IS 2150 / TEL 2810
Introduction to Security

James Joshi
Assistant Professor, SIS

Lecture 7
October 11, 2007

Role based
Access Control

Take Grant Model

1

Objective

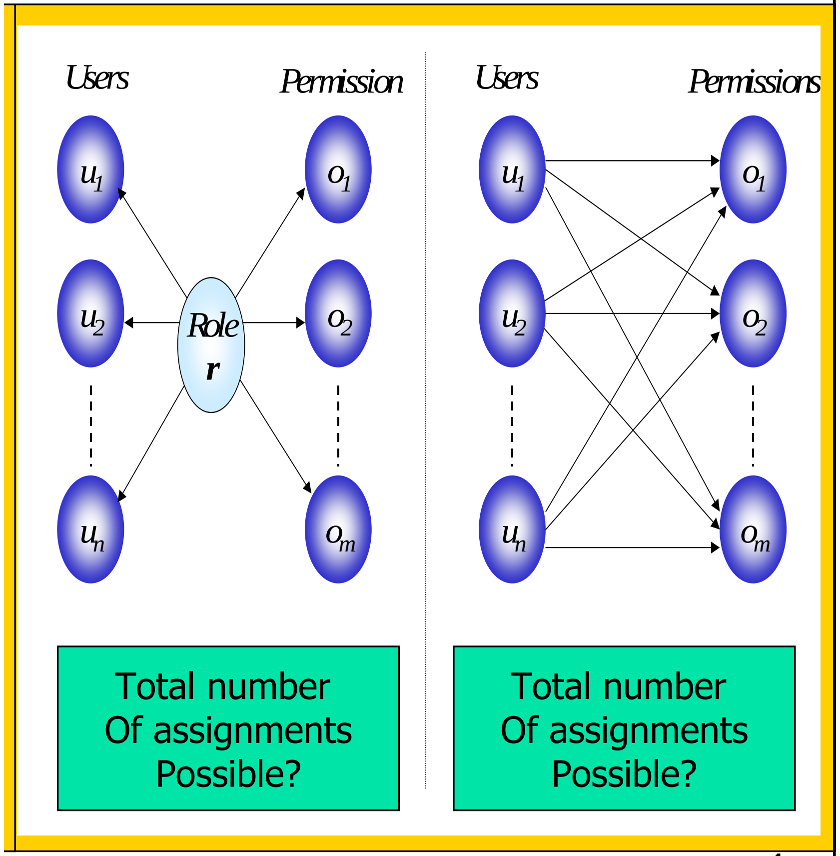
- Define/understand/represent formally
  - Take grant model
  - Role-based Access Control model
- Analyze/deduce (in TG or RBAC models)
  - stealing of permissions
  - Conspiracy
  - Static/Dynamic separation of duty
- Understand key issue related to secure interoperation

Role Based Access Control (RBAC)

- Access control in organizations is based on "roles that individual users take on as part of the organization"
  - Access depends on function, not identity
    - Example:

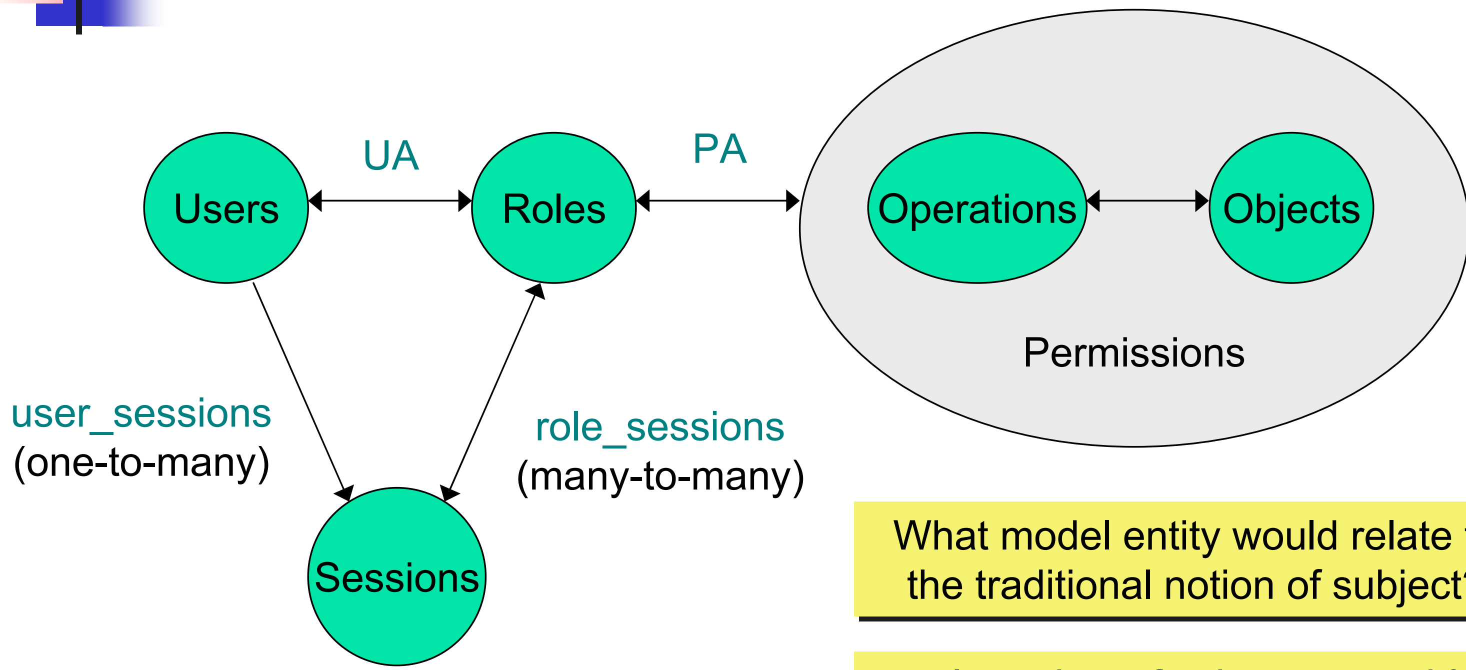
      Allison is bookkeeper for Math Dept. She has access to financial records. If she leaves and Betty is hired as the new bookkeeper, Betty now has access to those records. The role of "bookkeeper" dictates access, not the identity of the individual.

RBAC



Users
Permission
u1
u2
un
o1
o2
om
Role
r
Total number
Of assignments
Possible?
Users
Permissions
u1
u2
un
o1
o2
om
Total number
Of assignments
Possible?

RBAC (NIST Standard)



What model entity would relate to the traditional notion of subject?

Total number of subjects possible?

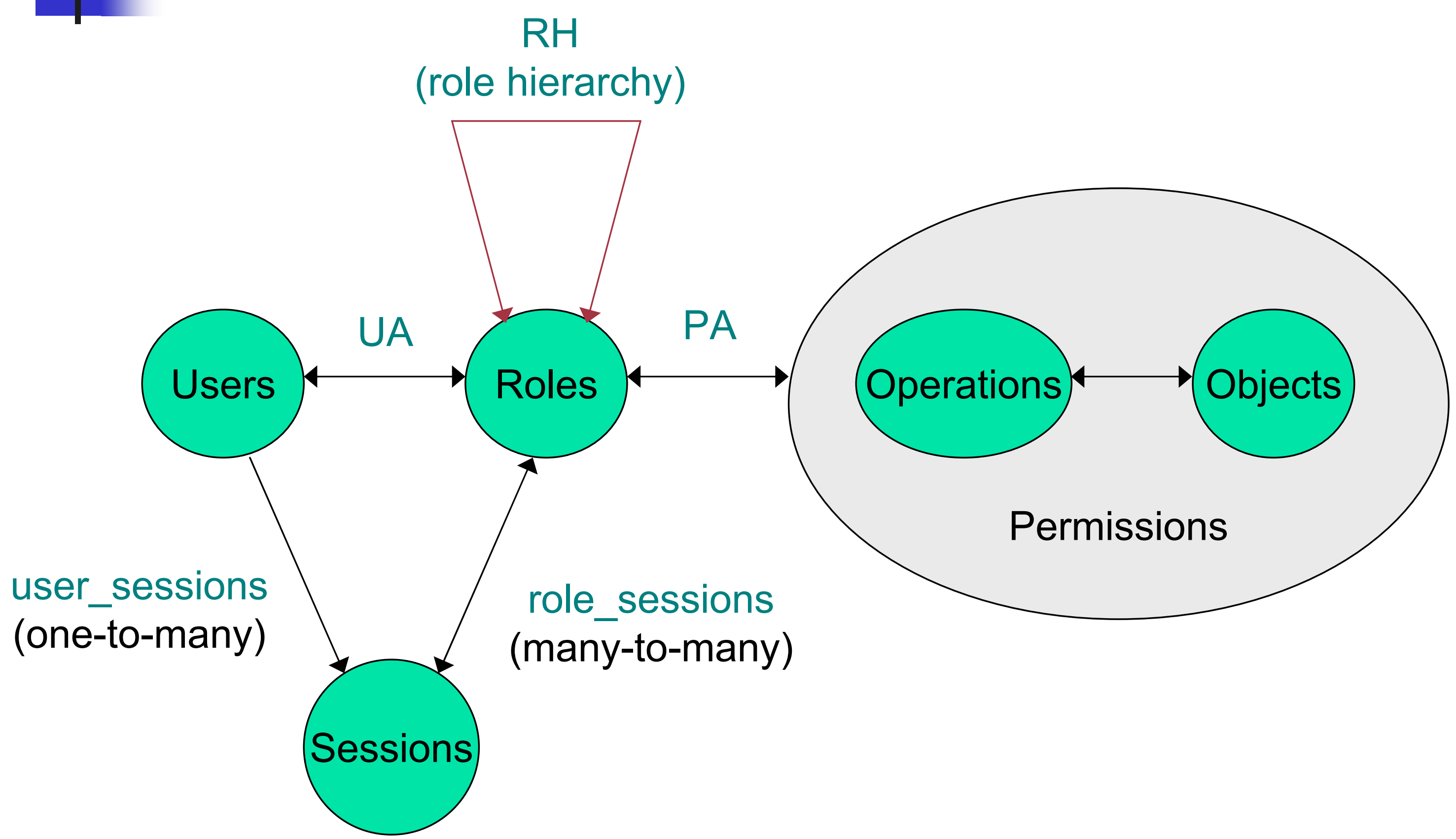
Role vs Group?

Core RBAC (relations)

- Permissions = $2^{\text{Operations x Objects}}$
- UA $\subseteq$ Users x Roles
- PA $\subseteq$ Permissions x Roles
- *assigned_users*: Roles $\rightarrow 2^{\text{Users}}$
- *assigned_permissions*: Roles $\rightarrow 2^{\text{Permissions}}$
- *Op*(p): set of operations associated with permission p
- *Ob*(p): set of objects associated with permission p
- *user_sessions*: Users $\rightarrow 2^{\text{Sessions}}$
- *session_user*: Sessions $\rightarrow$ Users
- *session_roles*: Sessions $\rightarrow 2^{\text{Roles}}$
  session_roles(*s*) =
  $\{r \mid (\text{session\_user}(s), r) \in \text{UA})\}$
- *avail_session_perms*: Sessions $\rightarrow 2^{\text{Permissions}}$

6

RBAC with Role Hierarchy



7

RBAC with General Role Hierarchy

- *authorized_users*: Roles $\rightarrow 2^{\text{Users}}$

  $authorized\_users(r) = \{u \mid r' \geq r \;\&\; (r', u) \in UA\}$

- *authorized_permissions*: Roles $\rightarrow 2^{\text{Permissions}}$

  $authorized\_permissions(\mathrm{r}) = \{p \mid r \geq r' \;\&\; (p, r') \in PA\}$

- RH $\subseteq$ Roles x Roles is a partial order
  - called the inheritance relation
  - written as $\geq$.

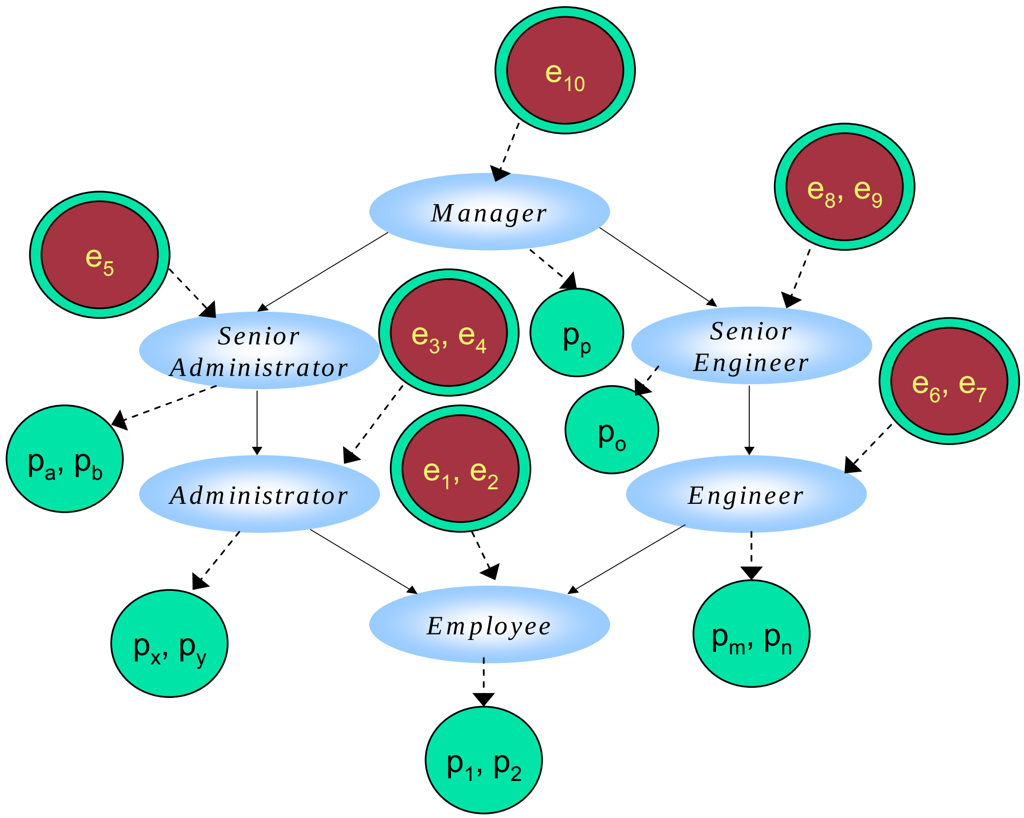
  $(r_1 \geq r_2) \rightarrow authorized\_users(r_1) \subseteq authorized\_users(r_2)$ & $authorized\_permisssions(r_2) \subseteq authorized\_permisssions(r_1)$

What do these mean?

8

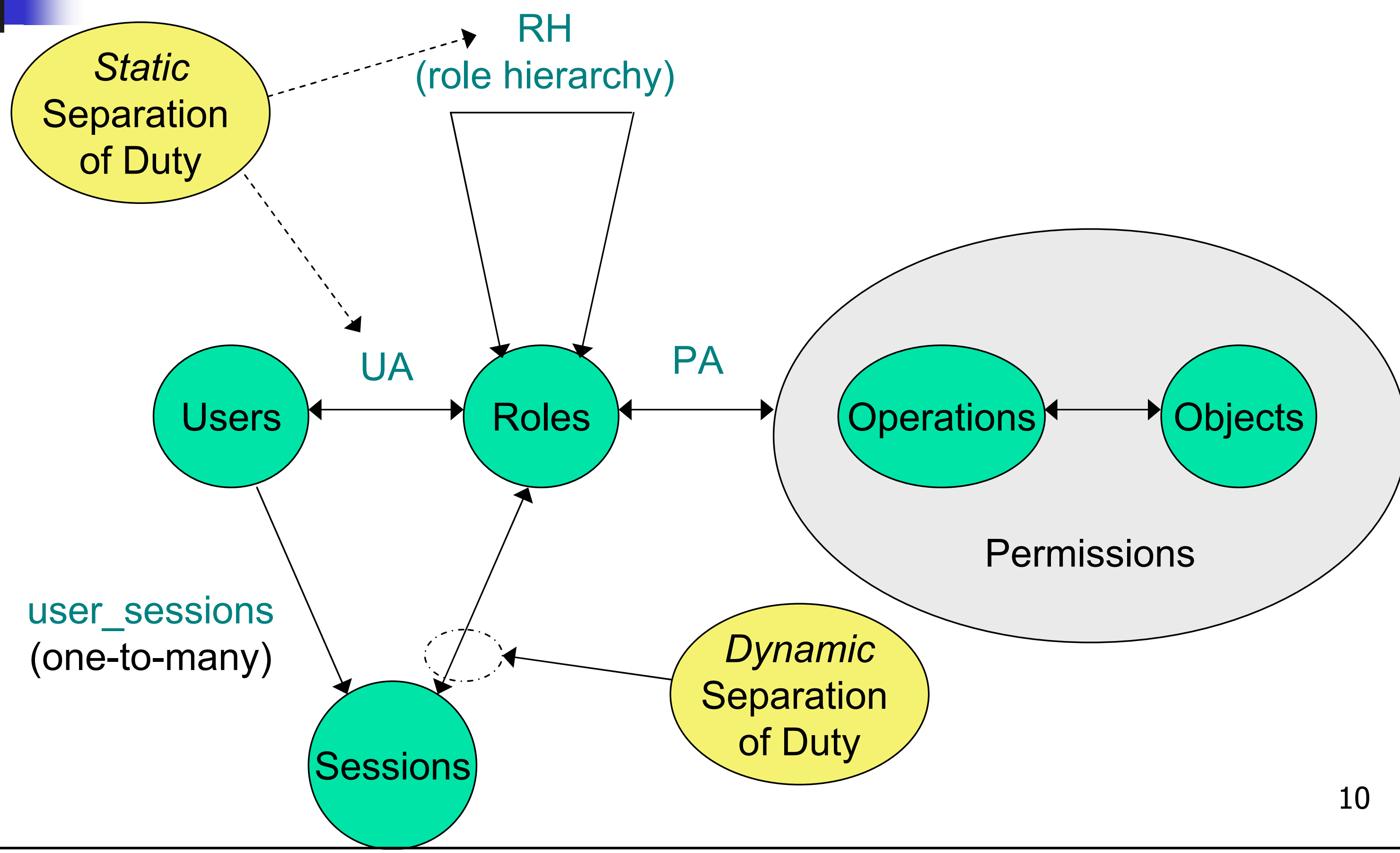
Example

authorized_users(Employee)?
authorized_users(Administrator)?
authorized_permissions(Employee)?
authorized_permissions(Administrator)?



9

Constrained RBAC



10

Static Separation of Duty

- *SSD* $\subseteq 2^{\text{Roles}} \times N$
- In absence of hierarchy
  - Collection of pairs (*RS*, *n*) where *RS* is a role set, $n \geq 2$

    for all $(RS, n) \in SSD$, *for all* $t \subseteq RS$:

    $|t| \geq n \rightarrow \bigcap_{r \in t} assigned\_users(\mathrm{r}) = \varnothing$

Describe!

- In presence of hierarchy
  - Collection of pairs (RS, n) where RS is a role set, n ≥ 2;

    for all $(RS, n) \in SSD$, *for all* $t \subseteq RS$:

    $|t| \geq n \rightarrow \bigcap_{r \in t} authorized\_uers(r) = \varnothing$

Describe!

Dynamic Separation of Duty

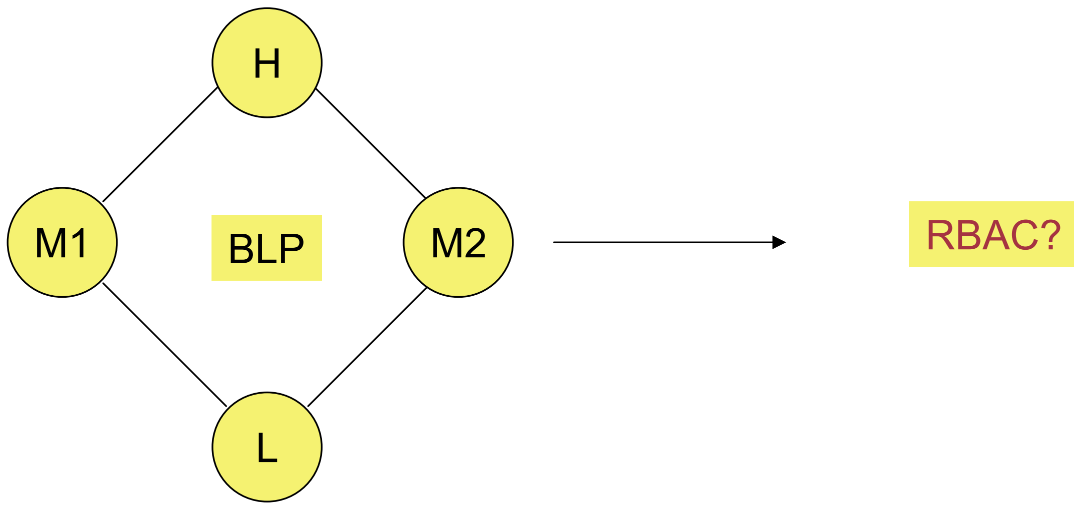
- $DSD \subseteq 2^{Roles} \times N$
  - Collection of pairs (RS, n) where RS is a role set, $n \geq 2$;
    - A user cannot activate n or more roles from RS
  - What is the difference between SSD or DSD containing:

    (RS, n)?

    - Consider (RS, n) = ({r_1, r_2, r_3}, 2)?
    - If SSD – can r_1, r_2 *and* r_3 be assigned to u?
    - If DSD – can r_1, r_2 *and* r_3 be assigned to u?

12

Can we represent BLP using RBAC?



13

Advantages of RBAC

- Allows Efficient Security Management
  - Administrative roles, Role hierarchy
- Principle of least privilege allows minimizing damage
- Separation of Duty constraints to prevent fraud
- Allows grouping of objects / users
- Policy-neutral - Provides generality
- Encompasses DAC and MAC policies

RBAC's Benefits

TABLE 1: ESTIMATED TIME (IN MINUTES) REQUIRED FOR ACCESS ADMINISTRATIVE TASKS			
TASK	RBAC	NON-RBAC	DIFFERENCE
Assign existing privileges to new users	6.14	11.39	5.25
Change existing users' privileges	9.29	10.24	0.95
Establish new privileges for existing users	8.86	9.26	0.40
Termination of privileges	0.81	1.32	0.51

Cost Benefits

- Saves about 7.01 minutes per employee, per year in administrative functions
  - Average IT admin salary - $59.27 per hour
  - The annual cost saving is:
    - $6,924/1000;
    - $692,471/100,000

How do we get this?

Take Grant Model

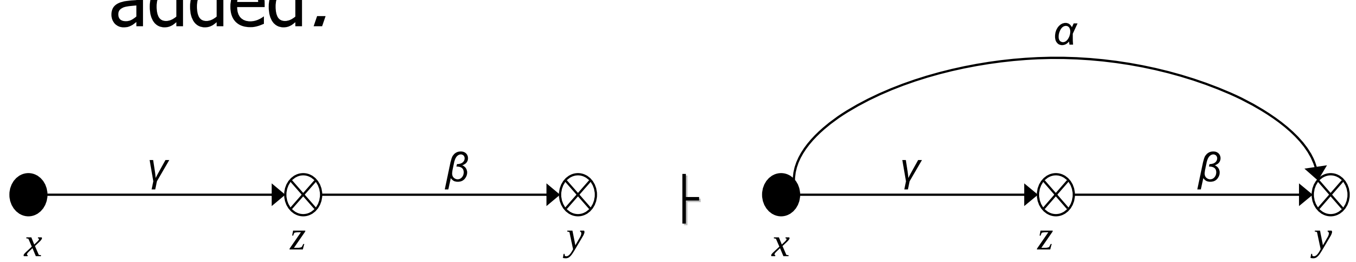
17

Take-Grant Protection Model

- System is represented as a directed graph
  - Subject: ●
  - Object: ○ Either: ⊗
  - Labeled edge indicates the rights that the source object has on the destination object
- Four graph rewriting rules ("de jure", "by law", "by rights")
  - The graph changes as the protection state changes according to these rules

Take rule

- if $t \in \gamma$, the take rule produces another graph with a transitive edge $\alpha \subseteq \beta$ added.

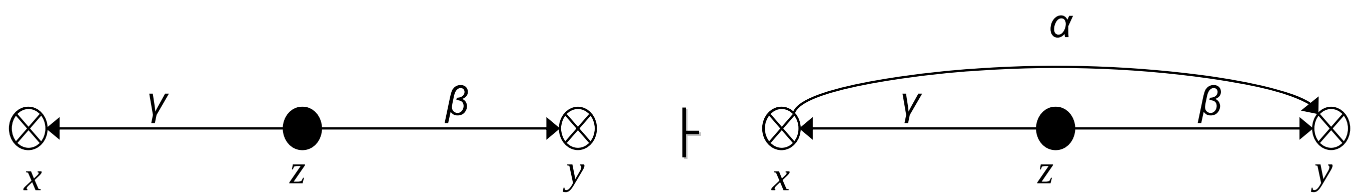


x takes (*α* to *y*) from *z*

19

Grant Rule

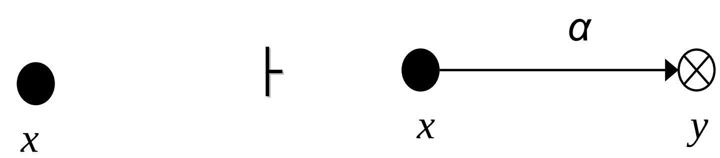
- if $g \in \gamma$, the grant rule produces another graph with a transitive edge $\alpha \subseteq \beta$ *added.*



z grants (α to y) to x

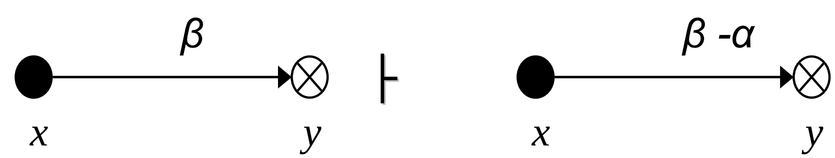
Create and Remove

3. Create rule:



x creates (*α* to new vertex) *y*

4. Remove rule:



x removes (*α* to) *y*

21

Exercise

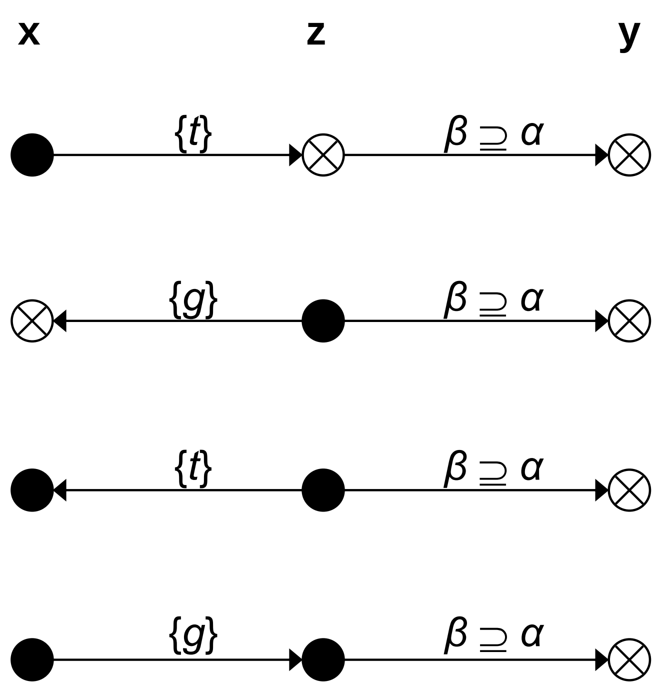
- Write a function using HRU operations that implement the
  - Take rule: call it TG_Take(??)
  - Grant rule: call it TG_Grant(??)

Take-Grant Protection Model: Sharing

- Given G_0, can vertex **x** obtain α rights over **y**?
  - Can_share(α, x, y, G_0) is true iff
    - $G_0 \vdash^* G_n$ using the four rules, &
    - There is an α edge from x to y in G_n
- *tg-path*: $\mathbf{v}_0, \ldots, \mathbf{v}_n$ with t or g edge between any pair of vertices $\mathbf{v}_i$, $\mathbf{v}_{i+1}$
  - Vertices *tg-connected* if *tg-path* between them
- Theorem: Any two subjects with *tg-path* of length 1 can share rights

Any two subjects with *tg-path* of length 1 can share rights

Can_share(α, *x*, *y*, G_0)

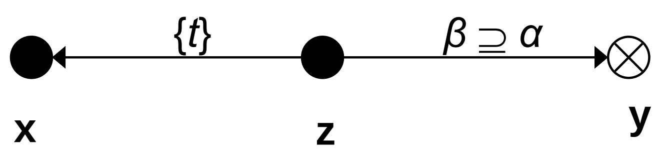


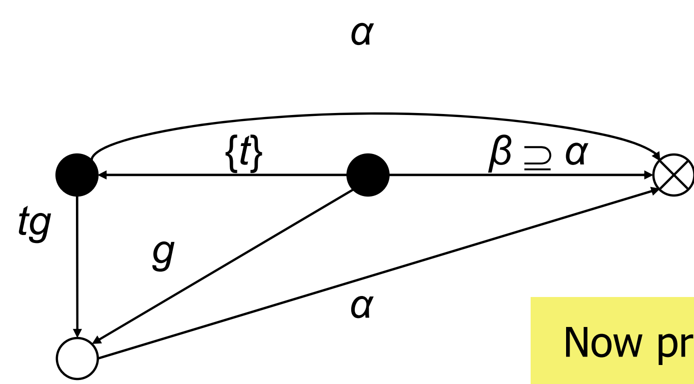
- Four possible length 1 *tg*-paths

1. Take rule
2. Grant rule
3. Lemma 3.1?
4. Lemma 3.2?

Any two subjects with *tg-path* of length 1 can share rights

Can_share(α, $\boldsymbol{x}$, $\boldsymbol{y}$, G_0)





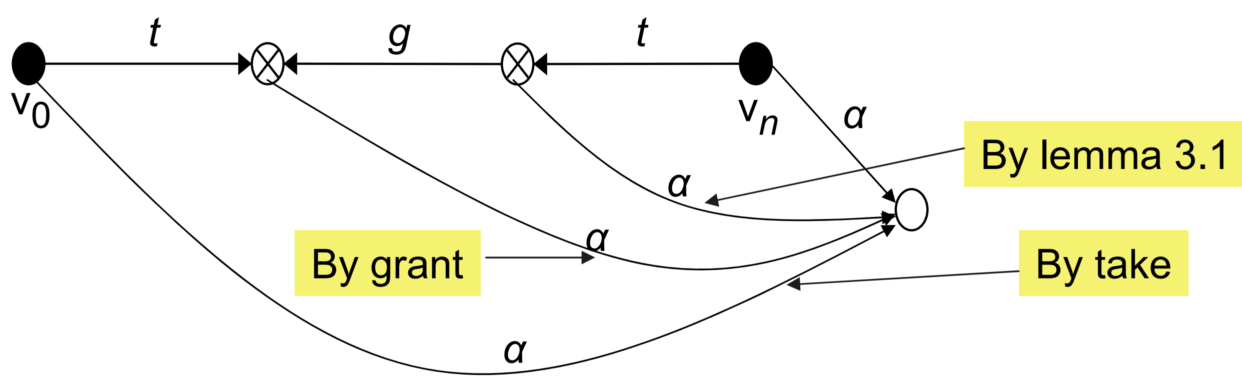
- Lemma 3.1
  - Sequence:
    - Create
    - Take
    - Grant
    - Take

Now prove lemma 3.2!

Other definitions

- Island: Maximal *tg*-connected subject-only subgraph
  - Can_share all rights in island
  - Proof: Induction from previous theorem
- Bridge: *tg*-path between subjects v_0 and v_n with edges of the following form:
  - $t_{\rightarrow}*, t_{\leftarrow}*$
  - $t_{\rightarrow}* g_{\rightarrow} t_{\leftarrow}*$
  - $t_{\rightarrow}*, g_{\leftarrow}, t_{\leftarrow}*$

Bridge

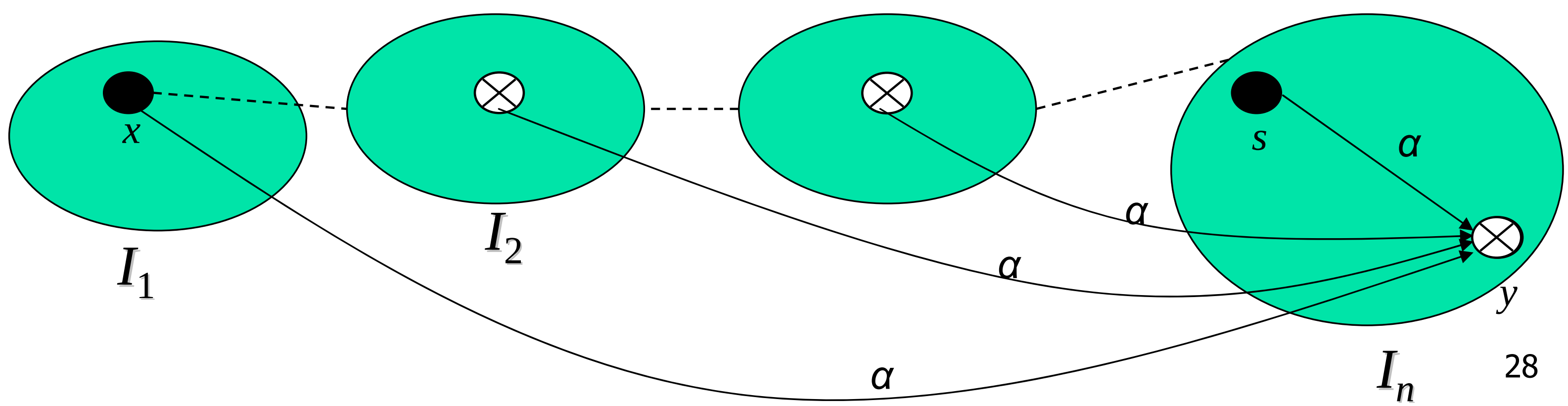


27

Theorem: Can_share(α,**x**,**y**,G_0) (for subjects)

- Subject_can_share(α, x, y, G_0) is true iff if x and y are subjects and
  - there is an α edge from x to y in G_0

  OR if:
  - ∃ a subject s ∈ G_0 with an ***s***-to-***y*** α edge, and
  - ∃ islands $I_1, \ldots, I_n$ such that $x \in I_1$, **s** ∈ I_n, and there is a bridge from I_j to I_{j+1}



28

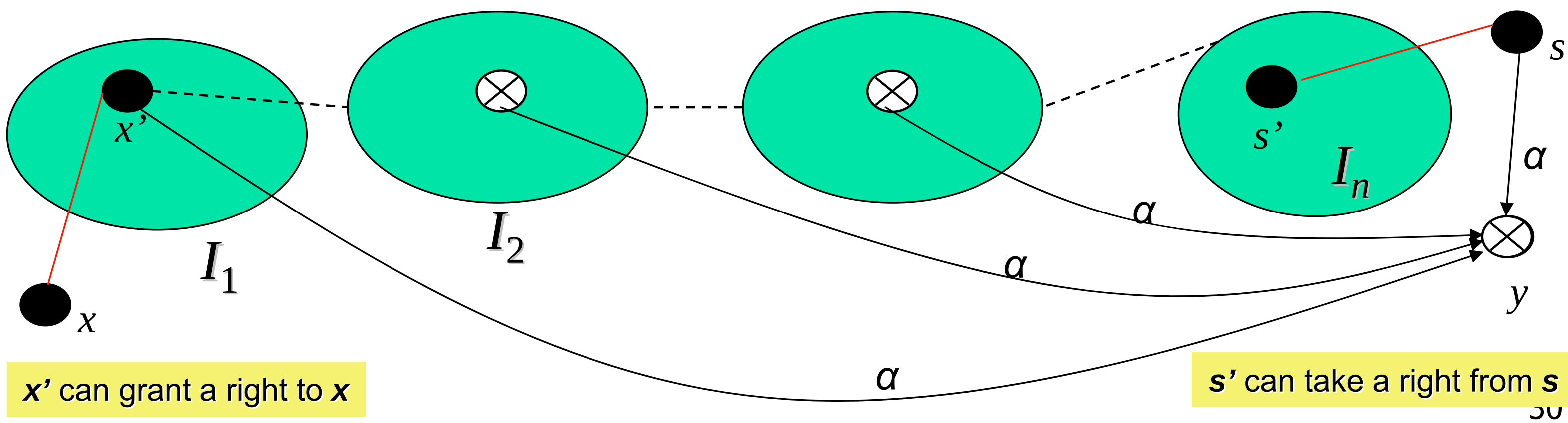
What about objects? Initial, terminal spans

- x *initially spans* to y if x is a subject and there is a *tg*-path between them with *t* edges ending in a *g* edge (i.e., $t_{\rightarrow}{}^{*}g_{\rightarrow}$)
  - x can grant a right to y
- x *terminally spans* to y if x is a subject and there is a *tg*-path between them with *t* edges (i.e., $t_{\rightarrow}{}^{*}$)
  - x can take a right from y

29

Theorem: Can_share(α,**x**,**y**,G_0)

- Can_share(α,x, y,G_0) iff there is an α edge from x to y in G_0 or if:
  - ∃ a vertex $s \in G_0$ with an s to y α edge,
  - ∃ a subject x' such that $x'=x$ or x' *initially spans* to x,
  - ∃ a subject s' such that $s'=s$ or s' *terminally spans* to s, and
  - ∃ islands I_1, …, I_n such that $x' \in I_1$, $s' \in I_n$, and there is a bridge from I_j to I_{j+1}



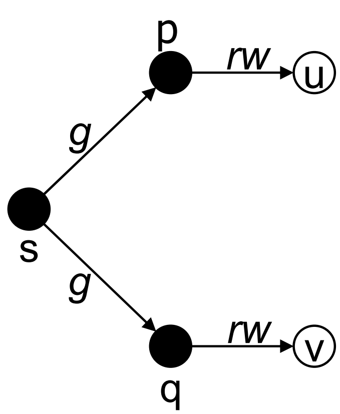
Theorem: Can_share(α,**x**,**y**,G_0)

- Corollary: There is an $O(|V|+|E|)$ algorithm to test can_share:
  - Decidable in linear time!!
- Protection state of the rules evolves
  - Following application on rules
  - Thus can characterize what set of states can be generated

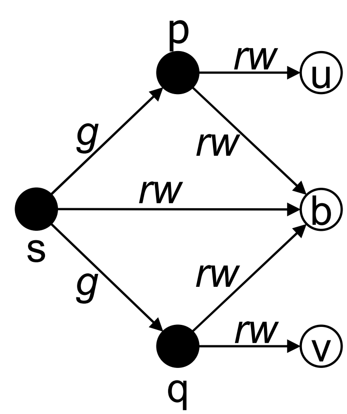
31

One example protection problem

- Sharing through a Trusted Entity
  - Let *p* and *q* be two processes
  - Let *b* be a buffer that they share to communicate
  - Let *s* be third party (e.g. operating system) that controls *b*



Witness
- S creates ({r, w}, to new object) b
- S grants ({r, w}, b) to p
- S grants ({r, w}, b) to q



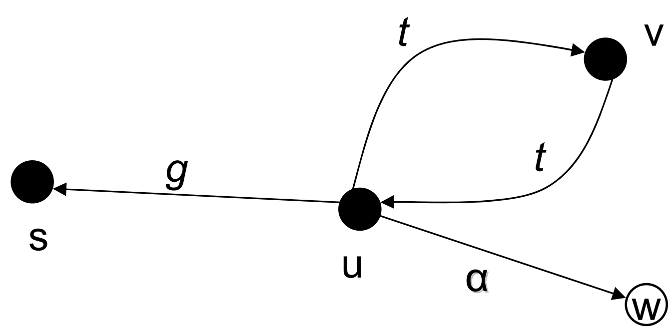
Theft in Take-Grant Model

- Can_steal(α,**x**,**y**,G_0) is true if there is no α edge from **x** to **y** in G_0 and $\exists$ sequence G_1, …, G_n s. t.:
  - $\exists$ α edge from **x** to **y** in G_n,
  - $\exists$ rules $\rho_1, \ldots, \rho_n$ that take $G_{i-1} \vdash \rho_i \; G_i$, and
  - $\forall$ **v**,**w** $\in G_i$, $1 \leq i < n$, if $\exists$ α edge from **v** to **y** in G_0 then ρ_i is not "**v** grants (α to **y**) to **w**"

- Disallows owners of α rights to y from transferring those rights
- Does not disallow them to transfer other rights
- Trojan horse??

A witness to theft

- Can u receive (α to w)?
  - u cannot grant (α to w) to anybody



34

Conspiracy

- Theft indicates cooperation: which subjects are actors in a transfer of rights, and which are not?
- Next question is
  - How many subjects are needed to enable Can_share(α,**x**,**y**,G_0)?
- Note that a vertex x
  - Can pass rights to any vertex to which it initially spans
    - $(t_{\rightarrow}{}^{*}g_{\rightarrow})$
  - Can take rights from any vertex to which it terminally spans
    - $(t_{\rightarrow}{}^{*})$

35

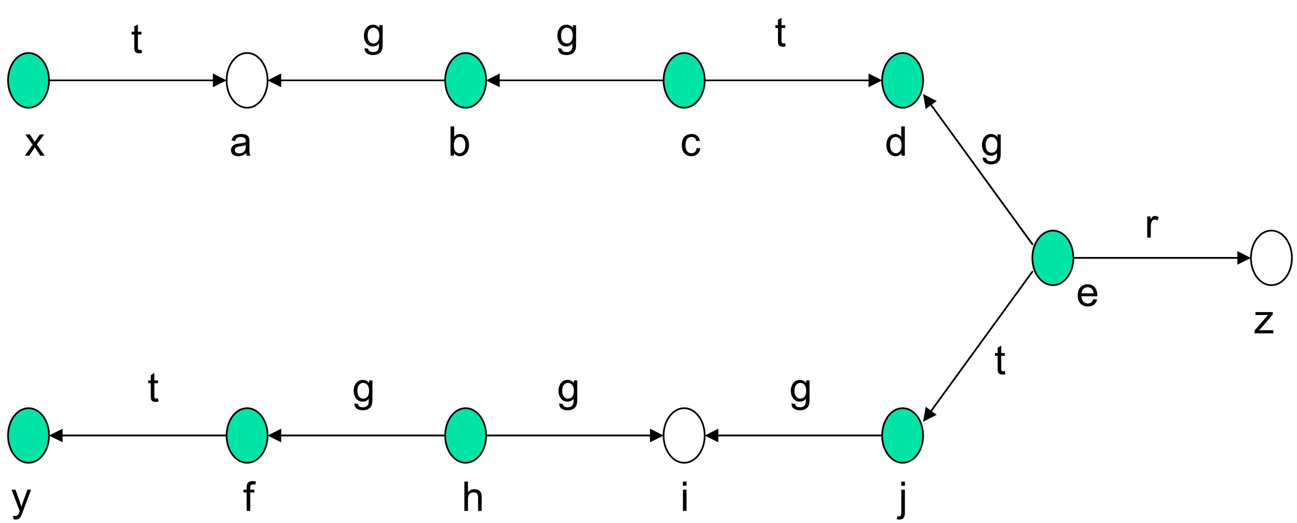
Conspiracy

- Access set A(**y**) with focus **y** (y is subject) is union of
  - set of vertices **y,**
  - vertices to which **y** initially spans, and
  - vertices to which **y** terminally spans
- Deletion set δ(y,y′): All z ∈ A(y) ∩ A(y′) for which
  - **y** initially spans to **z** and **y'** terminally spans to **z**
  - **y** terminally spans to **z** and **y'** initially spans to **z**
  - **z**=**y** & **z**=**y'**

Conspiracy

- Conspiracy graph H of G_0:
  - Represents the paths along which subjects can transfer rights
  - For each subject in G_0, there is a corresponding vertex h(x) in H
  - if δ(**y**,**y'**) not empty, edge from h(**y)** to h(**y')**

Example: draw the conspiracy graph



How many minimum conspirators involved in Can_share(α,**x**,**y**,G_0)?

Policy Composition

Problem: *Consistent* Policies

- Policies defined by different organizations
  - Different needs
  - But sometimes subjects/objects overlap
- Can all policies be met?
  - Different categories
    - Build lattice combining them
  - Different security levels
    - Need to be *levels* – thus must be able to order
  - What if different DAC and MAC policies need to be integrated?

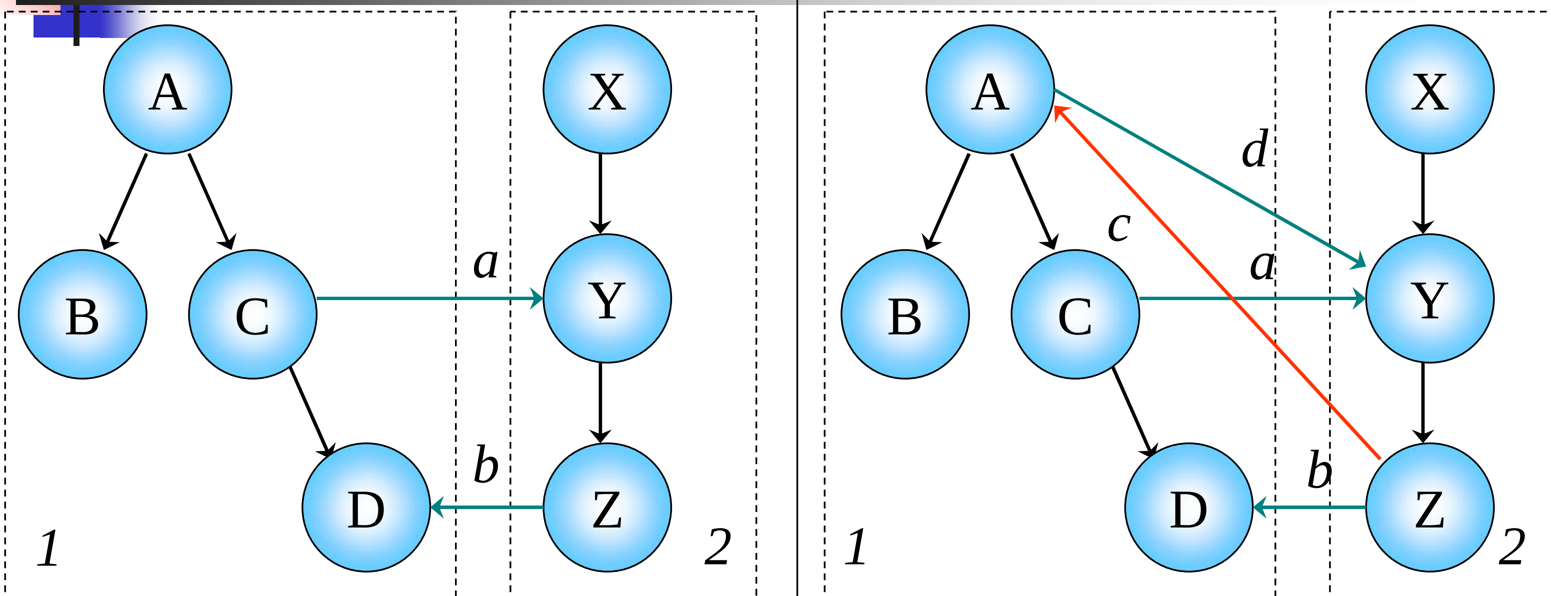
Secure Interoperability

- Principles of secure interoperation [Gong, 96]

  Principle of autonomy
  - If an access is permitted within an individual system, it must also be permitted under secure interoperation

  Principle of security
  - If an access is not permitted within an individual system, it must not be permitted under secure interoperation
- Interoperation of secure systems can create new security breaches

Secure Interoperability (Example)



F_{12} = {a, b}

F_{12} = {a, b, c, d}

F_{12} - permitted access between systems 1 and 2

(1) F_{12} = {a, b, d}
Direct access

(2) F_{12} = {c}
Indirect access

Summary

- RBAC is a promising approach
  - Lot of efforts currently expended for this
- Take Grant
  - Restricted model – easy to analyze
    - but usefulness?
- Secure interoperation
  - Growing problem