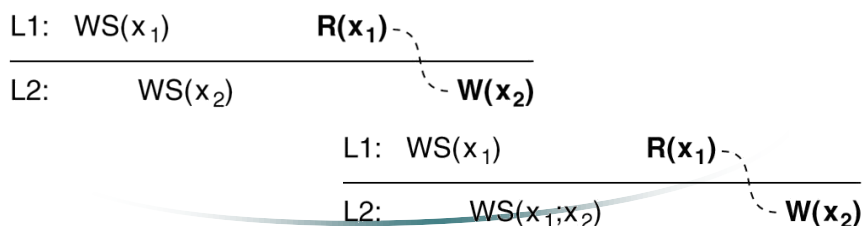
c. Read Your Writes

- Kho dữ liệu thỏa mãn các điều kiện sau đây được coi là đọc và ghi một cách nhất quán:
 - Sau khi TT thực hiện thành công thao tác ghi dữ liệu x , TT mới có thể thực hiện các thao tác đọc x tại các vị trí khác nhau.
- Ví dụ: bộ nhớ đệm của trình duyệt web: cập nhật nội dung trang web trên server rồi refresh trình duyệt trên máy → Có hay không việc đọc và ghi 1 cách nhất quán ?
- Đây là mô hình read your writes



d. Writes Follow Reads

- Kho dữ liệu thỏa mãn các điều kiện sau đây được coi là đọc và ghi một cách nhất quán:
 - Nếu tiến trình thực hiện thao tác đọc x , rồi thực hiện thao tác ghi x , thì thao tác ghi này được đảm bảo thực hiện trên cùng bản sao mà ở đó TT đã đọc giá trị x . hoặc trên bản sao mà ở đó x được đọc gần đây nhất
- Ví dụ: Sao lưu CSDL cho blog: khi đăng câu trả lời, nếu không tuân theo tính chất này thì sẽ có khả năng có người dùng đọc được câu trả lời chứ không đọc được bài viết



II. DUNG LỖI (FAULT TOLERANCE)

Lỗi (Faults)

Chương 4.
Đảm bảo chất
lượng

I. Nhất quán
và sao lưu

II. Dung lỗi

- Sai lệch so với các hành vi mong muốn
- Các nhân tố gây lỗi rất đa dạng
 - Phần cứng
 - Phần mềm
 - Người thao tác
 - Mạng



Phân loại lỗi

Chương 4.
Đảm bảo chất
lượng

I. Nhất quán
và sao lưu

II. Dung lỗi

- Theo tần suất xuất hiện:
 - transient faults: xuất hiện 1 lần rồi thôi
 - intermittent faults: xuất hiện rồi biến mất, rồi lại xuất hiện
 - permanent faults: xuất hiện và tồn tại cho đến khi thành phần gây lỗi được sửa chữa hay thay thế.
- Theo kết quả đầu ra
 - Fail-silent (fail-stop): thành phần gây lỗi ngừng hoạt động, không trả về kết quả nào hoặc kết quả trả về chỉ ra lỗi.
 - Byzantine: thành phần gây lỗi tiếp tục hoạt động, nhưng tạo ra kết quả sai



Dung lỗi

Chương 4.
Đảm bảo chất
lượng

I. Nhất quán
và sao lưu

II. Dung lỗi

- Dung lỗi trong hệ đồng bộ khác với dung lỗi trong hệ không đồng bộ
- Áp dụng tính dư thừa (redundancy) để dung lỗi
 - Dư thừa thông tin (information redundancy)
 - Dư thừa thời gian (time redundancy)
 - Dư thừa vật lý (physical redundancy)



Dung được bao nhiêu lỗi?

Chương 4.
Đảm bảo chất
lượng

I. Nhất quán
và sao lưu

II. Dung lỗi

- Không thể dung được 100 % các lỗi
 - Càng dung nhiều lỗi, chi phí càng cao
- 1 hệ thống được gọi là k-lỗi nếu nó có thể chịu đựng k lỗi
- Mức dư thừa cần thiết để chịu đựng k lỗi ?
 - Silent fault:



Dung được bao nhiêu lỗi?

Chương 4.
Đảm bảo chất
lượng

I. Nhất quán
và sao lưu

II. Dung lỗi

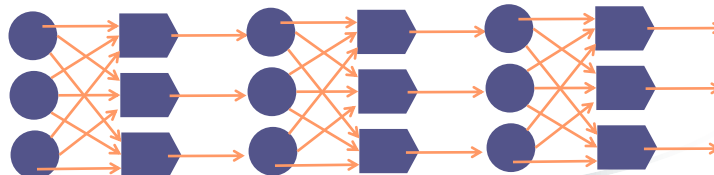
- Không thể dung được 100 % các lỗi
 - Càng dung nhiều lỗi, chi phí càng cao
- 1 hệ thống được gọi là k-lỗi nếu nó có thể chịu đựng k lỗi
- Mức dư thừa cần thiết để chịu đựng k lỗi ?
 - Byzantine fault:

Sao lưu chủ động (active replication) để dung lỗi

- Kỹ thuật dựa trên tính dư thừa vật lý
- Không dư thừa



- Triple Modular Redundancy (TMR):
 - Tạo 3 bản sao của các thành phần (threefold component replication) để phát hiện và sửa chữa lỗi của từng thành phần



Primary backup

Chương 4.
Đảm bảo chất
lượng

I. Nhất quán
và sao lưu

II. Dung lỗi

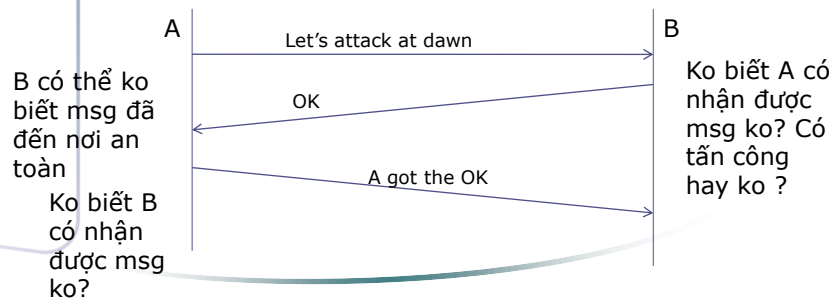
- Primary server: server chính, làm tất cả mọi việc
- Backup server, khi server chính lỗi, server dự phòng sẽ tiếp quản công việc
 - Gửi các thông điệp "are you alive" đến server chính để biết trạng thái hoạt động của server chính
- Ưu điểm:
- Nhược điểm



Thỏa thuận trong các hệ thống mắc lỗi (faulty systems)

Chương 4.
Đảm bảo chất
lượng
I. Nhất quán
và sao lưu
II. Dung lỗi

- Bài toán two army: 2 cánh của 1 đội quân cùng hiệp đồng tác chiến
 - Bộ xử lý tốt – đường truyền lỗi
 - Vấn đề: multiple acknowledgement



Thỏa thuận trong các hệ thống mắc lỗi: Bài toán: Byzantine Generals

- Đường truyền tin cậy, bộ xử lý lỗi
- n tướng cầm các cánh quân khác nhau
- m tướng phản bội, cố gắng ngăn cản các tướng khác làm theo thỏa thuận
 - 4 tướng muốn tấn công
 - 4 tướng muốn rút quân
 - 1 tướng phản bội nói với nhóm thứ nhất là muốn tấn công, nói với nhóm thứ 2 là muốn rút quân
- Các tướng trung thành làm thế nào để đạt được thỏa thuận?



Bài toán: Byzantine Generals

Chương 4.
Đảm bảo chất
lượng

I. Nhất quán
và sao lưu

II. Dung lỗi



III. AN TOÀN AN NINH (SECURITY)



Yêu cầu đảm bảo an toàn an ninh

III. An toàn an ninh

1. Mở đầu

- Bí mật (confidentiality)
 - Ngăn chặn việc rò rỉ thông tin cho những thực thể không được phép
- Toàn vẹn (integrity)
 - Chống lại việc thay đổi hay hủy hoại tài sản một cách lén lút hay trái phép
- Không thể chối bỏ (non-repudiation)
 - Ngăn chặn các thực thể chối bỏ vai trò của mình trong các hành động hay trong quá trình truyền thông
- Sẵn dùng (availability)
 - Bảo vệ hệ thống khỏi các tấn công từ chối dịch vụ



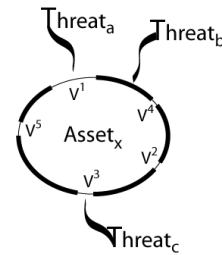
Phải trả giá đắt để đảm bảo an toàn

III. An toàn an ninh

1. Mở đầu

Đảm bảo an toàn: Phải quản lý được rủi ro

- Kiến trúc an toàn (security architecture):
 - Tập các kỹ thuật và chính sách áp dụng vào một hệ thống với mục đích giảm thiểu đe dọa và rủi ro
- Đe dọa (threat):
- Tấn công (attack):
- Điểm yếu (vulnerability):



Đảm bảo an toàn: tổn kém và lợi ích

Quyết định bảo vệ những thành phần nào của hệ thống

- | | |
|---|--|
| <ul style="list-style-type: none"> • Lợi ích: <ul style="list-style-type: none"> – Lường trước số lần bị đe dọa tấn công và các thiệt hại trong trường hợp bị tấn công – Nhận biết và đối phó với các cơ hội tấn công | <ul style="list-style-type: none"> • Chi phí <ul style="list-style-type: none"> – Phát triển và bảo trì <ul style="list-style-type: none"> • Phần mềm phức tạp – Cơ sở hạ tầng <ul style="list-style-type: none"> • Bản quyền sản phẩm • Các máy chuyên dụng hay lớn hơn – Giảm hiệu năng <ul style="list-style-type: none"> • Mã hóa, xác thực, phân quyền... |
|---|--|

→ Quan điểm của kẻ tấn công

- | | |
|--|--|
| <ul style="list-style-type: none"> • Lợi ích của việc tấn công thành công | <ul style="list-style-type: none"> • Chi phí phát động tấn công • Các án phạt (dân sự) |
|--|--|

Lựa chọn các giải pháp đảm bảo an toàn dựa trên việc phân tích chi phí và lợi ích

deciding how much to protect

• Lợi ích:

- Lường trước số lần bị đe dọa tấn công và các thiệt hại trong trường hợp bị tấn công
- Nhận biết và đối phó với các cơ hội tấn công

ex: losses \$200k
chances 50%
benefit \$100k

Kỹ thuật 1

- Chi phí
- Hiệu quả

ex: cost \$30k, 60% effective

?

Kỹ thuật 2

ex: cost \$50k, 90% effective

Kỹ thuật 3

ex: cost \$5k, 50% effective

Có thể lường trước mọi đe dọa hay không?

Tổ chức các giải pháp đảm bảo an toàn theo công việc

III. An toàn an ninh
1. Mở đầu

- Bảo vệ
 - Kênh thông tin an toàn (secure channels)
 - Xác thực (authentication): Các bên tham gia vào quá trình truyền thông đúng là các thành phần đã định
 - Bí mật (confidentiality) và toàn vẹn (integrity): Các bên thứ ba không thể can thiệp vào quá trình truyền thông
 - Kiểm soát truy nhập (access control)
 - Phân quyền (authorization): Thực thể nào có thể truy nhập các tài sản nào, thực hiện các thao tác nào
 - Quy trách nhiệm (accountability): Luôn biết ai đã làm gì
 - Chống chối bỏ (non-repudiation): Các thực thể không thể chối bỏ những việc đã làm
- Phát hiện
- Phục hồi



Nhiệm vụ

III. An toàn an ninh

1. Mở đầu

2. Mã hóa

- Đảm bảo kênh truyền thông an toàn (secure channel)
 - Bí mật
 - Toàn vẹn
 - Xác thực



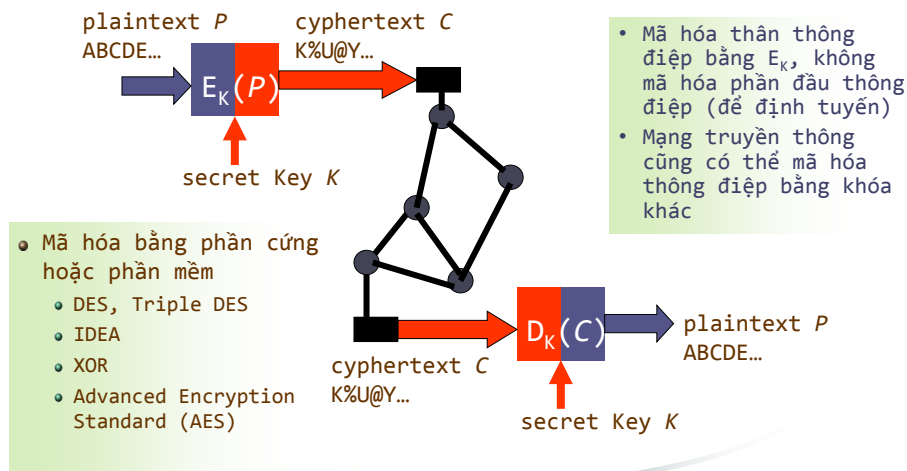
Công cụ mã hóa

- Mã hóa đối xứng (symmetric encryption): chia sẻ khóa
 - $M = D_K(E_K(M))$
- Mã hóa bất đối xứng (asymmetric encryption): khóa bí mật/khóa công khai
 - $M = D_K^-(E_K^+(M)) = D_K^+(E_K^-(M))$
- Băm (hashing) và mã xác thực thông điệp (MAC)
 - $S = H(M)$, or $S = \text{MAC}_K(M)$:
 - Kích thước của S cố định và không phụ thuộc vào M

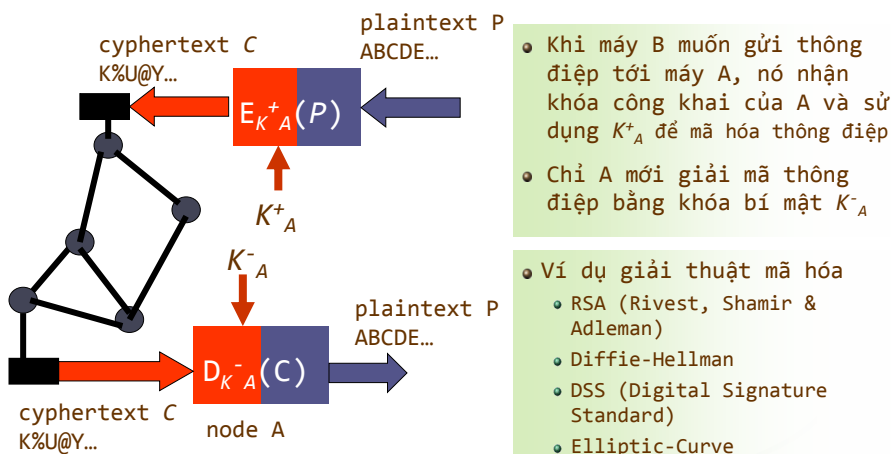
Các đặc tính của E/D/H

- one-way: cho μ , không biết E_K , không thể tính được m sao cho $E_K(m) = \mu$
- weak collision resistance: cho m , $E_K(m)$, không thể tính được $n \neq m$ sao cho $E_K(m) = E_K(n)$
- strong collision resistance: cho E_K , không thể tính được cặp m và n , $n \neq m$, sao cho $E_K(m) = E_K(n)$

- Truyền thông tin cậy sử dụng mã đối xứng:
- khóa được chia sẻ giữa các bên tin cậy, giữ bí mật với các bên khác



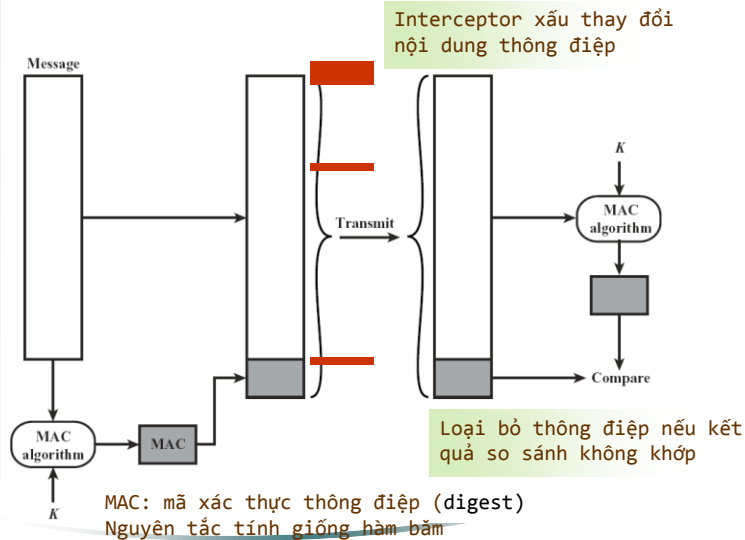
- Truyền thông tin cậy sử dụng mã bất đối xứng: thông báo khóa công khai của từng máy cho tất cả các thành viên



Dùng chung khóa bí mật

III. An toàn an ninh

1. Mở đầu
2. Mã hóa
3. **Kiểm tra tính toàn vẹn**

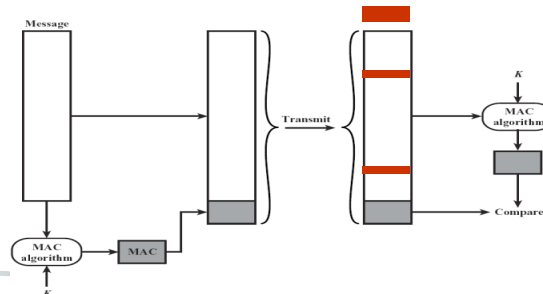


Bổ sung hoặc thay thế phần kiểm tra tính bí mật

III. An toàn an ninh

1. Mở đầu
2. Mã hóa
3. **Kiểm tra tính toàn vẹn**

- Có cần kiểm tra tính toàn vẹn khi thông điệp không bị mã hóa hay không?
 - Trong bối cảnh nào muốn gửi thông điệp không mã hóa?
 - read vs. tamper
 - performance
- Điều gì xảy ra nếu thông điệp bị mã hóa
 - Tại sao lại phải thêm 1 MAC vào 1 thông điệp đã mã hóa rồi?





Không tách biệt xác thực và kiểm tra tính toàn vẹn

- Bối cảnh:
 - Mã hóa đảm bảo rằng chỉ những thành phần tham gia truyền thông hiểu được nội dung thông điệp
 - Kiểm tra tính toàn vẹn đảm bảo bên nhận nhận được thông điệp không bị thay đổi trong khi truyền
- Điều gì xảy ra nếu bên nhận không biết chắc ai gửi thông điệp?
- Ngược lại, điều gì xảy ra nếu bên nhận biết chắc ai là người gửi thông điệp, nhưng không chắc là thông điệp đó có bị sửa đổi hay không?

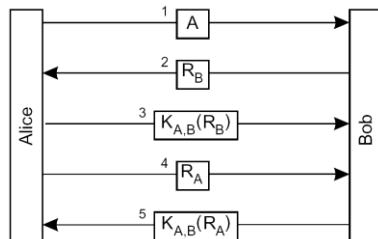


a. Xác thực dựa trên khóa bí mật chung

- III. An toàn an ninh
1. Mở đầu
 2. Mã hóa
 3. Kiểm tra tính toàn vẹn
 4. **Xác thực**

- Giả sử Alice và Bob chia sẻ một khóa bí mật.
- Làm thế nào họ có thể thiết lập một kênh truyền thông an toàn trong một môi trường truyền thông không an toàn?

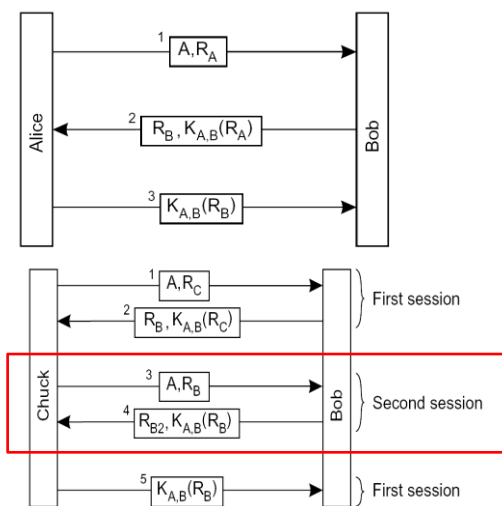
a. Xác thực dựa trên khóa bí mật chung



1. Alice gửi ID cho Bob.
2. Bob gửi câu đố (random number?).
3. Alice phải mã hóa câu đố và gửi lại.
4. Alice gửi câu đố tới Bob.
5. Bob phải mã hóa câu đố và gửi lại.

Thiết kế giao thức xác thực hoạt động tốt là công việc khó khăn

1. Chuck gửi ID của Chuck và câu đố của Chuck cho Bob.
2. Bob phải mã hóa câu đố của Chuck và gửi lại cùng với câu đố của Bob.
3. **Replay attack: Chuck lừa Bob tiết lộ kết quả mã hóa câu đố của Alice: Chuck thiết lập kênh truyền thứ 2, gửi ID của Alice và câu đố của Bob cho Bob.**
4. Bob phải mã hóa câu đố của Bob và gửi lại.
5. Chuck dùng chính kết quả này gửi lại cho Bob (trên kênh truyền thứ nhất) để chứng minh rằng Chuck biết $K_{A,B}$

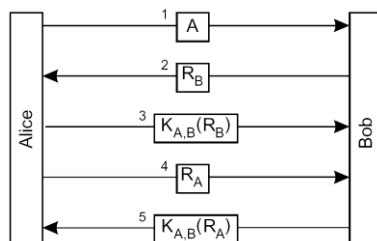


a. Xác thực dựa trên khóa bí mật chung

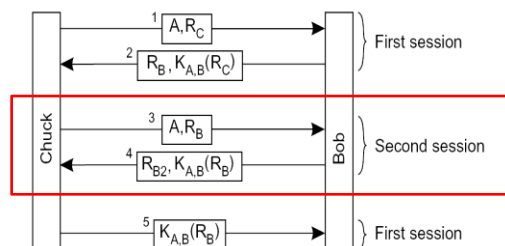
- Giao thức này có thể tối ưu không?
 - piggyback challenges and replies
1. Alice gửi ID và câu đố cho Bob.
 2. Bob phải mã hóa câu đố của A và gửi lại cùng với câu đố của Bob.
 3. Alice phải mã hóa câu đố của Bob và gửi lại.



Thiết kế giao thức xác thực hoạt động tốt là công việc khó khăn



- Tại sao giao thức 5 bước hoạt động tốt, mà giao thức 3 bước lại bị tấn công?

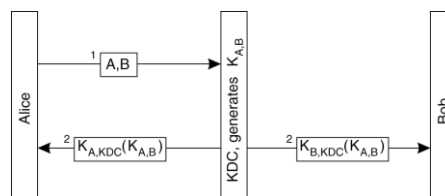


b. Xác thức 3 bên (trusted third party)

- Nếu có N bên chia sẻ khóa bí mật, vậy phải giữ tổng cộng bao nhiêu khóa?
- Giải pháp thay thế: dùng trung tâm phân phối khóa (KDC)

Key Distribution Centers

1. Alice gửi ID của Alice và Bob cho KDC.
2. KDC sinh ra khóa bí mật cho kênh truyền thông giữa Alice và Bob, mã hóa và gửi lại cho cả Alice và Bob.

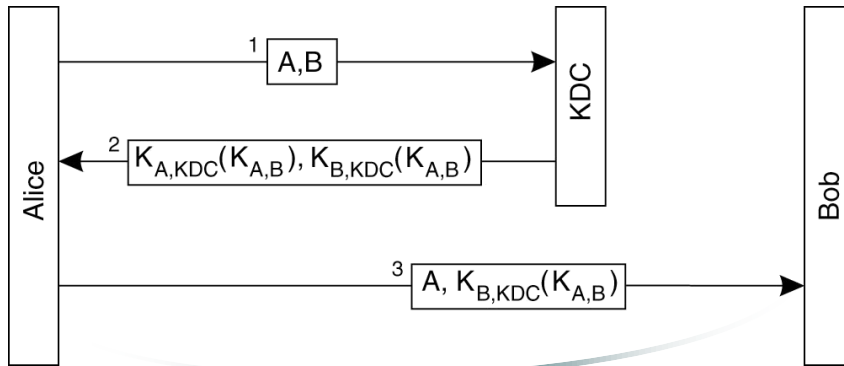


- Làm thế nào để Alice có thể bắt đầu giao tiếp với Bob trước khi KDC gửi khóa bí mật cho Bob??

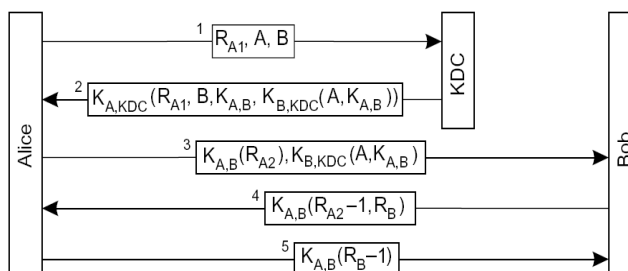


Giải pháp: Tickets

- KDC tạo ra 1 ticket $K_{B,KDC}(K_{A,B})$, gửi cho Alice và để cho Alice tự thiết lập kết nối tới Bob.
- Bob có tin được Alice hay không?



Giao thức Needham-Schroeder

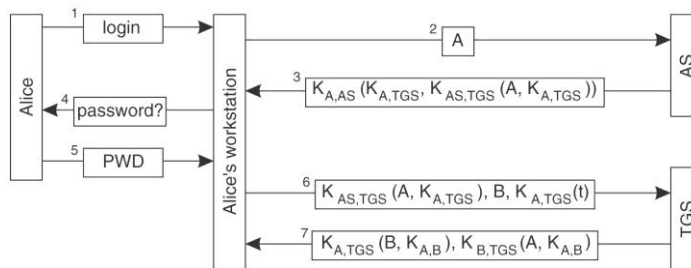


- 1 không được mã hóa – tại sao?
- Mỗi câu đố chỉ được sử dụng 1 lần, tại sao?

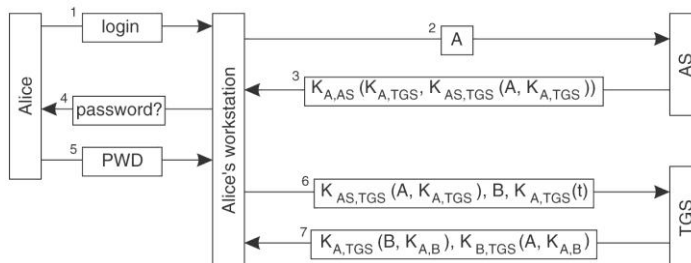
1. A gửi cho KDC câu đố R_{A1} , ID của A và của B
 2. R_{A1} được mã hóa bằng $K_{A,KDC}$ → giúp A xác thực KDC.
ID của B giúp phòng tránh tấn công kiểu man in the middle : C không thể thay đổi 1 thành R_{A1} , A, C và làm cho A tưởng rằng đang nói chuyện với B
 - 3, 4. R_{A2} và $R_{A2}-1$ xác thực B với A
 5. xác thực A với B
- Mỗi kênh truyền được KDC cấp cho 1 khóa mới: hạn chế tấn công kiểu phân tích thông điệp

Kerberos

- Sử dụng cơ chế chia sẻ khóa bí mật để hỗ trợ client thiết lập kết nối an toàn đến các server của 1 HPT
- Authentication Server (AS):
 - Xác thực người dùng
 - Cung cấp khóa để thiết lập kết nối an toàn với server
- Ticket Granting Service (TGS).
 - Cung cấp tickets để giúp server xác thực người dùng
 - Điều khiển kênh kết nối an toàn



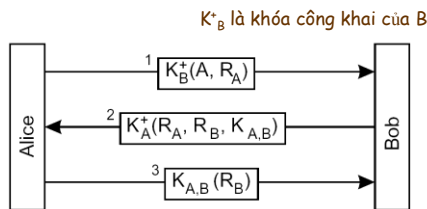
Kerberos



- Alice gõ tên truy cập vào máy
- Máy chuyển tên này đến AS
- AS trả về khóa $K_{A,TGS}$ để tạo ra kênh truy cập an toàn giữa A và TGS và 1 ticket $K_{AS,TGS}(A, K_{A,TGS})$ để TGS xác thực Alice. Các thông tin này được mã hóa bằng khóa $K_{A,AS}$ (sinh ra bằng cách bấm pwd của Alice)
- t là 1 timestamp được mã hóa nhằm hạn chế các tấn công kiểu replay
- Tương ứng với thông điệp 2 trong giao thức Needham-Schroeder



c. Xác thực với mã bất đối xứng



- Chỉ có B mới giải mã được R_A bằng khóa bí mật của B
- Chỉ có A mới giải mã được R_B bằng khóa bí mật của A
- 2: R_A giúp A xác thực B
- 3: R_B được mã hóa bằng khóa kênh truyền giữa A và B giúp B xác thực A

• Giả thiết:

- Mỗi bên đều biết khóa công khai của bên kia
- Không thể phá khóa công khai

Câu hỏi:

Làm thế nào để lấy được khóa công khai một cách an toàn?



Mở đầu

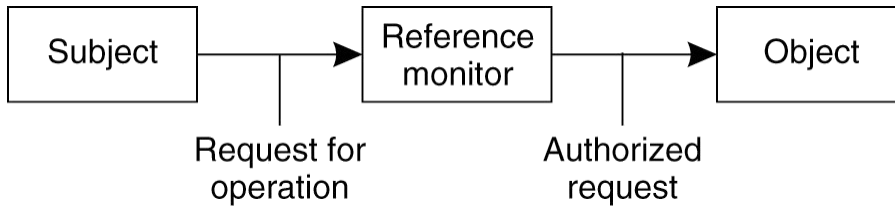
III. An toàn an ninh

1. Mở đầu
2. Mã hóa
3. Kiểm tra tính toàn vẹn
4. Xác thực
5. **Kiểm soát truy cập**

- Cho 1 HPT gồm 1 server và 1 tập các đối tượng nằm dưới quyền kiểm soát của server.
- Tất cả các đối tượng đều có thể gửi yêu cầu, tuy nhiên server chỉ xử lý yêu cầu của các đối tượng có đủ quyền
- Authorization: làm thế nào để cấp quyền
- Access control: làm thế nào để kiểm tra quyền



a. Mô hình chung



b. Access Control Matrix

- Sử dụng ma trận để mô hình các quyền: $M[s,o]$ chứa các phương thức được phép truy nhập

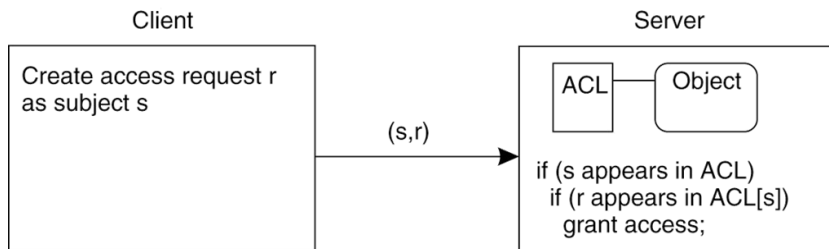
		Objects	
		O1	O2
Subjects	S1	m1, m3	-
	S2	m2	m3

- Nhược điểm: kích thước ma trận lớn, nhiều phần tử rỗng



c. Access Control Lists

- Sử dụng danh sách, cấp quyền (grant rights) hoặc hủy quyền (remove rights).
 - Bob, read
 - Alice, write
 - Chuck, check timestamp



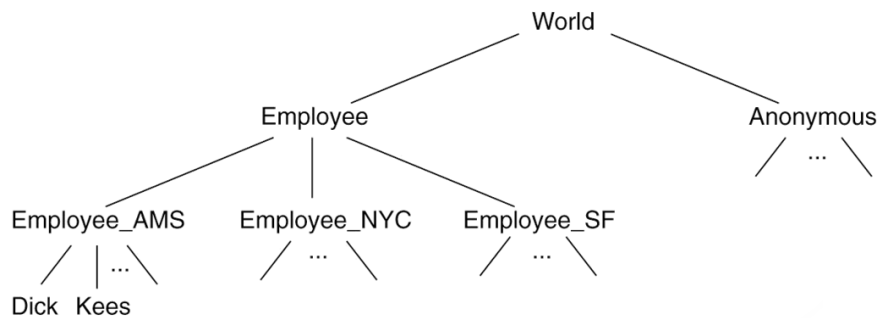
Các vùng được bảo vệ (Protection Domains)

- Giả sử server phải quản lý 10000 user. Kích thước của ACL cho 1 dịch vụ của server là bao nhiêu, biết rằng mỗi dịch vụ này cung cấp khoảng 100 phép toán ?
 - Việc đặc tả những phép toán được phép thực hiện cho từng user rất tốn kém
 - Giải pháp?



Giải pháp: Groups

- Khi 1 user yêu cầu thực hiện một phép toán, server kiểm tra xem user đó thuộc nhóm nào
- Các nhóm có thể phân cấp.
- Một người dùng có thể thuộc nhiều nhóm hay không?



Làm thế nào server biết người dùng thuộc nhóm nào?



Giải pháp: Roles

- Users có các vai trò khác nhau
- Role khác group như thế nào?



Bài tập

- A muốn gửi 1 thông điệp có kích thước 500 Kb đến B, sao cho:
 - Chỉ B đọc được thông điệp
 - Xác thực đúng là thông điệp do A gửi
 - Có thể mã hóa/giải mã trong vòng 15 giây
- Giả sử:
 - Giải thuật khóa công khai có thể mã hóa/giải mã thông điệp với tốc độ 1Kbps
 - Giải thuật khóa đối xứng có thể mã hóa/giải mã thông điệp với tốc độ 1Mbps
 - Ban đầu không chia sẻ khóa đối xứng
 - Giải thuật băm 256 bit có thể xử lý các thông điệp với tốc độ 100Kbps
 - Kích thước của ID của A và khóa đối xứng là 256 bit
- Làm thế nào để định dạng thông điệp A?