CSC474/574 Information Systems Security
Homework 3 Solutions Sketch

1. (20 points) Consider a computer system with three users: Alice, Bob, and Cyndy. Alice owns the file alicerc, and Bob and Cyndy can read it. Cyndy can read and write Bob's file Bobrc, but Alice can only read it. Only Cyndy can read and write her file cyndyrc. Assume the owner of each of these files can execute it.

a. **(5 points) Create the corresponding access control matrix.**
b. **(5 points) Use ACL to represent the access control policy specified in (a).**
c. **(5 points) Use Capabilities to represent the access control policy specified in (a).**
d. **(5 points) Cyndy gives Alice permission to read cyndyrc, and Alice removes Bob's ability to read alice rc. Show the new access control matrix.**

A)
a.

	alicerc	bobrc	cyndyrc
Alice	{O,X}	{R}	---
Bob	{R}	{O, X}	---
Cyndy	{R}	{R, W}	{R, W, X, O}

b.

Acl(alicerc) = {(Alice, O), (Alice, X), (Bob, R), (Cyndy, R)}

Acl(bobrc) = {(Alice, R), (Bob, O), (Bob, X), (Cyndy, R), (Cyndy, W)}

Acl(cyndyrc) = {(Cyndy, R), (Cyndy, W), (Cyndy, X), (Cyndy, O)}

c.

Capability(Alice) = {(alicerc, {O}), (alicerc, {X}), (bobrc, {R}) }

Capability(Bob) = {(alicerc, {R}), (bobrc, {O}), (bobrc, {X})}

Capability(Cyndy) = {(alicerc, {R}), (bobrc, {R}), (bobrc, {W}), (cyndyrc, R), (cyndyrc, W), (cyndyrc, O), (cyndyrc, X)}

d.

	alicerc	bobrc	cyndyrc
Alice	{O,X}	{R}	{R}
Bob	---	{O, X}	---
Cyndy	{R}	{R, W}	{R, W, X, O}

2. (30 points) Consider two categories A and B. It is well known that MAC with four compartments can be constructed from these two categories.

a. **(10 points) Customize the RBAC0 model to implement the security policy represented by the above compartments. Assume there is no user in the system. Represent permissions as o(c), where o is either 'r' (read) or 'w' (write) and c is a security class. For example, a permission r({AB}) refers to the permission to read information from the class {AB}. You need to specify the components of the RBAC0 model. Note that you may have some empty components (e.g., users, sessions).**
b. **(10 points) Assume you add the first user X, who is cleared to have the security class {A}. What changes would you make to the result of (a)?**
c. **(10 points) Assume that you are allowed to use RBAC1. Develop a role hierarchy and the corresponding permission assignments. Note that you may not have a clean hierarchy, since the set of permissions of one role is often not a subset of that of another role.**

A)

a. RBAC0 Model:

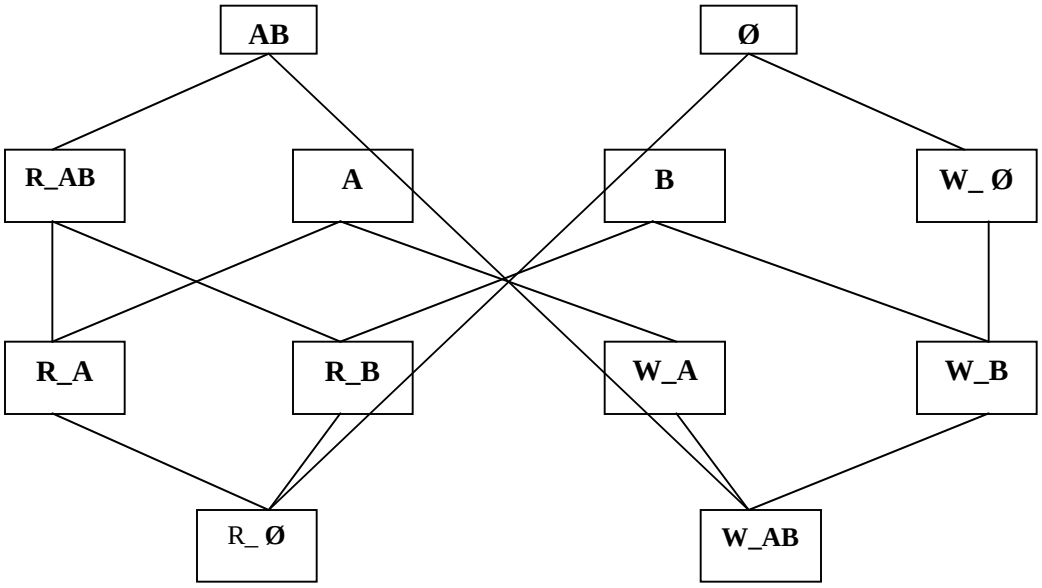
Users = { }
Roles = { Ø, A, B, AB }
Permissions= { r(Ø), w(Ø), r(A), w(A),r(B), w(B),r(AB), w(AB) }
Permission Assignments = { (Ø, r(Ø)), (Ø, w(Ø)), (Ø, w(A)), (Ø, w(B)), (Ø, w(AB)), (A, r(Ø)), (A, r(A)), (A, w(A)), (A, w(AB)), (B, r(Ø)), (B, r(B)), (B, w(B)), (B, w(AB)), (AB, r(Ø)), (AB, r(A)), (AB, r(B)), (A, r(AB)), (AB, w(AB)) }
User Assignments = { }
Sessions = { }
Users(S → U) = { }
Roles(S → 2^R) = { }

b.

Users = {X}
Roles = { Ø, A, B, AB }
Permissions= { r(Ø), w(Ø), r(A), w(A),r(B), w(B),r(AB), w(AB) }
Permission Assignments = { (Ø, r(Ø)), (Ø, w(Ø)), (Ø, w(A)), (Ø, w(B)), (Ø, w(AB)), (A, r(Ø)), (A, r(A)), (A, w(A)), (A, w(AB)), (B, r(Ø)), (B, r(B)), (B, w(B)), (B, w(AB)), (AB, r(Ø)), (AB, r(A)), (AB, r(B)), (A, r(AB)), (AB, w(AB)) }
Sessions = {S1}
Users(S → U) = {S1 → X}
Roles(S → 2_R) = {S1 → {A}}

c. Role Hierarchy:

Users = { }
Roles = { Ø, A, B, AB, R_ Ø, W_ Ø , R_A, W_A, R_B, W_B, R_AB, W_AB }
Permissions= { r(Ø), w(Ø), r(A), w(A),r(B), w(B),r(AB), w(AB) }
Permission Assignments = { (R_ Ø , r(Ø)), (W_Ø, w(Ø)), (R_A, r(A)), (W_A, w(A)), (R_B, r(B)), (W_B, w(B)), (R_AB, r(AB)), (W_AB, w(AB)) }
User Assignments = { }
Sessions = { }



3. (40 points) A company has the following security policies regarding the information that the company considers confidential:

- **The company divides information into the following classes: CEO (C), Managers (M), and public (P);**
- **Information from Managers and public classes can flow to the CEO class;**
- **Information from public class can flow to the Manager class;**
- **Information can flow within each class;**
- **There is no other allowed information flow between classes.**

    a. **(10 points) Specify this security policy using Denning's formalism. Assume the security classes are C (CEO), M (Manager), P (Public).**
    b. **(10 points) Draw the security policy in (a) in a lattice (in Hasse diagram). Assume that information can flow upwards.**
    c. **(10 points) Assume that in the Manager class, the information is further divided into compartments with categories A, B, and C. Draw the resulting lattice for the entire security policy.**
    d. **(10 points) Continue from (b). Assume the company further divides information into high integrity (H) and low integrity (L) classes. Information can only flow from high integrity class to low integrity class. Combine the policies for both confidential and integrity, and draw the resulting policy in a lattice.**

A)

a.

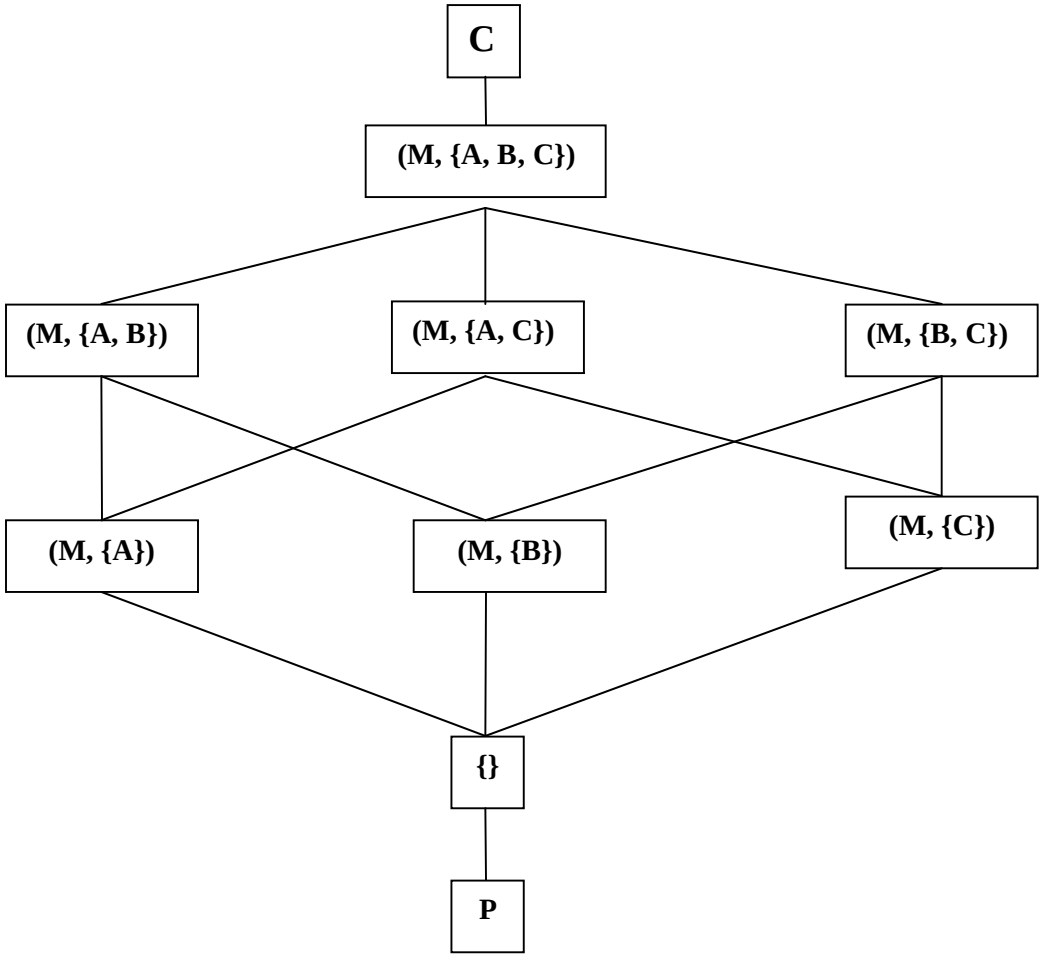
SC = { C, M, P}

→ = {C_C, M_M, P_P, P_M, M_C }

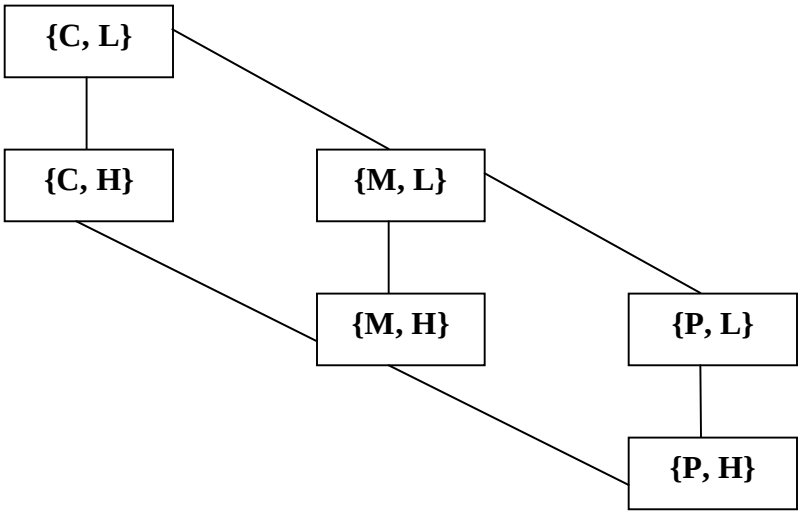
+ = { P+P → P,P+M → M,P+C → C,M+M → M,M+C → C,C+C → C }

b.

c.



d.



4) (5 points) Give an example for storage covert channel

A) Covert Channel uses shared resources as paths of connection. This requires sharing of space or time.

Storage Covert Channel:
Suppose 2 processes P & Q are not supposed to communicate with each other and P & Q share a file system. For process P to send a message to process Q, it creates a file called send in a directory that both processes can read. Before Q is ready to read the information, it deletes the send file. Process P then transmits a bit by creating a file named "0bit" or "1bit". Q records the bit and then deletes the file. This continues until P creates a file called end.

Here the shared resource is the directory and names of files in that directory. Processes communicate by altering characteristics (filenames and file extensions) of shared resource.

5) (5 points) Give an example for timing covert channel

Timing Covert Channel:
Two virtual machines can establish a covert channel based on the CPU quantum that each machine receives. If sending machine wishes to send a "0" bit, it relinquishes the CPU immediately; to send a "1", it uses its full quantum. By determining how quickly it got the CPU, second virtual machine can deduce whether the first was sending "1" or a "0". The shared resource is CPU. The processes communicate using a real-time clock to measure the intervals between accesses to shared resource.