Object-to-Object Relationship Based Access Control: Model and Multi-Cloud Demonstration

by

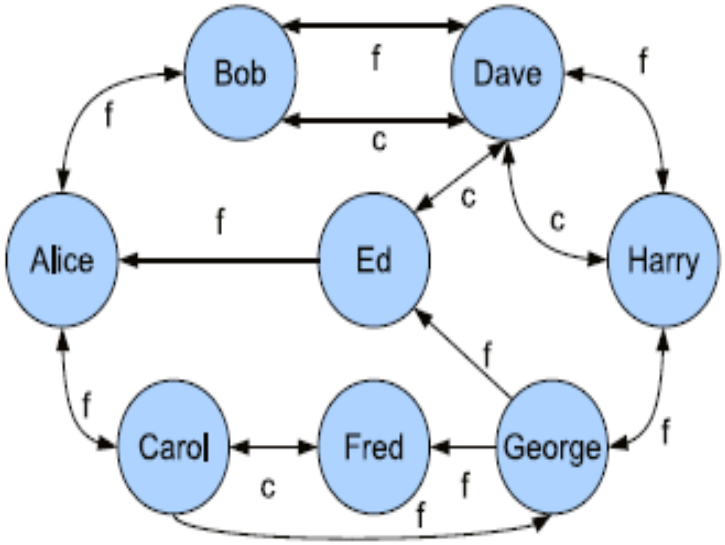
Tahmina Ahmed, Farhan Patwa and Ravi Sandhu
Department of Computer Science
University of Texas at San Antonio

7/29/2016

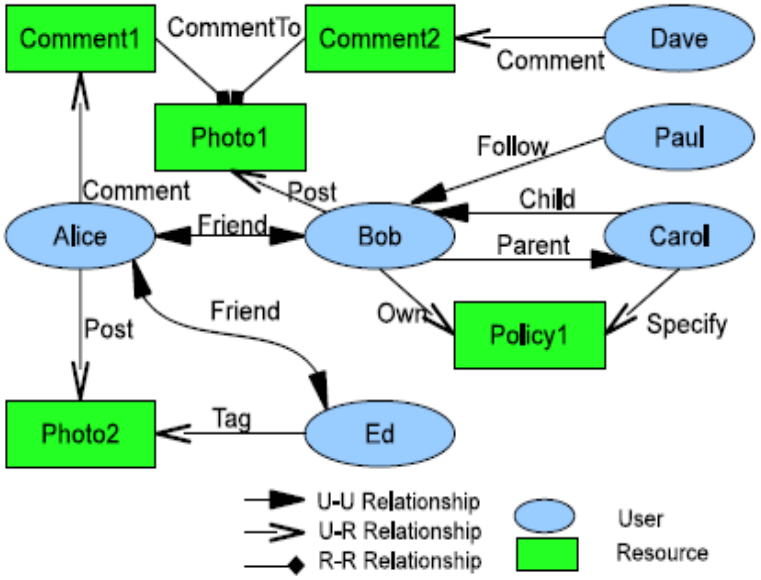
Outline

- Introduction
- Background and Motivation
  - Relationship in OSN
  - ReBAC beyond OSN
  - Existence of Object Relationship independent of user
  - Limitations of Existing ReBAC Models
- Model Characteristics
- OOReBAC Model
- OOReBAC: Application
- Implementation

Relationships in OSN

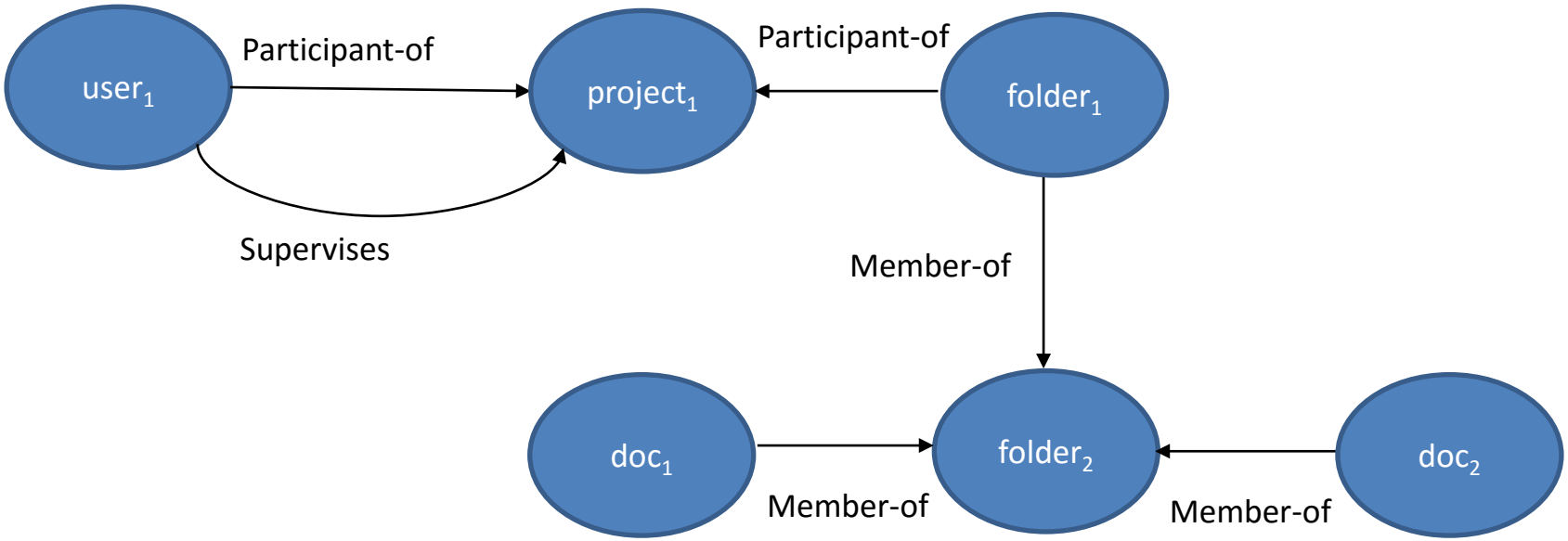


User to user relationships in a sample social graph [UURAC, Cheng et al. 2012]



User to user, user to resource and resource to resource relationships in a sample social graph [URRAC, Cheng et al. 2012]

ReBAC in General Computing System beyond OSN

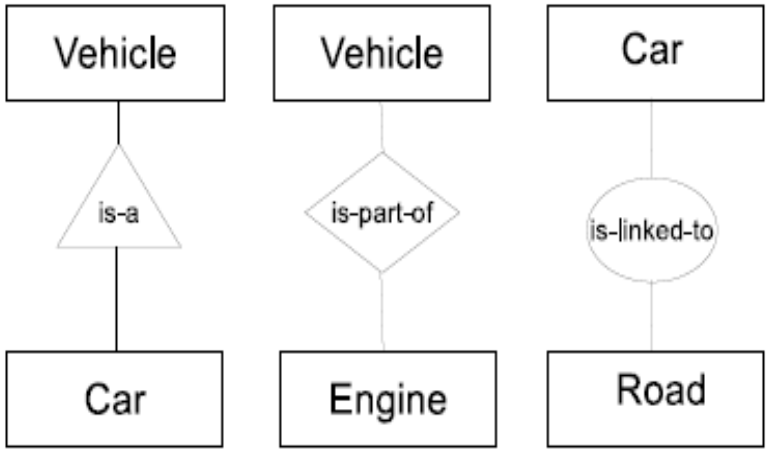


A sample Relationship Graph for Organizational Environment [RPPM, Crampton et al. ,2014]

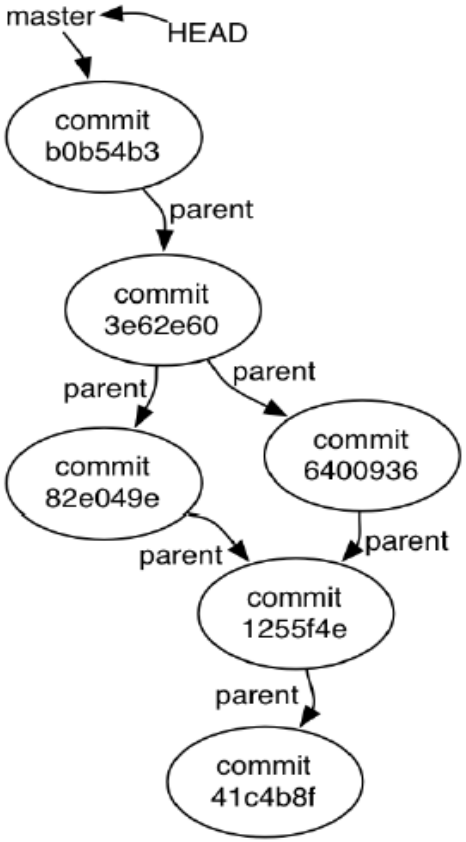
Object Relationships in Existing ReBAC

- Most of the ReBAC for OSN considers only user to user relationship
- OSN has very specific types of resources – photos, notes, comments. Which only makes sense along with users.
- Even though some ReBAC models consider general computing system they still need users/subjects existence in relationship graph.

Existence of Object Relationship Independent of User



Object Relationship in Object –Oriented System (Inheritance, Composition and Association)



History of a Git Project (Version Control System) is a DAG

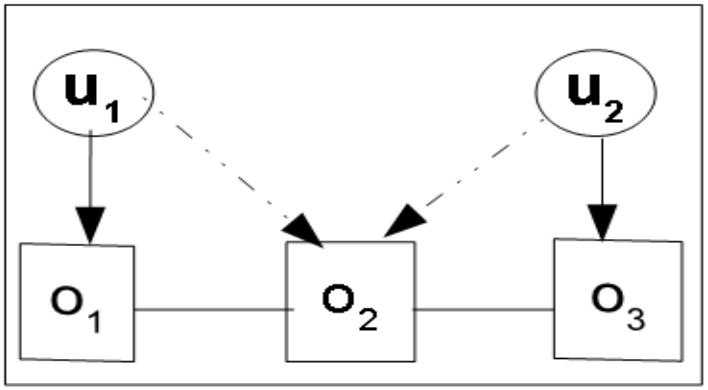
Limitations of Existing ReBAC Models

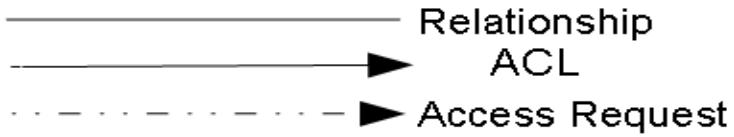
- Cannot configure relationship between objects independent of user.
- Cannot express authorization policy solely considering object relationship.

Considering these limitations we are proposing an object-to-object relationship based access control model.

How the model would look like?

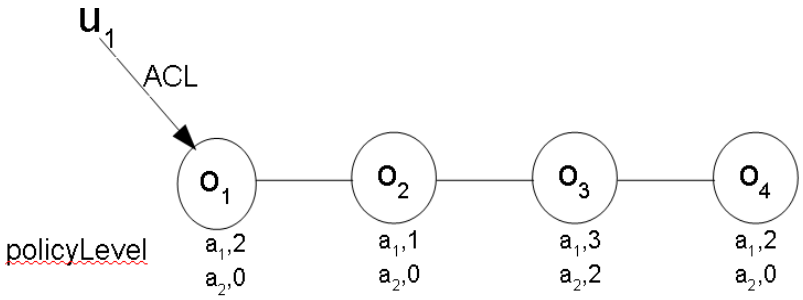
An Object to Object Relationship Based Access Control





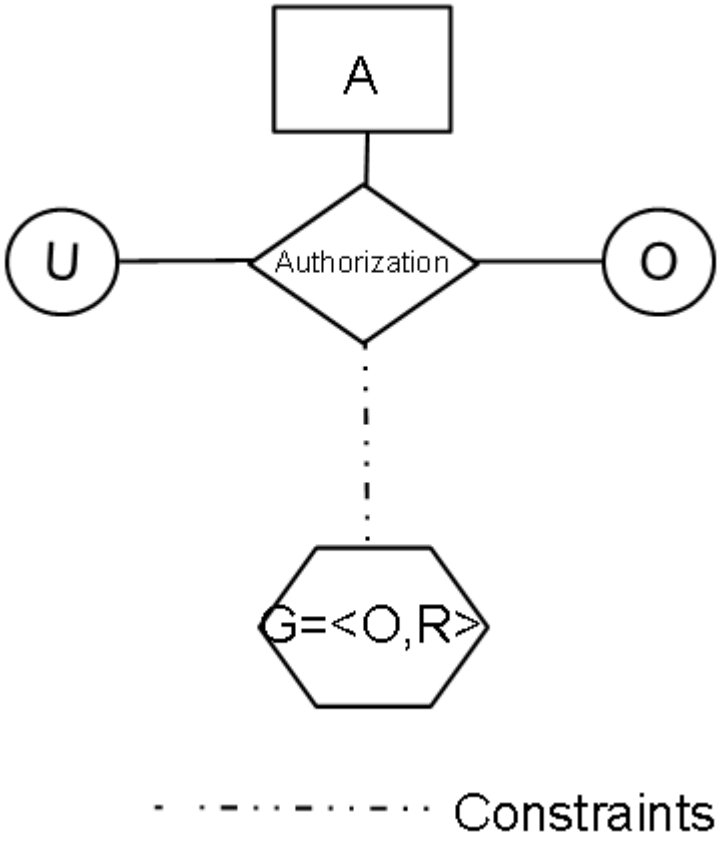
$ACL(o_1) = \{u_1\}$
$ACL(o_2) = \{\}$
$ACL(o_3) = \{u_2\}$

Policy Level Example



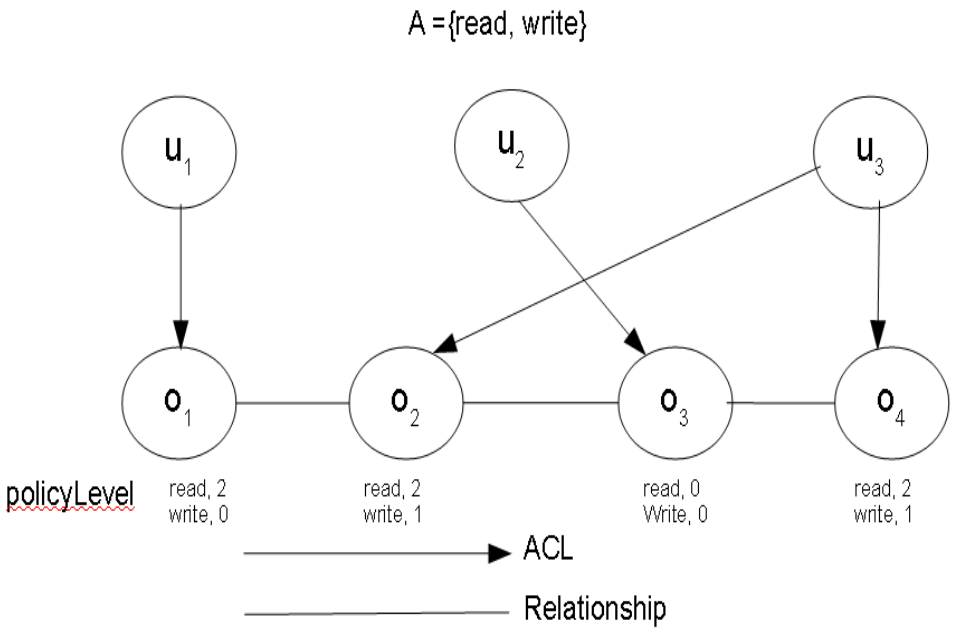
$policyLevel(a_1, o_1) = 2$
$policyLevel(a_2, o_1) = 0$
$policyLevel(a_1, o_2) = 1$
$policyLevel(a_2, o_2) = 0$
$policyLevel(a_1, o_3) = 3$
$policyLevel(a_2, o_3) = 2$
$policyLevel(a_1, o_4) = 2$
$policyLevel(a_2, o_4) = 0$

OOReBAC: Model Components and Definition



- U is a set of users
- O is a set of objects
- $R \subseteq \{z \mid z \subset O \wedge |z| = 2\}$
- $G=\langle O, R\rangle$ is an undirected relationship graph with vertices O and edges R
- A is a set of actions
- $P^p(o_1) = \{ o_2 \mid$ there exists a simple path of length p in graph G from o_1 to $o_2 \}$
- policyLevel: $O \times A \rightarrow \mathbb{N}$
- ACL: $O \rightarrow 2^U$ which returns the Access control List of a particular object.
- There is a single policy configuration point. Authorization Policy. for each action $a \in A$, $Authz_a(u{:}U, o{:}O)$ is a boolean function which returns true or false and u and o are formal parameters.
- Authorization Policy Language:
  Each action "a" has a single authorization policy $Authz_a(u{:}U, o{:}O)$ specified using the following language.
  $\phi := u \in PATH_i$
  $PATH_i := ACL(P^0(o)) \cup \ldots \cup ACL(P^i(o))$ where $i = \min(|O| - 1, policyLevel(a,o))$
  where for any set X, $ACL(X) = \bigcup_{x \in X} ACL(x)$

OOReBAC: An Example



Sequence of operations and its outcome:

- $U = \{u_1, u_2, u_3\}$
- $O = \{o_1, o_2, o_3, o_4\}$
- $R = \{\{o_1, o_2\}, \{o_2, o_3\}, \{o_3, o_4\}\}$
- $ACL(o_1) = \{u_1\}$
  $ACL(o_2) = \{u_3\}$
  $ACL(o_3) = \{u_2\}$
  $ACL(o_4) = \{u_3\}$
- $policyLevel(read, o_1) = 2$
  $policyLevel(write, o_1) = 0$
  $policyLevel(read, o_2) = 2$
  $policyLevel(write, o_2) = 1$
  $policyLevel(read, o_3) = 0$
  $policyLevel(write, o_3) = 0$
  $policyLevel(read, o_4) = 2$
  $policyLevel(write, o_4) = 1$

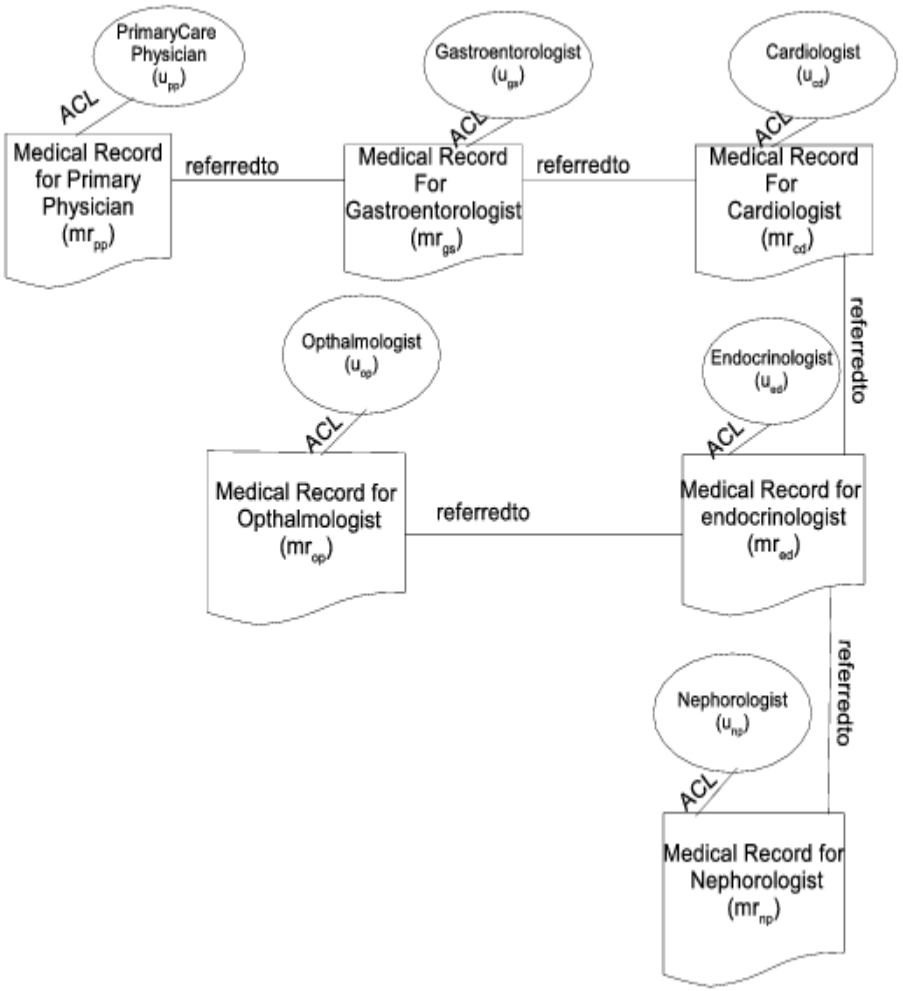
Configuration:

- $A = \{read, write\}$
- $Authz_{read}(u{:}U, o{:}O) \equiv u \in P^{policyLevel(read,o)}$
- $Authz_{write}(u{:}U, o{:}O) \equiv u \in P^{policyLevel(write,o)}$

Sequence of operations and its outcome:

- $read(u_1, o_3)$, $write(u_1, o_3)$ are denied
- $read(u_2, o_1)$ is allowed, $write(u_2, o_1)$ is denied
- $read(u_1, o_4)$, $write(u_1, o_4)$ are denied

OOReBAC: Application



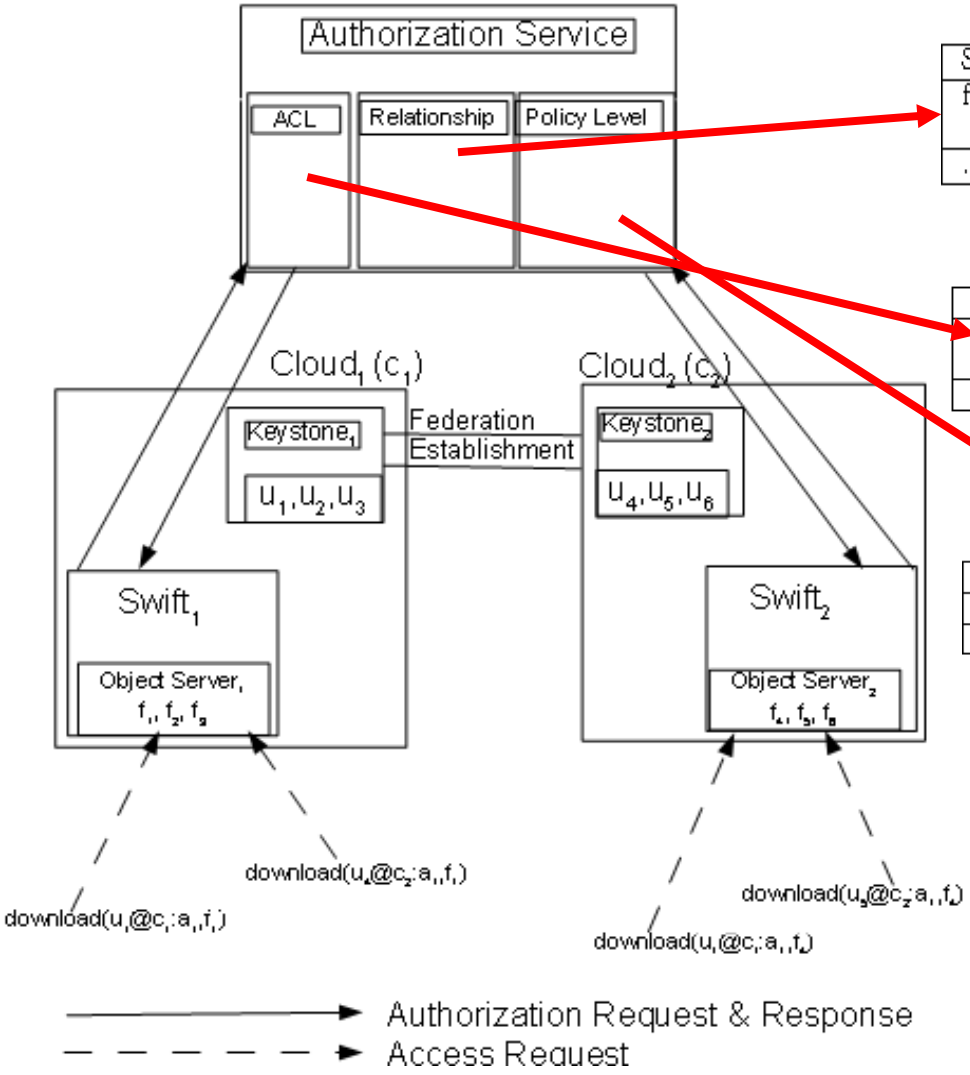
An OOReBAC Instantiation

- $U = \{ u_{pp}, u_{gs}, u_{cd}, u_{op}, u_{ed}, u_{np} \}$
- $O = \{ mr_{pp}, mr_{gs}, mr_{cd}, mr_{op}, mr_{ed}, mr_{np} \}$
- $R = \{\{mr_{pp}, mr_{gs}\}, \{mr_{gs}, mr_{cd}\}, \{mr_{cd}, mr_{ed}\}, \{mr_{op} mr_{ed}\},\{mr_{np}, mr_{ed}\}\}\}$
- $ACL(mr_{pp}) = \{u_{pp}\}$, $ACL(mr_{gs}) = \{u_{gs}\}$, $ACL(mr_{cd}) = \{u_{cd}\}$, $ACL(mr_{op}) = \{u_{op}\}$, $ACL(mr_{ed}) = \{u_{ed}\}$, $ACL(mr_{np}) = \{u_{np}\}$
- Action $=\{read, write\}$
- $policyLevel(read,mr_{pp})=\infty$, $policyLevel(write,mr_{pp})=0$, $policyLevel(read,mr_{gs})=\infty$, $policyLevel(write,mr_{gs})=0$, $policyLevel(read,mr_{cd})=\infty$, $policyLevel(write,mr_{cd})=0$, $policyLevel(read,mr_{op})=\infty$, $policyLevel(write,mr_{op})=0$, $policyLevel(read,mr_{ed})=\infty$, $policyLevel(write,mr_{ed})=0$, $policyLevel(read,mr_{np})=\infty$, $policyLevel(write,mr_{np})=0$
- Authorization policy:
  $Authz_{read}(u,o) \equiv u \in P^{policyLevel(read,o)}$
  $Authz_{write}(u,o) \equiv u \in P^{policyLevel(write,o)}$

Sequence of Operations and Outcomes

1) $read(u_{np}, mr_{pp})$: authorized
2) $read(u_{cd}, mr_{np})$: authorized
3) $write(u_{np}, mr_{np})$: authorized
4) $write(u_{np}, mr_{pp})$: denied
5) $write(u_{np}, mr_{pp})$: denied

Implementation: Openstack Object Storage (Swift)



Relationship

SourceFileName	TargetFileList
f_1 @$cloud_1$:$account_1$:$container_1$	{f_2 @$cloud_1$:$account_1$:$container_1$, f_3 @$cloud_2$:$account_1$:$container_1$}
...	...

ACL

Filename	UserList
f_1 @$cloud_1$:$account_1$:$container_1$	{u_1 @$cloud_1$:$account_1$, u_2 @$cloud_1$:$account_1$}
...	...

PolicyLevel

FileName	Policy Level
f_3 @$cloud_1$:$account_2$:$container_3$	(download,2)
...	...

Implementation

Functional Specification:

Functions	Conditions	Updates
	Administrative Actions	
CreateRelationship $(u,filename_1,filename_2)$	$admin \in role(u) \wedge$ $cloud(filename_1) = cloud(u)$ $\wedge\ filename_1 \notin RelationshipSet(filename_2)$ $\wedge\ filename_2 \notin RelationshipSet(filename_1)$	$RelationshipSet(filename_1) \cup= \{filename_2\}$ $RelationshipSet(filename_2) \cup= \{filename_1\}$
DeleteRelationship $(u,filename_1,filename_2)$	$admin \in role(u) \wedge$ $cloud(filename_1) = cloud(u)$ $filename_1 \in RelationshipSet(filename_2)$ $\wedge\ filename_2 \in RelationshipSet(filename_1)$	$RelationshipSet(filename_1) \setminus= \{filename_2\}$ $RelationshipSet(filename_2) \setminus= \{filename_1\}$
IncludeAUserinACL $(u,filename_1,username_1)$	$Role(u) \in Admin \wedge$ $cloud(filename_1) = cloud(u) \wedge$ $username_1 \notin ACLSet(filename_1)$	$ACLSet(filename_1) \cup= \{username_1\}$
ExcludeAUserFromACL $(u,filename_1,username_1)$	$Role(u) \in Admin \wedge$ $cloud(filename_1) = cloud(u) \wedge$ $username_1 \in ACLSet(filename_2)$	$ACLSet(filename_1) \setminus= \{username_1\}$
ConfigurePolicyLevel $(u,filename,num)$	$Role(u) \in Admin \wedge$ $cloud(filename_1) = cloud(u)$ $num \leq \lvert O \rvert$	$PolicyLevel(filename) = num$
	Operational Command	
download $(u,filename_1)$	$u \in U$ $\wedge\ authorize(u,filename_1,G)$	allow user u to download file $filename_1$

Algorithm for Authorization

Algorithm 1 authorize(u,f,G)

```
if u in ACL(f) then
    return true
else
    policyLevel = policyLevel(f)
    for depth limited search upto min(policyLevel, |O| - 1)
    do
        if if any of the file's ACL contains u then
            return true
        end if
    end for
    return false
end if
```

Conclusion and Future Work

- OOReBAC is the first attempt towards using object relationship independent of user in authorization policy specification and can only do where single type symmetric relationship is used.
- Limitations of OOReBAC:
  - Version Control system uses asymmetric relationship.
  - Object oriented Programming needs multiple Type asymmetric relationships.

We need to extend this model to accommodate multiple type asymmetric relationships to configure version control and object oriented system.

Questions?