ECS 289M Lecture 4

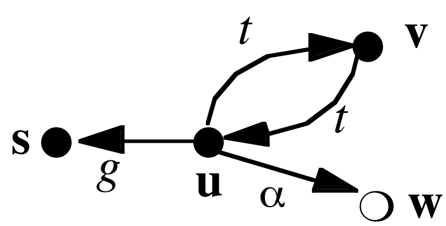
April 7, 2006

can•steal Predicate

Definition:

- *can•steal*(r, **x**, **y**, G_0) if, and only if, there is no edge from **x** to **y** labeled r in G_0, and the following hold simultaneously:
  - There is edge from **x** to **y** labeled r in G_n
  - There is a sequence of rule applications $\rho_1, \ldots, \rho_n$ such that $G_{i-1} \vdash G_i$ using ρ_i
  - For all vertices **v**, **w** in G_{i-1}, if there is an edge from **v** to **y** in G_0 labeled r, then ρ_i is ***not*** of the form "**v** grants (r to **y**) to **w**"

Example



- *can•steal*(α, **s**, **w**, G_0):

1. **u** grants (*t* to **v**) to **s**
2. **s** takes (*t* to **u**) from **v**
3. **s** takes (α to **w**) from **u**

can•steal Theorem

- *can•steal*(*r*, **x**, **y**, G_0) if, and only if, the following hold simultaneously:
  a) There is no edge from **x** to **y** labeled *r* in G_0
  b) There exists a subject **x**′ such that **x**′ = **x** or **x**′ initially spans to **x**
  c) There exists a vertex **s** with an edge labelled α to **y** in G_0
  d) *can•share*(*t*, **x**′, **s**, G_0) holds

Outline of Proof

$\Rightarrow$: Assume conditions hold

- **x** subject
  - **x** gets *t* rights to **s**, then takes α to **y** from **s**
- **x** object
  - *can•share*(*t*, **x**′, **s**, G_0) holds
  - If **x**′ has no α edge to **y** in G_0, **x**′ takes (α to **y**) from **s** and grants it to **x**
  - If **x**′ has a edge to **y** in G_0, **x’** creates surrogate **x**″, gives it (*t* to **s**) and (*g* to **x**″); then **x**″ takes (α to **y**) and grants it to **x**

Outline of Proof

$\Leftarrow$: Assume *can•steal*(α, **x**, **y**, G_0) holds

- First two conditions immediate from definition of *can•steal*, *can•share*
- Third condition immediate from theorem of conditions for *can•share*
- Fourth condition: ρ minimal length sequence of rule applications deriving G_n from G_0; *i* smallest index such that $G_{i-1} \vdash G_i$ by rule ρ_i and adding α from some **p** to **y** in G_i
  - What is ρ_i?

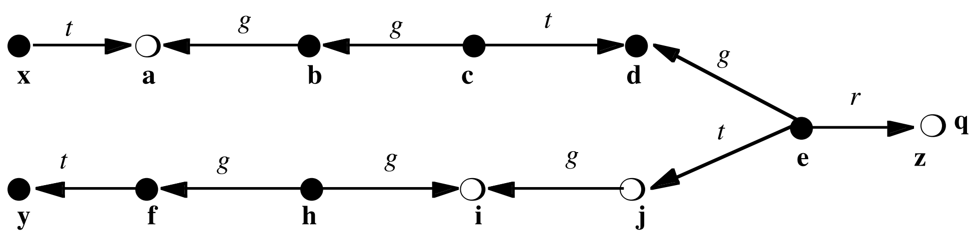
Outline of Proof

- Not remove or create rule
  - **y** exists already
- Not grant rule
  - G_i first graph in which edge labeled α to **y** is added, so by definition of *can•share*, cannot be grant
- take rule: so *can•share*(t, **p**, **s**, G_0) holds
  - So is subject **s**′ such that **s**′ = **s** or terminally spans to **s**
  - Sequence of islands with **x**′ $\in I_1$ and **s**′ $\in I_n$
- Derive witness to *can•share*(t, **x**′, **s**, G_0) that does not use "**s** grants (α to **y**) to" anyone

Conspiracy

- Minimum number of actors to generate a witness for *can•share*(α, **x**, **y**, G_0)
- Access set describes the "reach" of a subject
- Deletion set is set of vertices that cannot be involved in a transfer of rights
- Build *conspiracy graph* to capture how rights flow, and derive actors from it

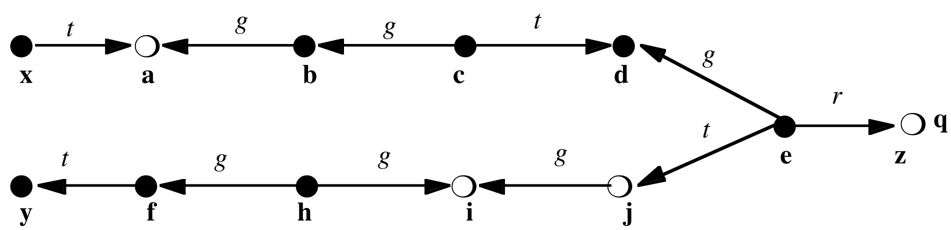
Example



Access Set

- *Access set A(**y**) with focus* **y**: set of vertices:
  - { **y** }
  - { **x** | **y** initially spans to **x** }
  - { **x’** | **y** terminally spans to **x** }
- Idea is that focus can give rights to, or acquire rights from, a vertex in this set

Example

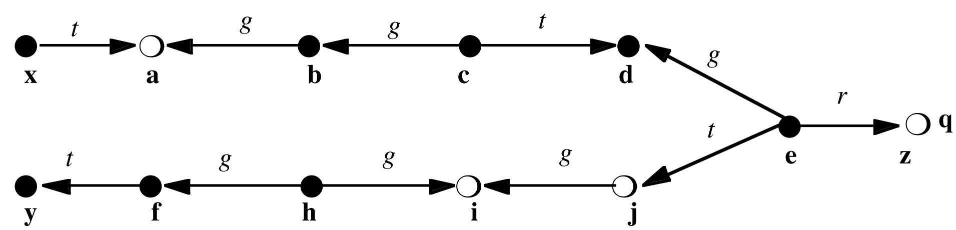


- A(**x**) = { **x**, **a** }
- A(**b**) = { **b**, **a** }
- A(**c**) = { **c**, **b**, **d** }
- A(**d**) = { **d** }
- A(**e**) = { **e**, **d**, **i**, **j** }
- A(**y**) = { **y** }
- A(**f**) = { **f**, **y** }
- A(**h**) = { **h**, **f**, **i** }

Deletion Set

- Deletion set δ(**y**, **y′**): contains those vertices in A(**y**) ∩ A(**y′**) such that:
  - **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′**
- Idea is that rights can be transferred between **y** and **y′** if this set non-empty

Example

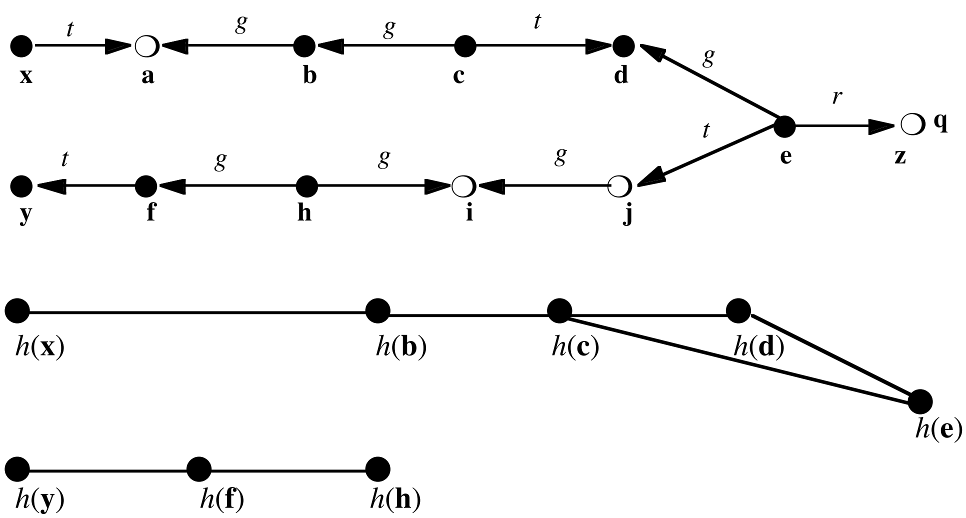


- δ(**x**, **b**) = { **a** }
- δ(**b**, **c**) = { **b** }
- δ(**c**, **d**) = { **d** }
- δ(**c**, **e**) = { **d** }
- δ(**d**, **e**) = { **d** }
- δ(**y**, **f**) = { **y** }
- δ(**h**, **f**) = { **f** }

Conspiracy Graph

- Abstracted graph *H* from G_0:
  - Each subject **x** ∈ G_0 corresponds to a vertex *h*(**x**) ∈ *H*
  - If δ(**x**, **y**) ≠ ∅, there is an edge between *h*(**x**) and *h*(**y**) in *H*
- Idea is that if *h*(**x**), *h*(**y**) are connected in *H*, then rights can be transferred between **x** and **y** in G_0

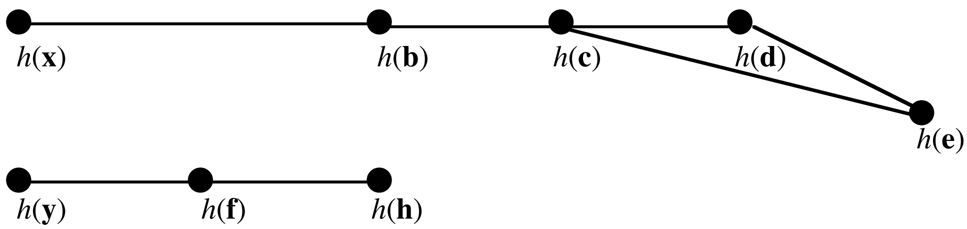
Example



Results

- *I*(**x**): *h*(**x**), all vertices *h*(**y**) such that **y** initially spans to **x**
- *T*(**x**): *h*(**x**), all vertices *h*(**y**) such that **y** terminally spans to **x**
- Theorem: *can•share*(α, **x**, **y**, G_0) iff there exists a path from some *h*(**p**) in *I*(**x**) to some *h*(**q**) in *T*(**y**)
- Theorem: *l* vertices on shortest path between *h*(**p**), *h*(**q**) in above theorem; *l* conspirators necessary and sufficient to witness

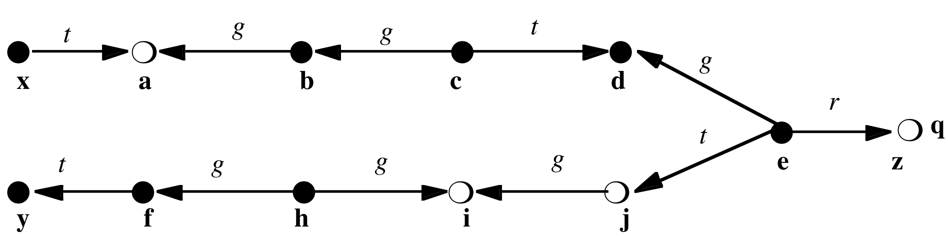
Example: Conspirators



- I(**x**) = { h(**x**) }, T(**z**) = { h(**e**) }
- Path between h(**x**), h(**e**) so *can•share*(r, **x**, **z**, G_0)
- Shortest path between h(**x**), h(**e**) has 4 vertices

⇒ Conspirators are **e**, **c**, **b**, **x**

Example: Witness



- **e** grants (r to **z**) to **d**
- **c** takes (r to **z**) from **d**
- **c** grants (r to **z**) to **b**
- **b** grants (r to **z**) to **a**
- **x** takes (r to **z**) from **a**

Key Question

- Characterize class of models for which safety is decidable
  - Existence: Take-Grant Protection Model is a member of such a class
  - Universality: In general, question undecidable, so for some models it is not decidable
- What is the dividing line?

Schematic Protection Model

- Type-based model
  - Protection type: entity label determining how control rights affect the entity
    - Set at creation and cannot be changed
  - Ticket: description of a single right over an entity
    - Entity has sets of tickets (called a *domain*)
    - Ticket is **X**/*r*, where **X** is entity and *r* right
  - Functions determine rights transfer
    - Link: are source, target "connected"?
    - Filter: is transfer of ticket authorized?

Link Predicate

- Idea: $link_i$(**X**, **Y**) if **X** can assert some control right over **Y**
- Conjunction of disjunction of:
  - **X**/*z* ∈ *dom*(**X**)
  - **X**/*z* ∈ *dom*(**Y**)
  - **Y**/*z* ∈ *dom*(**X**)
  - **Y**/*z* ∈ *dom*(**Y**)
  - **true**

Examples

- Take-Grant:

  link(**X**, **Y**) = **Y**/*g* ∈ *dom*(**X**) v **X**/*t* ∈ *dom*(**Y**)
- Broadcast:

  link(**X**, **Y**) = **X**/*b* ∈ *dom*(**X**)
- Pull:

  link(**X**, **Y**) = **Y**/*p* ∈ *dom*(**Y**)

Filter Function

- Range is set of copyable tickets
  - Entity type, right
- Domain is subject pairs
- Copy a ticket **X**/*r*:*c* from *dom*(**Y**) to *dom*(**Z**)
  - **X**/*rc* ∈ *dom*(**Y**)
  - $link^i$(**Y**, **Z**)
  - τ(**Y**)/*r*:*c* ∈ f_i(τ(**Y**), τ(**Z**))
- One filter function per link function

Example

- *f*(τ(**Y**), τ(**Z**)) = *T* × *R*
  - Any ticket can be transferred (if other conditions met)
- *f*(τ(**Y**), τ(**Z**)) = *T* × *RI*
  - Only tickets with inert rights can be transferred (if other conditions met)
- *f*(τ(**Y**), τ(**Z**)) = ∅
  - No tickets can be transferred

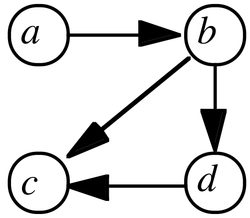
Example

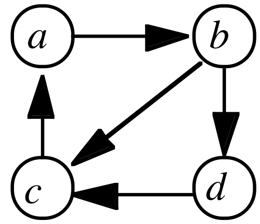
- Take-Grant Protection Model
  - *TS* = { subjects }, *TO* = { objects }
  - *RC* = { *tc*, *gc* }, *RI* = { *rc*, *wc* }
  - *link*(**p**, **q**) = **p**/*t* ∈ *dom*(**q**) ∨ **q**/*g* ∈ *dom*(**p**)
  - *f*(*subject*, *subject*) = { *subject*, *object* } ×

    { *tc*, *gc*, *rc*, *wc* }

Create Operation

- Must handle type, tickets of new entity
- Relation *cc*(*a*, *b*) [*cc* for *can-create*]
  - Subject of type *a* can create entity of type *b*
- Rule of acyclic creates:





Types

- $cr(a, b)$: tickets created when subject of type a creates entity of type b [*cr* for *create-rule*]
- **B** object: $cr(a, b) \subseteq \{ b/r{:}c \in RI \}$
  - **A** gets **B**/$r{:}c$ iff $b/r{:}c \in cr(a, b)$
- **B** subject: $cr(a, b)$ has two subsets
  - $cr_P(a, b)$ added to **A**, $cr_C(a, b)$ added to **B**
  - **A** gets **B**/$r{:}c$ if $b/r{:}c \in cr_P(a, b)$
  - **B** gets **A**/$r{:}c$ if $a/r{:}c \in cr_C(a, b)$

Non-Distinct Types

$cr(a, a)$: who gets what?

- $self/r{:}c$ are tickets for creator
- $a/r{:}c$ tickets for created

$cr(a, a) = \{ a/r{:}c, self/r{:}c \mid r{:}c \in R\}$

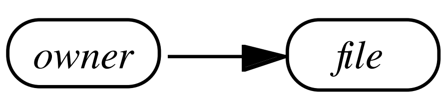
Attenuating Create Rule

cr(*a*, *b*) attenuating if:

1. $cr_C(a, b) \subseteq cr_P(a, b)$ and
2. $a/r{:}c \in cr_P(a, b) \Rightarrow self/r{:}c \in cr_P(a, b)$

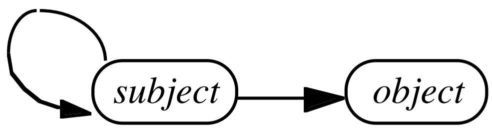
Example: Owner-Based Policy

- Users can create files, creator can give itself any inert rights over file
  - *cc* = { (*user* , *file*) }
  - *cr*(*user*, *file*) = { *file*/*r*:*c* | *r* ∈ *RI* }
- Attenuating, as graph is acyclic, loop free



Example: Take-Grant

- Say subjects create subjects (type *s*), objects (type *o*), but get only inert rights over latter
  - $cc = \{ (s, s), (s, o) \}$
  - $cr_C(a, b) = \varnothing$
  - $cr_P(s, s) = \{s/tc, s/gc, s/rc, s/wc\}$
  - $cr_P(s, o) = \{s/rc, s/wc\}$
- Not attenuating, as no *self* tickets provided; *subject* creates *subject*



Safety Analysis

- Goal: identify types of policies with tractable safety analyses
- Approach: derive a state in which additional entries, rights do not affect the analysis; then analyze this state
  - Called a *maximal state*