Relationship-Based Access Control (ReBAC or RAC)

Prof. Ravi Sandhu
Executive Director and Endowed Chair

March 4, 2016

ravi.sandhu@utsa.edu
www.profsandhu.com

RAC Models

U2U, U2R, R2R with attributes	URRAC_A	Cheng et al 2014
U2U, U2R, R2R	URRAC	Cheng et al 2012-09
U2U	UURAC	Cheng et al 2012-07 Cheng et al 2015

RAC Models

U2U, U2R, R2R with attributes	**$URRAC_A$**	**Cheng et al 2014**
U2U, U2R, R2R	**URRAC**	**Cheng et al 2012-09**
U2U	**UURAC**	**Cheng et al 2012-07** **Cheng et al 2015**

UURAC Motivation

- Users in Online Social Networks (OSNs) are connected by social relationships (user-to-user relationships U2U)

- Owner of a resource can control its release based on U2U relationships between the access requester and the owner

UURAC Motivation

- OSNs keep massive resources and support varied activities for users
- Users want to regulate access to their resources and activities related to them (as a requester or target)
- Some related users also expect control on how the resource or user can be exposed

UURAC Motivation

- What current friend-of-friend approach cannot do?
  - User who is tagged in a photo wants to keep her image private (Related User’s Control)
  - Mom doesn’t want her kid to become friend with her colleagues (Parental Control)
  - Employee promotes his resume to headhunters without letting his current employer know (Allowing farther users but preventing closer users)

Characteristics of Access Control in OSNs

- Policy Individualization
  - Users define their own privacy and activity preferences
  - Related users can configure policies too
  - Collectively used by the system for control decision
- User and Resource as a Target
  - e.g., poke, messaging, friendship invitation, etc.
- User Policies for Outgoing and Incoming Actions
  - User can be either requester or target of activity
  - Allows control on 1) activities w/o knowing a particular resource and 2) activities against the user w/o knowing a particular access requestor
  - e.g., block notification of friend's activities; restrict from viewing violent contents
- Relationship-based Access Control

UURAC Approach

- Using regular expression-based path pattern for arbitrary combination of relationship types
- Given relationship path pattern and hopcount limit, graph traversal algorithm checks the social graph to determine access

Related Works

	Fong [7]	Fong [8, 9]	Carminati [6]	Carminati [2, 3]	UURAC
Relationship Category					
Multiple Relationship Types		✓	✓	✓	✓
Directional Relationship		✓	✓		✓
U2U Relationship	✓	✓	✓	✓	✓
U2R Relationship				✓	
Model Characteristics					
Policy Individualization	✓	✓	✓	✓	✓
User & Resource as a Target				(partial)	✓
Outgoing/Incoming Action Policy				(partial)	✓
Relationship Composition					
Relationship Depth	0 to 2	0 to n	1 to n	1 to n	0 to n
Relationship Composition	f, f of f	exact type sequence	path of same type	exact type sequence	path pattern of different types

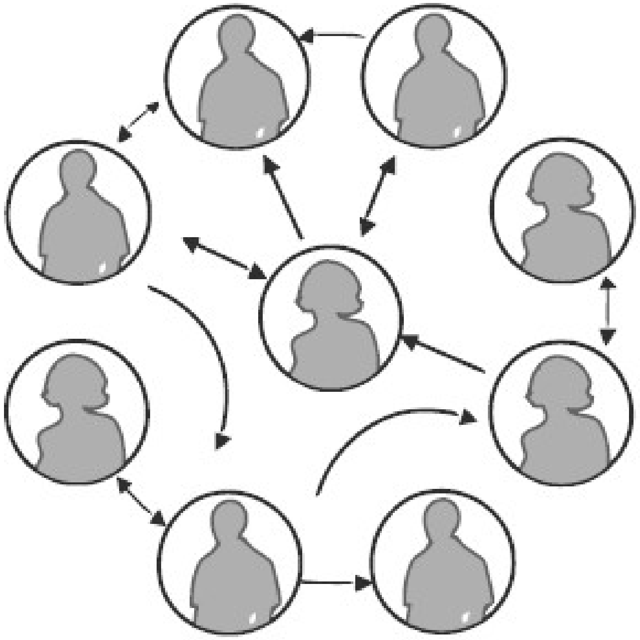
- The advantages of this approach:
  - Passive form of action allows outgoing and incoming action policy
  - Path pattern of different relationship types make policy specification more expressive

Related Works

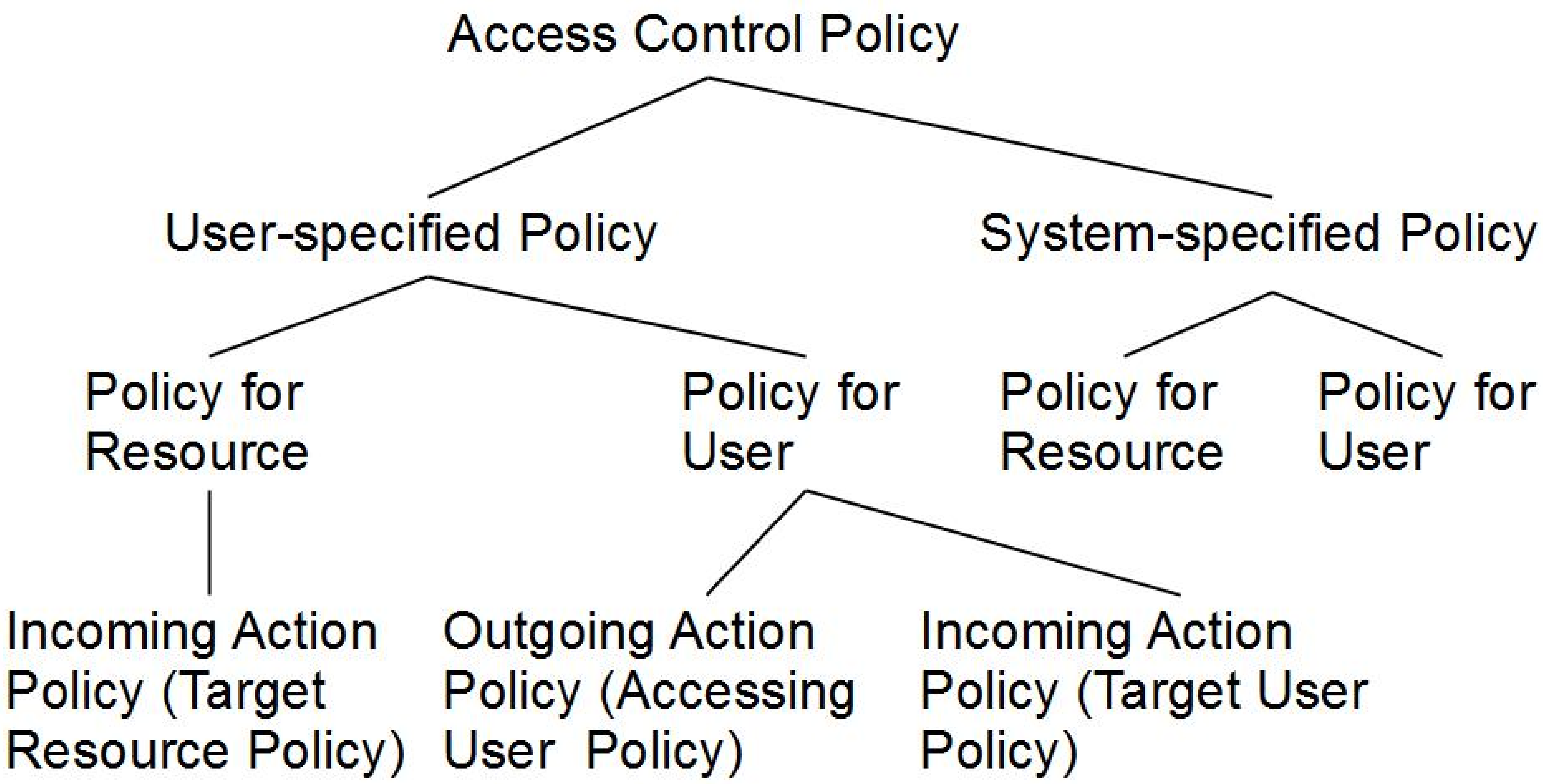
- Not to forget PGP web of trust (mid 1990s)

Social Networks

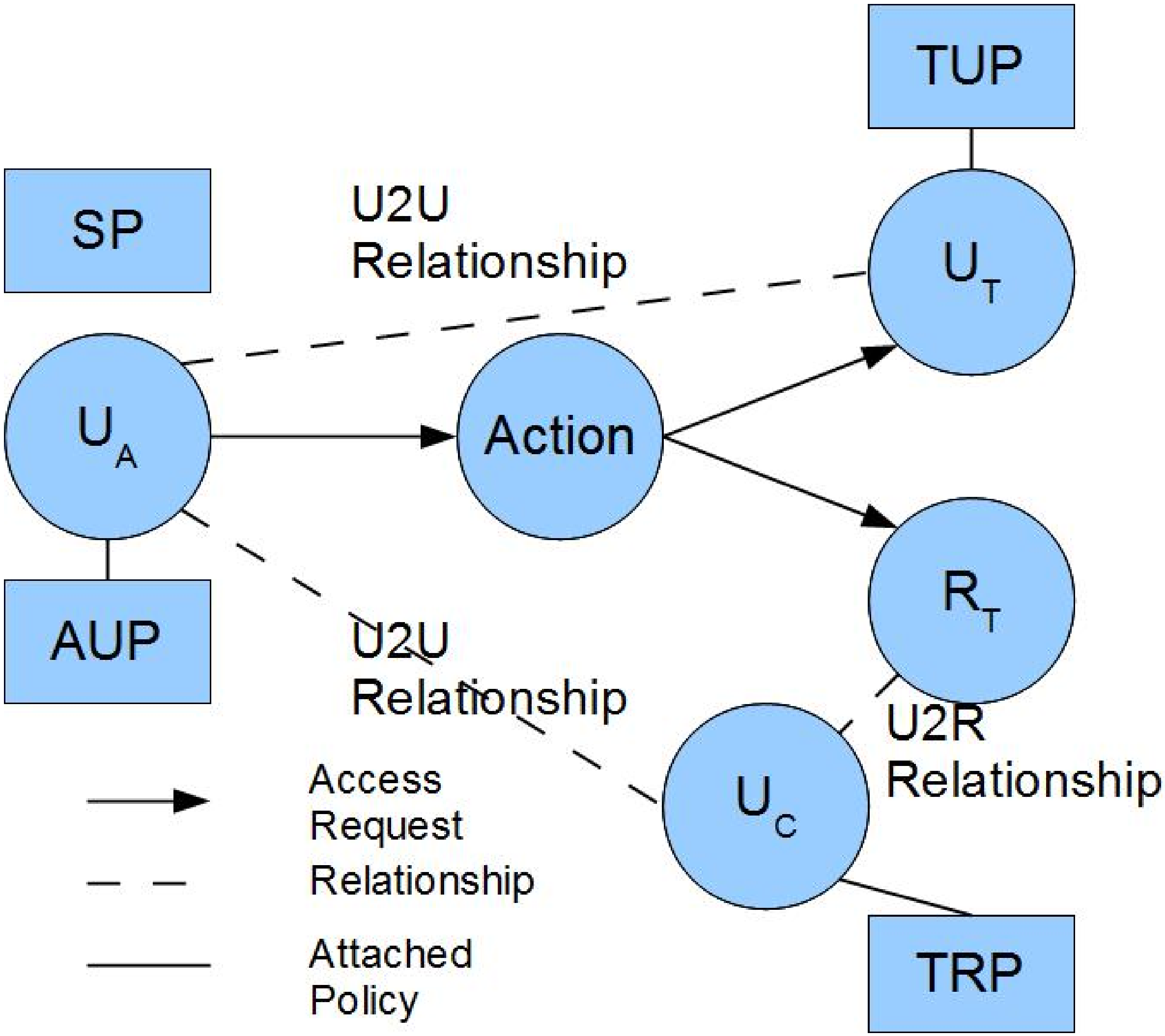
- Social graph is modeled as a directed labeled simple graph *G*=<*U*, *E*, *Σ*>
  - Nodes *U* as users
  - Edges *E* as relationships
  - $\Sigma=\{\sigma_1, \sigma_2, \ldots, \sigma_n, \sigma_1^{-1}, \sigma_2^{-1}, \ldots, \sigma_n^{-1}\}$ as relationship types supported



Policy Taxonomy



UURAC Model Components



U_A: Accessing User
U_T: Target User
U_C: Controlling User
R_T: Target Resource
AUP: Accessing User Policy
TUP: Target User Policy
TRP: Target Resource Policy
SP: System Policy

Access Request and Evaluation

- Access Request <u_a, *action*, *target*>
  - u_a tries to perform *action* on *target*
  - Target can be either user u_t or resource r_t
- Policies and Relationships used for Access Evaluation
  - When u_a requests to access a user u_t
    - u_a's AUP, u_t's TUP, SP
    - U2U relationships between u_a and u_t
  - When u_a requests to access a resource r_t
    - u_a's AUP, r_t's TRP (associated with u_c), SP
    - U2U relationships between u_a and u_c

Policy Representations

Accessing User Policy	$< action, (start, path\ rule) >$
Target User Policy	$< action^{-1}, (start, path\ rule) >$
Target Resource Policy	$< action^{-1}, r_t, (start, path\ rule) >$
System Policy for User	$< action, (start, path\ rule) >$
System Policy for Resource	$< action, r.type, (start, path\ rule) >$

- *$action^{-1}$* in TUP and TRP is the passive form since it applies to the recipient of action
- TRP has an extra parameter r_t to distinguish the actual target resource it applies to
  - owner(r_t)→ a list of u_c→U2U relationships between u_a and u_c
- SP does not differentiate the active and passive forms
- SP for resource needs *r.type* to refine the scope of the resource

Graph Rule Grammar

$GraphRule ::= \text{“(”} < StartingNode > \text{“,”} < PathRule > \text{“)”}$
$PathRule ::= < PathSpecExp > | < PathSpecExp >< Connective >< PathRule >$
$Connective ::= \vee | \wedge$
$PathSpecExp ::= < PathSpec > | \neg < PathSpec >$
$PathSpec ::= \text{“(”} < Path > \text{“,”} < HopCount > \text{“)”} | \text{“(”} < EmptySet > \text{“,”} < Hopcount > \text{“)”}$
$HopCount ::= < Number >$
$Path ::= < TypeExp > | < TypeExp >< Path >$
$EmptySet ::= \emptyset$
$TypeExp ::= < TypeSpecifier > | < TypeSpecifier >< Wildcard >$
$StartingNode ::= u_a | u_t | u_c$
$TypeSpecifier ::= \sigma_1 | \sigma_2 | .. | \sigma_n | \sigma_1^{-1} | \sigma_2^{-1} | .. | \sigma_n^{-1} | \Sigma \text{ where } \Sigma = \{\sigma_1, \sigma_2, .., \sigma_n, \sigma_1^{-1}, \sigma_2^{-1}, .., \sigma_n^{-1}\}$
$Wildcard ::= \text{“} * \text{”} | \text{“?”} | \text{“} + \text{”}$
$Number ::= [0-9]+$

Example

- Alice's policy P_{Alice}: $< poke, (u_a, (f*, 3))>$ $< poke^{-1}, (u_t, (f, 1))>$ $< read, (u_a, (\Sigma*, 5))>$ $< read^{-1}, file1, (u_c, (cf*, 4))>$
- Harry's policy P_{Harry}: $< poke, (u_a, (cf*, 5) \vee (f*, 5))>$ $< poke^{-1}, (u_t, (f*, 2))>$ $< read^{-1}, file2, (u_c, \neg(p+, 2)>$
- System's policy P_{Sys}: $< poke, (u_a, (\Sigma*, 5))>$ $< read, photo, (u_a, (\Sigma*, 5))>$

- "Only Me"
  - <poke, (ua, (Ø, 0))> says that ua can only poke herself
  - <poke^{-1}, (ut, (Ø, 0))> specifies that ut can only be poked by herself
- The Use of Negation Notation
  - (fffc ∧ ¬fc) allows the coworkers of the user's distant friends to see, while keeping away the coworkers of the user's direct friends

Policy Collecting

- To authorize (u_a, action, target) if target = u_t
  - E.g., (Alice, poke, Harry)

AUP

TUP

SP

P_{Alice}

<poke, (ua, (f*,3))>

< poke^{-1}, (ua, (f*,3))>

P_{Harry}

<poke, (ua, (cf*,5)$\vee$(f*,5))>

<poke^{-1}, (ut, (f*,2))>

P_{Sys}

<poke, (ua, (Σ*,5))>

Policy Collecting

- To authorize (u_a, action, target) if target = u_t
  - E.g., (Alice, poke, Harry)

AUP

<poke, (ua, (f*,3))>

TUP

<poke^{-1}, (ut, (f*,2))>

SP

<poke, (ua, (Σ*,5))>

P_{Alice}

< poke^{-1}, (ua, (f*,3))>

P_{Harry}

<poke, (ua, (cf*,5)∨(f*,5))>

P_{Sys}

Policy Collecting

- To authorize (u_a, action, target) if target = r_t
  - Determine the controlling user for r_t:
    - $u_c \leftarrow owner(r_t)$
  - E.g., (Alice, read, file2)

AUP

P_{Alice}

<read, (u_a, (Σ*, 5))>

<read^{-1}, file1, (u_c, (cf*, 4))>

TRP

P_{Harry}

<read^{-1}, file2, (u_c, ¬(p+, 2))>

SP

P_{Sys}

<read, photo, (ua, (Σ*, 5))>

Policy Collecting

- To authorize (u_a, action, target) if target = r_t
  - Determine the controlling user for r_t:
    - $u_c \leftarrow owner(r_t)$
  - E.g., (Alice, read, file2)

AUP

<read, (u_a, (Σ^*, 5))>

TRP

<read^{-1}, file2, (u_c, ¬(p+, 2))>

SP

P_{Alice}

<read^{-1}, file1, (u_c, (cf*, 4))>

P_{Harry}

P_{Sys}

<read, photo, (ua, (Σ^*, 5))>

Policy Extraction

- Policy: <*action*, *r.type*, *graph rule*>

↓

- Graph Rule: *start*, *path rule*

↓

- Path Rule: *path spec* ∧|∨ *path spec*

↓

- Path Spec: *path*, *hopcount*

Policy Evaluation

- Evaluate a combined result based on conjunctive or disjunctive connectives between path specs
- Make a collective result for multiple policies in each policy set.
  - Policy conflicts may arise. We assume system level conflict resolution strategy is available (e.g., disjunctive, conjunctive, prioritized).
- Compose the final result from the result of each policy set (AUP, TUP/TRP, SP)

Path Checking Algorithm Overview

- Parameters: G, path, hopcount, s, t
- Traversal Order: Depth-First Search
  - Why not BFS?
    - Activities in OSN typically occur among people with close distance
    - DFS needs only one pair of variables to keep the current status and history of exploration
    - Hopcount limit prevents DFS from lengthy useless search

Path Checking Algorithm Complexity

- Time complexity is bounded between [$O(dmin^{Hopcount})$,$O(dmax^{Hopcount})$], where *dmax* and *dmin* are maximum and minimum out-degree of node
  - Users in OSNs usually connect with a small group of users directly, the social graph is very sparse
  - Given the constraints on the relationship types and hopcount limit, the size of the graph to be explored can be dramatically reduced

RAC Models

U2U, U2R, R2R with attributes	URRAC_A	Cheng et al 2014
U2U, U2R, R2R	URRAC	Cheng et al 2012-09
U2U	UURAC	Cheng et al 2012-07 Cheng et al 2015

Limitation of U2U Relationships

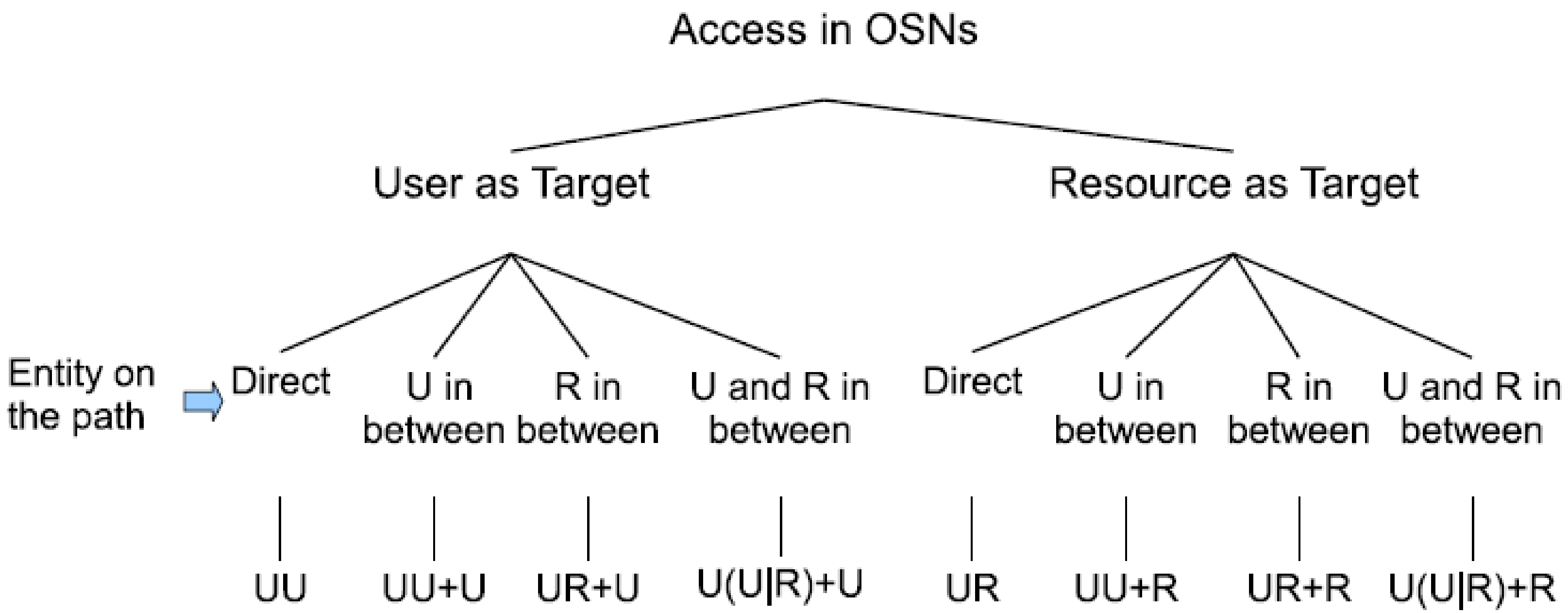
- We rely on the controlling user and ownership to regulate access to resources in UURAC (U2U Relationship-based AC)

- Needs more flexible control
  - Parental control, related user’s control (e.g., tagged user)
  - User relationships to resources (e.g., U-U-R)
  - User relationships via resources (e.g., U-R-U)

Beyond U2U Relationships

- There are various types of relationships between users and resources in addition to U2U relationships and ownership
  - e.g., share, like, comment, tag, etc
- U2U, U2R and R2R
- U2R further enables relationship and policy administration

Access Scenarios



(b) Taxonomy based on Entity on the path

Related Works

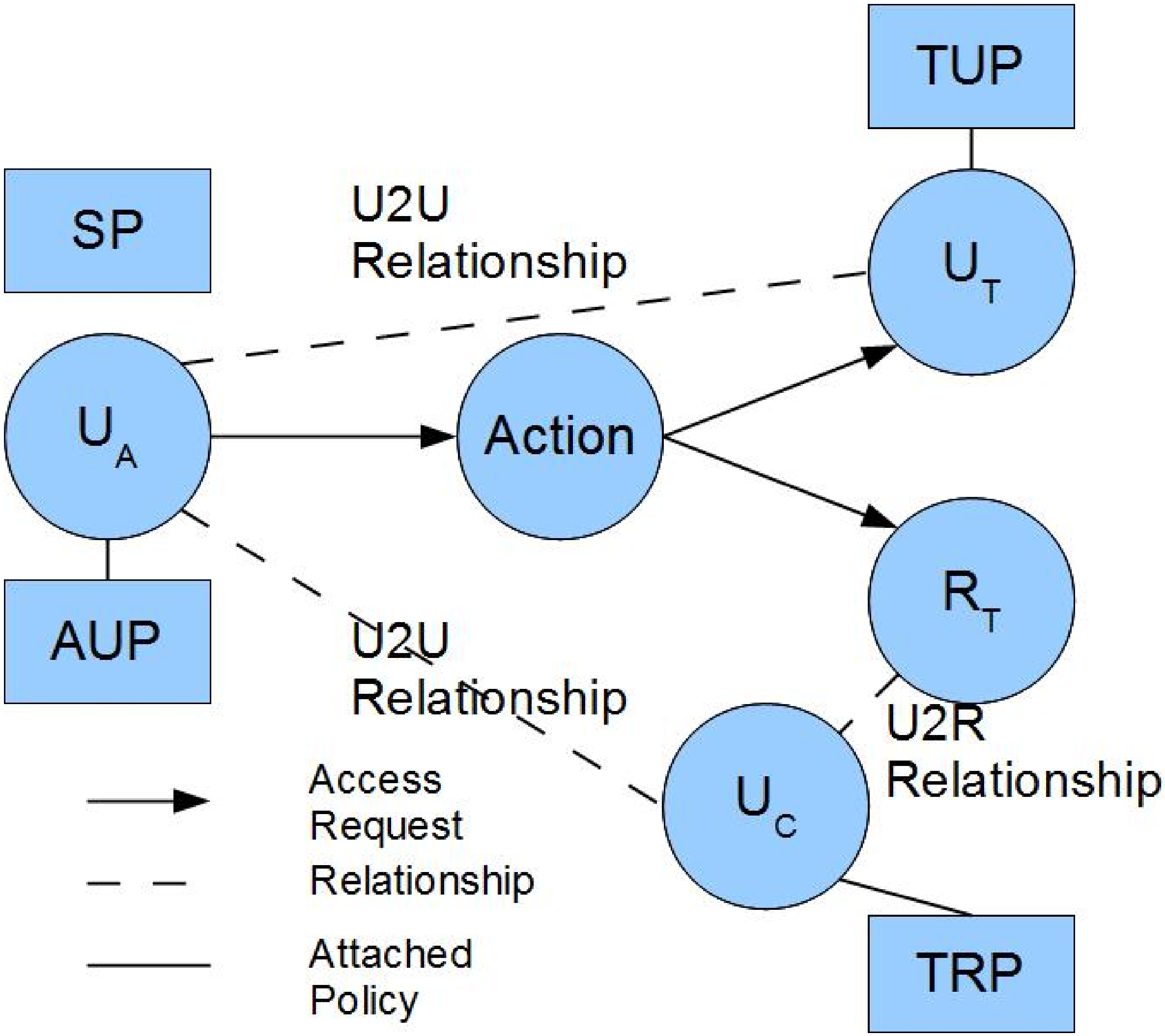
- Access Control Models for OSNs

Comparison of Access Control Models for OSNs

	Fong [14]	Fong [15], [16]	Carminati [10]	Carminati [6], [7]	UURAC	URRAC
Relationship Category						
Multiple Relationship Types		✓	✓	✓	✓	✓
Directional Relationship		✓	✓		✓	✓
U2U Relationship	✓	✓	✓	✓	✓	✓
U2R Relationship				✓		✓
Model Characteristics						
Policy Individualization	✓	✓	✓	✓	✓	✓
User & Resource as a Target				(partial)	✓	✓
Outgoing/Incoming Action Policy				(partial)	✓	✓
Relationship Composition						
Relationship Depth	0 to 2	0 to n	1 to n	1 to n	0 to n	0 to n
Relationship Composition	f, f of f	exact type sequence	path of same type	exact type sequence	path pattern of different types	path pattern of different types, hopcount skipping

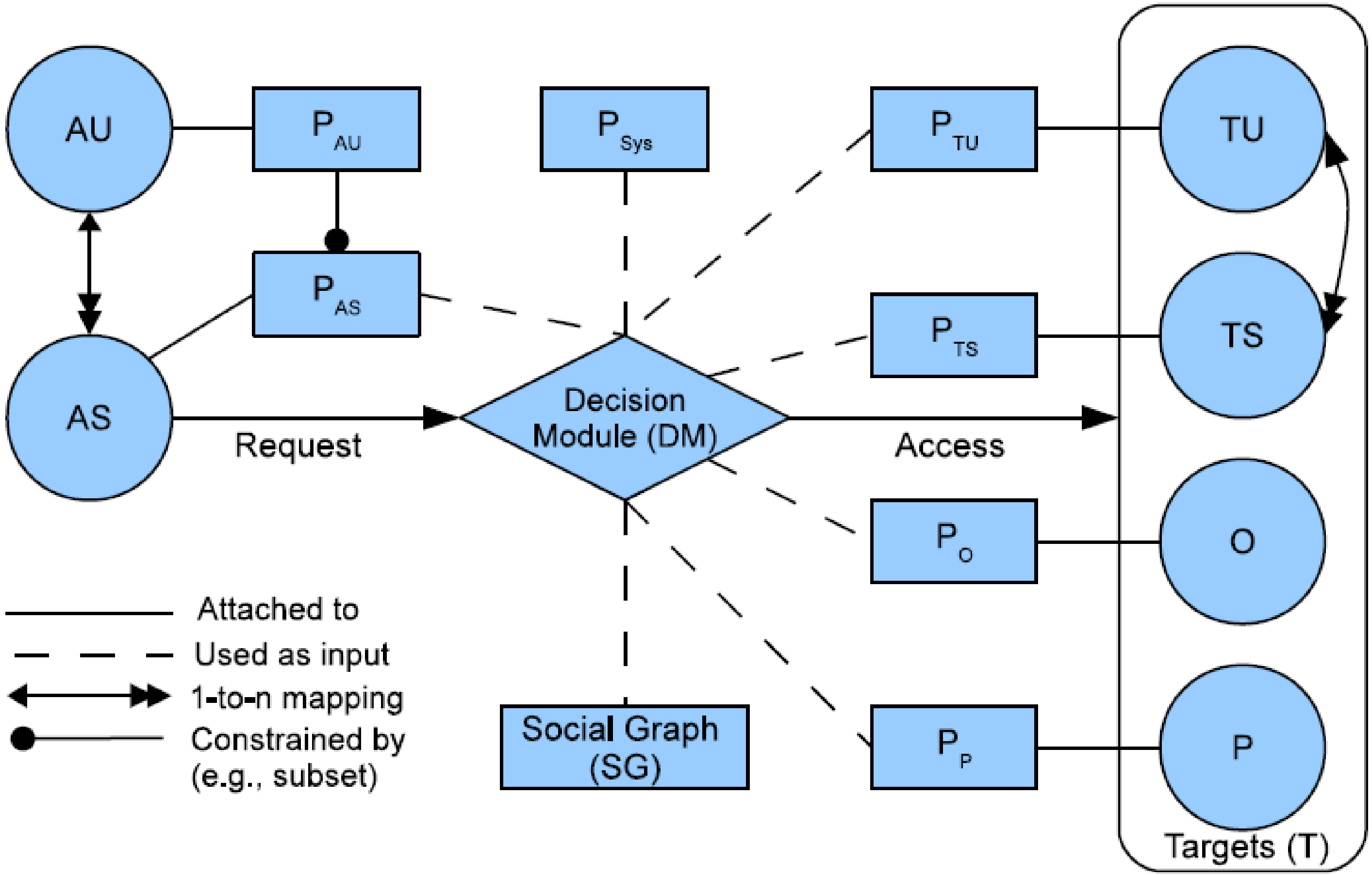
- The advantages of URRAC:
  - Path pattern of different relationship types and hopcount skipping make policy specification more expressive
  - System-level conflict resolution policy

UURAC Model Components



U_A: Accessing User
U_T: Target User
U_C: Controlling User
R_T: Target Resource
AUP: Accessing User Policy
TUP: Target User Policy
TRP: Target Resource Policy
SP: System Policy

URRAC Model Components



AU: Accessing User
AS: Accessing Session
TU: Target User
TS: Target Session
O: Object
P: Policy
P_{AU}: Accessing User Policy
P_{AS}: Accessing Session Policy
P_{TU}: Target User Policy
P_{TS}: Target Session Policy
P_O: Object Policy
P_P: Policy for Policy
P_{Sys}: System Policy

Action and Access Request

- $ACT = \{act_1, act_2, . . ., act_n\}$ is the set of OSN supported actions
- Access Request <s, act, T>
  - s tries to perform *act* on T
  - Target $T \subseteq (2^{TU \cup R} - \emptyset)$ is a non-empty set of users and resources
    - T may contain multiple targets

Authorization Policy

Accessing User Policy	$< act, graphrule >$
Accessing Session Policy	$< act, graphrule >$
Target User Policy	$< act^{-1}, graphrule >$
Target Session Policy	$< act^{-1}, graphrule >$
Object Policy	$< act^{-1}, graphrule >$
Policy for Policy	$< act^{-1}, graphrule >$
System Policy for User	$< act, graphrule >$
System Policy for Resource	$< act, o.type, graphrule >$ where $o.type$ is optional

- *action^{-1}* in TUP, TSP, OP and PP is the passive form since it applies to the recipient of action
- SP does not differentiate the active and passive forms
- SP for resource needs *o.type* to refine the scope of the resource

Hopcount Skipping

- Six degrees of separation
  - Any pair of persons are distanced by about 6 people on average. (4.74 shown by recent study)
  - Hopcount for U2U relationships is practically small
- U2R and R2R relationships may form a long sequence
  - Omit the distance created by resources
  - Local hopcount stated inside "[[]]" will not be counted in global hopcount.
  - E.g., "([*f**,3][[*c**, 2]],3)", the local hopcount 2 for *c** does not apply to the global hopcount 3, thus allowing *f** to have up to 3 hops.

Policy Conflict Resolution

- System-defined conflict resolution for potential conflicts among user-specified policies
- Disjunctive, conjunctive and prioritized order between relationship types
  - ∧,∨, > represent disjunction, conjunction and precedence
  - @ is a special relationship “null’’ that denotes “self”

Policy Conflict Resolution (cont.)

$$< read^{-1}, (own \wedge tag) >$$

The more rigid one between the owner's and the tagged users' "$read^{-1}$" policies over the photo is honored.

$$< friend\_request, (parent > @) >$$

When child attempts friendship request to someone, parents' policies get precedence over child's own will.

$$< share^{-1}, (own \vee tag \vee share) >$$

A weblink is sharable if either the original owner, or any of the tagged users or shared users allows.

RAC Models

U2U, U2R, R2R with attributes	**$URRAC_A$**	**Cheng et al 2014**
U2U, U2R, R2R	**URRAC**	**Cheng et al 2012-09**
U2U	**UURAC**	**Cheng et al 2012-07** **Cheng et al 2015**

URRAC$_A$ Motivation

- ReBAC usually relies on type, depth, or strength of relationships, but cannot express more complicated topological information
- ReBAC lacks support for attributes of users, resources, and relationships
- Useful examples include common friends, duration of friendship, minimum age, etc.

Attributes in OSNs

- Node attributes
  - Define user's identity and characteristics: e.g., name, age, gender, etc.
- Edge attributes
  - Describe the characteristics of the relationship: e.g., weight, type, duration, etc.
- Count attributes
  - Depict the occurrence requirements for the attribute-based path specification, specifying the lower bound of the occurrence of such path

RAC Models

U2U, U2R, R2R with attributes	**$URRAC_A$**	**Cheng et al 2014**
U2U, U2R, R2R	**URRAC**	**Cheng et al 2012-09**
U2U	**UURAC**	**Cheng et al 2012-07 Cheng et al 2015**