Attribute Based Access Control and Implementation in Infrastructure as a Service Cloud

Dissertation Defense

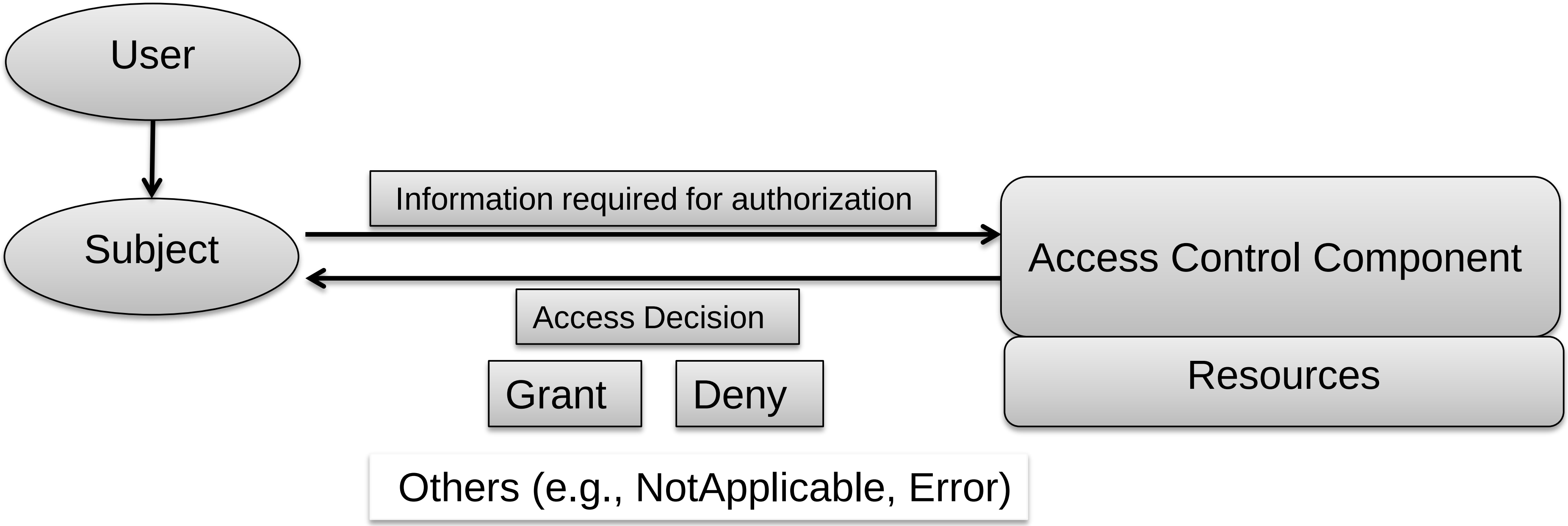
Xin Jin

Advisor: Dr. Ravi Sandhu
Co-Advisor: Dr. Ram Krishnan
Dr. Rajendra V. Boppana
Dr. Hugh Maynard
Dr. Jianwei Niu

Presentation Outline

- Introduction
- ABAC Operational Models
- ABAC Administrative Model
- ABAC In IaaS Cloud
- Conclusion

Access Control Scenario



User

Subject

Information required for authorization

Access Decision

Grant

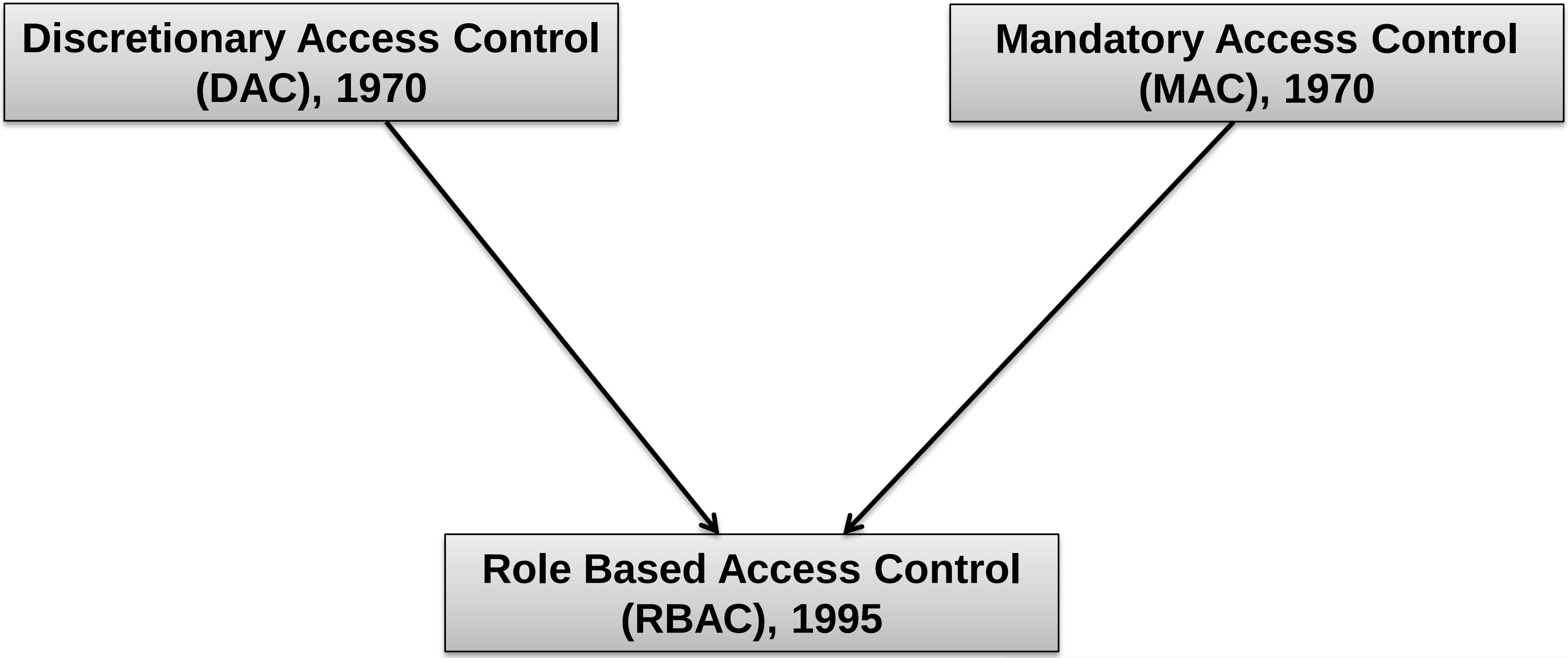
Deny

Others (e.g., NotApplicable, Error)

Access Control Component

Resources

Classical Access Control Models



Discretionary Access Control (DAC), 1970

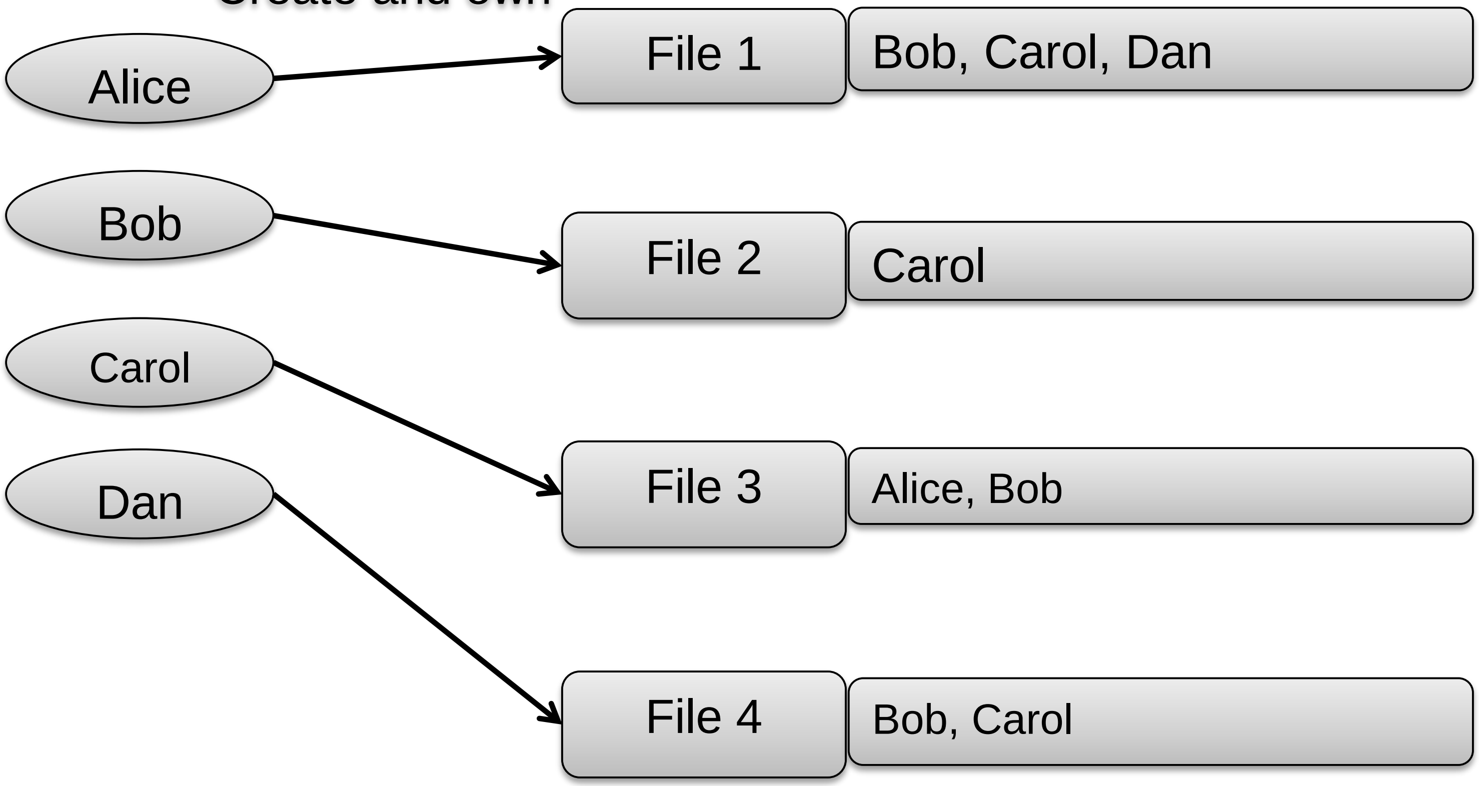
Mandatory Access Control (MAC), 1970

Role Based Access Control (RBAC), 1995

1. Policy Neutral
2. Administrative Convenience
3. Configure both DAC and MAC

Figure from http://profsandhu.com/miscppt/iri_130815.pptx

User Discretionary DAC



Create and own

The list of users who can read the file

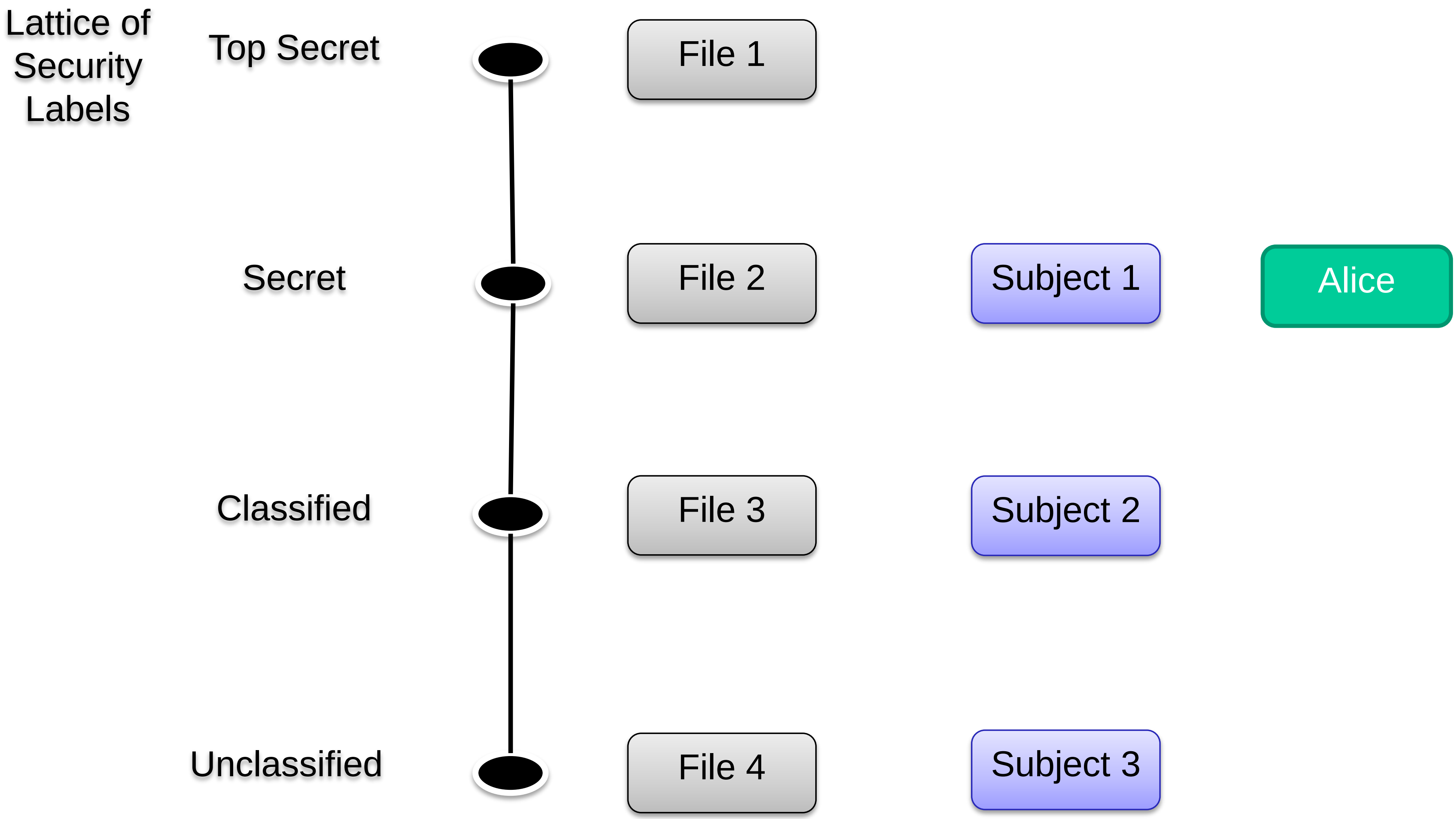
Alice → File 1: Bob, Carol, Dan

Bob → File 2: Carol

Carol → File 3: Alice, Bob

Dan → File 4: Bob, Carol

Mandatory Access Control (Lattice based Access Control)



Lattice of Security Labels

Top Secret

Secret

Classified

Unclassified

File 1

File 2

File 3

File 4

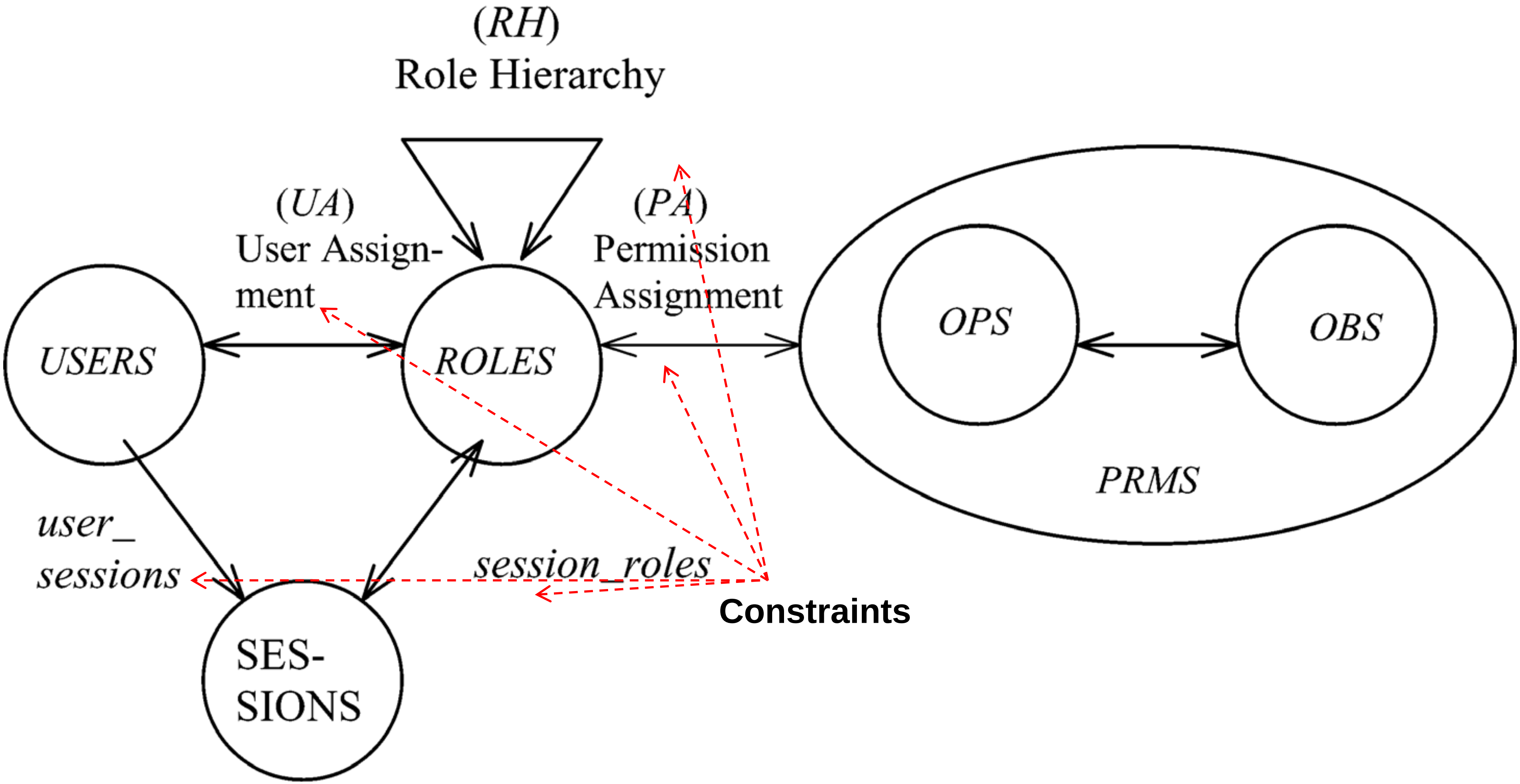
Subject 1

Subject 2

Subject 3

Alice

NIST-RBAC

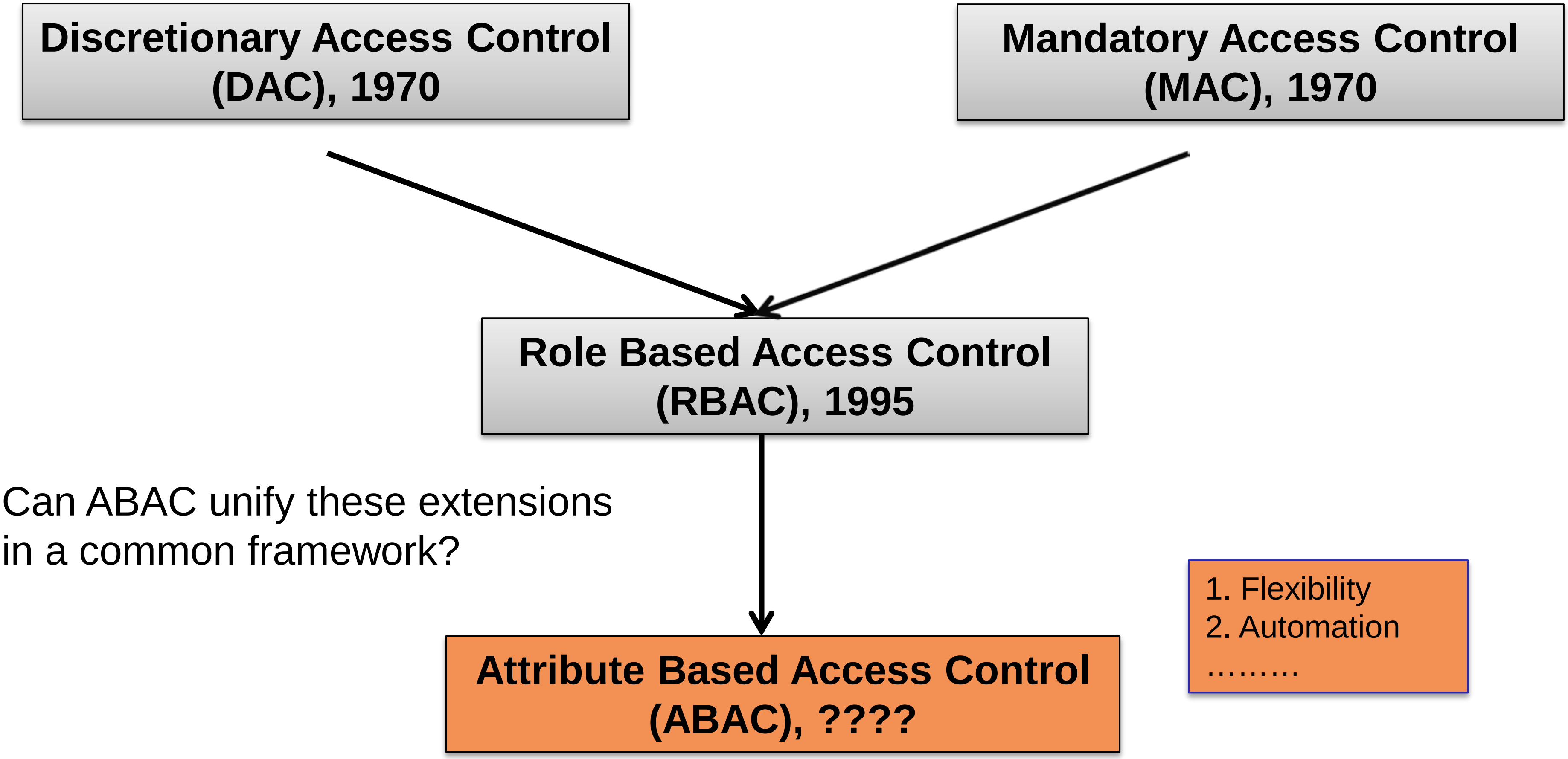

(RH)
Role Hierarchy
(UA)
User Assignment
(PA)
Permission Assignment
USERS
ROLES
OPS
OBS
PRMS
user_sessions
session_roles
SESSIONS
Constraints

RBAC Limitations

- Role explosion
  - Parameterized privileges, role templates, parameterized roles (1997-)
- Difficult role design and engineering
  - Role engineering top down or bottom up (1996-), and on role mining (2003-)
- Assignment of users/permissions to roles is cumbersome
  - Decentralized administration (1997-), attribute-based implicit user-role assignment (2002-), role-delegation (2000-), role-based trust management (2003-), attribute-based implicit permission-role assignment (2012-)
- Adjustment based on local/global situational factors is difficult
  - Temporal (2001-) and spatial (2005-) extensions to RBAC proposed
- RBAC does not offer an extension framework
  - Every shortcoming seems to need a custom extension

Slide from http://profsandhu.com/miscppt/iri_130815.pptx

Classical Access Control



Discretionary Access Control (DAC), 1970

Mandatory Access Control (MAC), 1970

Role Based Access Control (RBAC), 1995

Can ABAC unify these extensions in a common framework?

Attribute Based Access Control (ABAC), ????

1. Flexibility
2. Automation

………

General Idea of ABAC

- Attributes are **name** and **value** pairs
- Attributes are associated with different entities
  - User: *role, group, department, project, research_topic*
  - Subject: *clearance, role, admin, network*
  - Object: *sensitivity, date, owner, size, last_modified*
  - Context: *CPU usage, server_location, risk_level, time*
  - Attribute (i.e., meta-attribute): *risk_level_of_role, size_of_organization, head_of_department, trust_of_clearance*
- Converted by policies into rights just in time
  - Retrieve attributes related with each request: *(subject, object, operation)*

Related Work

- Formal Model
  - UCONABC (Park and Sandhu, 01): authorization, mutable attributes, continuous enforcement
  - Logical framework (Wang et al, 04): set-theory to model attributes
  - NIST ABAC draft (Hu et al, 13): enterprise enforcement

No distinguish between user and subject (classical models can not be configured)
No relationship of user, subject and object attributes.

- Policy Specification Language
  - SecPAL (Becker et al 03, 04), DYNPAL (Becker et al 09), Rule-based policy (Antoniou et al, 07), Binder (DeTreville 02) , EPAL1.2 (IBM, 03) , FAF (Jajodia et al 01)
- Enforcement Models
  - ABAC for web service (Yuan et al 06), PolicyMaker (Blaze et al 96)
- Implementations
  - XACML: authorization
  - SAML: pass attributes
  - OAuth: authorization

Focus on authorization and attribute release among organizations

Limited Policy Language

- Attribute Based Encryption
  - KP-ABE (Goyal et al 06), CP-ABE (Bethencourt et al 07)

Thesis Content

➢Problem Statement

- ➢ No widely agreed ABAC model that strictly distinguishes user and subject

➢Thesis Statement

- ➢ ABAC is suitable for flexible access control specification with reasonable complexity

Policy Specification	• ABAC-alpha model to unify DAC, MAC and RBAC • ABAC-beta model to cover operational RBAC models and extensions
Policy Administration	• Extend user-role assignment model to manage user-attribute assignment • Reachability analysis on policy
Policy Enforcement And Implementation	• Design ABAC model for access control in Infrastructure as a Service cloud • Implement it in OpenStack and evaluate cost

Presentation Outline

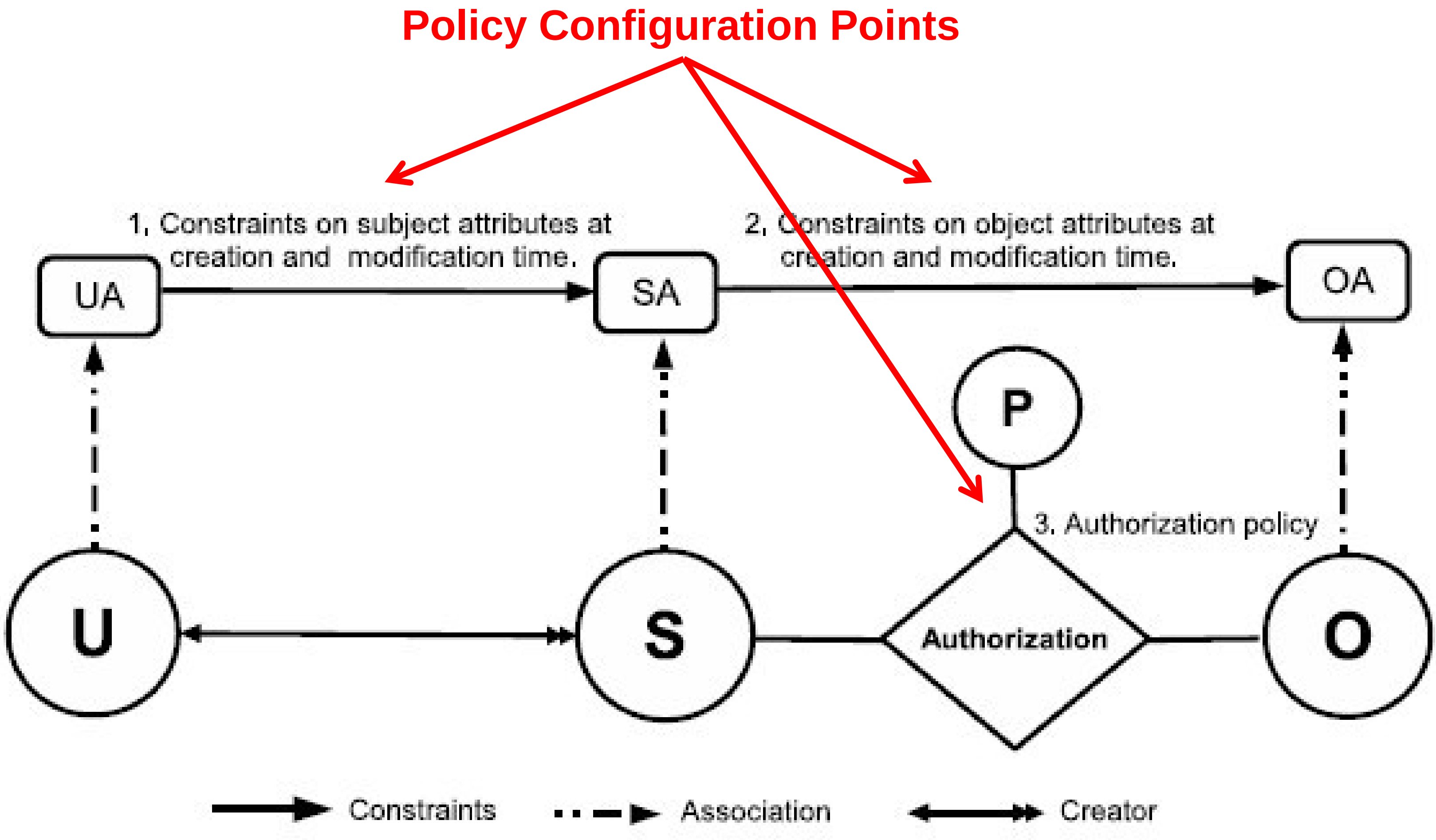
- Introduction
- ABAC Operational Models
- ABAC Administrative Model
- ABAC In IaaS Cloud
- Conclusion and Future Work

ABAC-alpha Model Features

- ABAC-alpha Cover DAC, MAC and RBAC
  - DAC: user-discretionary access control
  - MAC: LBAC with tranquility
  - RBAC: RBAC$_0$ and RBAC$_1$

	Subject attribute Value constrained by creating user ?	Object attribute value constrained by creating subject ?	Attribute range ordered?	Attribute function returns set value?	Object attribute modification?	Subject attribute modification by creating user?
DAC	YES	YES	NO	YES	YES	NO
MAC	YES	YES	YES	NO	NO	NO
RBAC0	YES	NA	NO	YES	NA	YES
RBAC1	YES	NA	YES	YES	NA	YES
ABAC-alpha	YES	YES	YES	YES	YES	YES

ABAC-alpha Model Structure


Policy Configuration Points
1, Constraints on subject attributes at creation and modification time.
2, Constraints on object attributes at creation and modification time.
UA
SA
OA
P
3. Authorization policy
U
S
Authorization
O
Constraints
Association
Creator

SubCreator as a distinguished subject attribute.

Example User Attributes

UA = {Clr, Dept, Proj, Skill}

Attribute	**Type**	**Scope**
Clr	atomic	unclassified, classified, secret, topsecret
Dept	atomic	software, hardware, finance, market
Proj	set	search, game, mobile, social, cloud
Skill	set	web, system, server, windows, security

Attributes assignment for Alice:

Clr(Alice) = classified
Dept(Alice) = finance
Proj(Alice) = {search, game, cloud}
Skill(Alice) = {web, server}

Policy Configuration Points

1. Authorization policies for each operation

$\text{Authorization}_{op}(s, o)$

2. Subject attribute assignment and modification constraints

ConstrSub(u, s, saset)

Exp: Set of Subject Attributes = {location, role, cls}

saset = {(location, CSConference), (role,{faculty, PhD}), (cls, classified)}

3. Object attribute constraints at object creation time

ConstrObj(s, o, oaset)

4. Object attribute constraints at object modification

ConstrObjMod(s, o, oaset)

Policy Configuration Languages

$$\varphi ::= \varphi \wedge \varphi \mid \varphi \vee \varphi \mid (\varphi) \mid \neg \varphi \mid \exists x \in set.\varphi \mid \forall x \in set.\varphi \mid set\ setcompare\ set \mid$$

$$atomic \in set \mid atomic\ atomiccompare\ atomic$$

$$setcompare ::= \subset \mid \subseteq \mid \not\subset$$

$$atomiccompare ::= < \mid = \mid \leq$$

➢ Authorization policy

- Attributes of the involved subject and object

➢ Subject attributes constraints

- User attributes and the proposed attributes for subjects

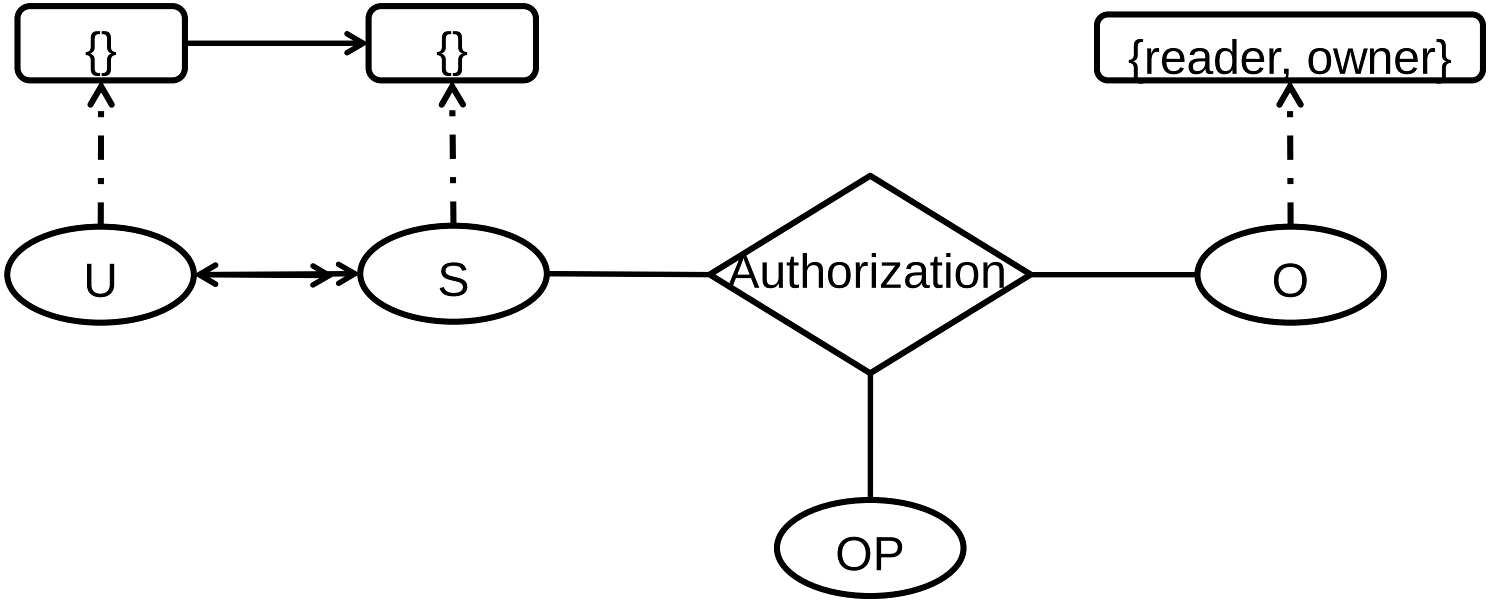
➢ Object attribute constraints at creation time

- Attributes of the subject and the proposed value of object

➢ Object attributes constraints at modification time

- Attributes of the subject and object and the proposed value of object

User Discretionary DAC



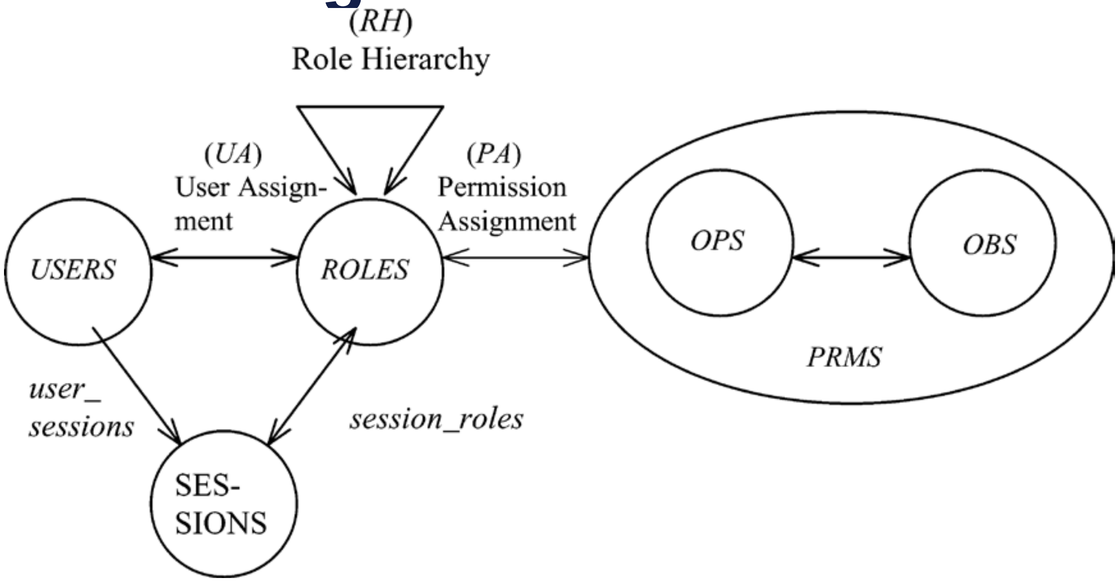
ConstrObj(s, o, {(owner, owner_name), (reader, {name1, name2})}):
owner_name = SubCreator(s)

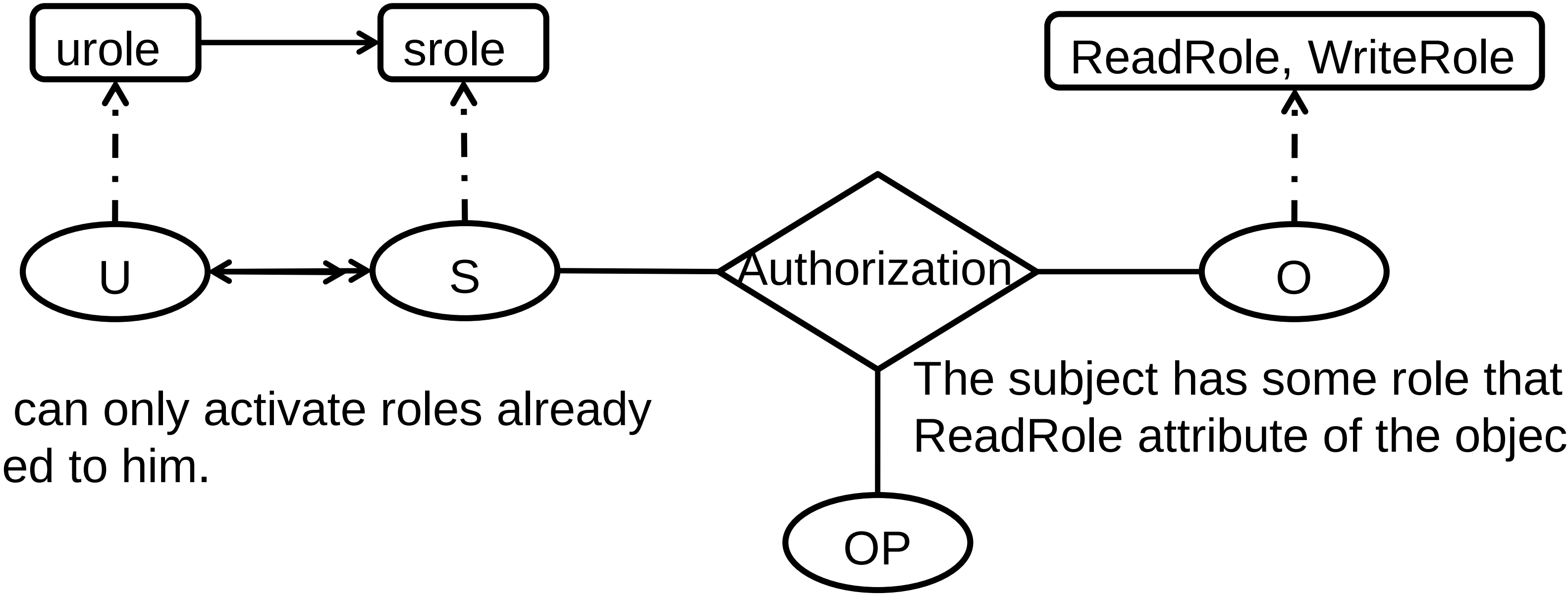
Example:
ConstrObj(Sub-Alice, docA, {(owner, Alice), (reader, {Bob, Carol})}):
Alice = SubCreator(Sub-Alice)

ConstrObj(Sub-Alice, docA, {(owner, Dan), (reader, {Bob, Carol})}):
Dan = SubCreator(Sub-Alice)

ConstrObjMod(s, o, {(owner, owner_name), (reader, {name1, name2})}):
owner(o) = SubCreator(s) and owner_name = SubCreator(s)

Configuration for RBAC

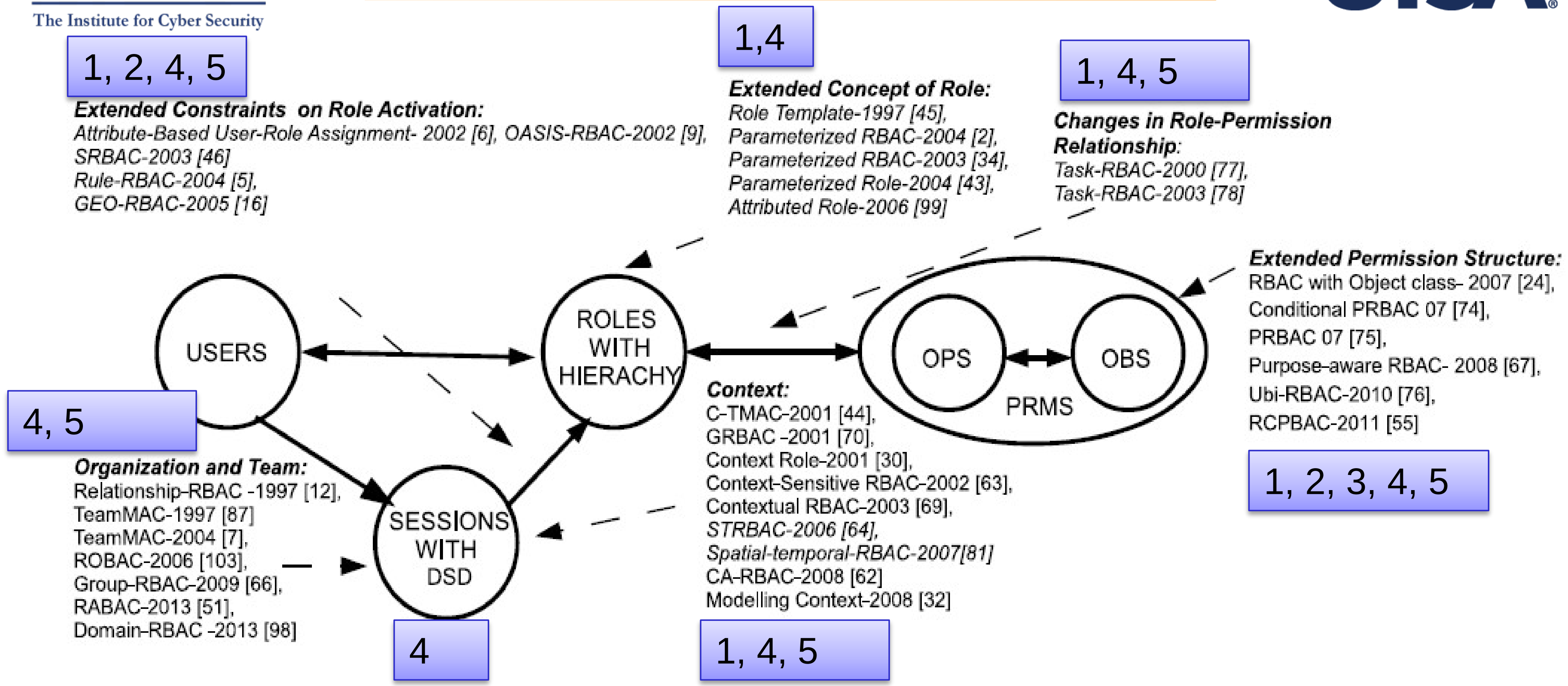




A user can only activate roles already assigned to him.

The subject has some role that is in the ReadRole attribute of the object.

ABAC-beta Scope



1, 2, 4, 5

Extended Constraints on Role Activation:
Attribute-Based User-Role Assignment- 2002 [6], OASIS-RBAC-2002 [9],
SRBAC-2003 [46]
Rule-RBAC-2004 [5],
GEO-RBAC-2005 [16]

1,4

Extended Concept of Role:
Role Template-1997 [45],
Parameterized RBAC-2004 [2],
Parameterized RBAC-2003 [34],
Parameterized Role-2004 [43],
Attributed Role-2006 [99]

1, 4, 5

Changes in Role-Permission Relationship:
Task-RBAC-2000 [77],
Task-RBAC-2003 [78]

Extended Permission Structure:
RBAC with Object class- 2007 [24],
Conditional PRBAC 07 [74],
PRBAC 07 [75],
Purpose-aware RBAC- 2008 [67],
Ubi-RBAC-2010 [76],
RCPBAC-2011 [55]

1, 2, 3, 4, 5

4, 5

Organization and Team:
Relationship-RBAC -1997 [12],
TeamMAC-1997 [87]
TeamMAC-2004 [7],
ROBAC-2006 [103],
Group-RBAC-2009 [66],
RABAC-2013 [51],
Domain-RBAC -2013 [98]

4

Context:
C-TMAC-2001 [44],
GRBAC -2001 [70],
Context Role-2001 [30],
Context-Sensitive RBAC-2002 [63],
Contextual RBAC-2003 [69],
STRBAC-2006 [64],
Spatial-temporal-RBAC-2007[81]
CA-RBAC-2008 [62]
Modelling Context-2008 [32]

1, 4, 5

1. Context Attributes

2. Subject attribute constraints policy are different at creation and modification time.

3. Subject attributes constrained by attributes of subjects created by the same user.

4. Policy Language

5. Meta-Attributes

Examples

1. Context Attributes

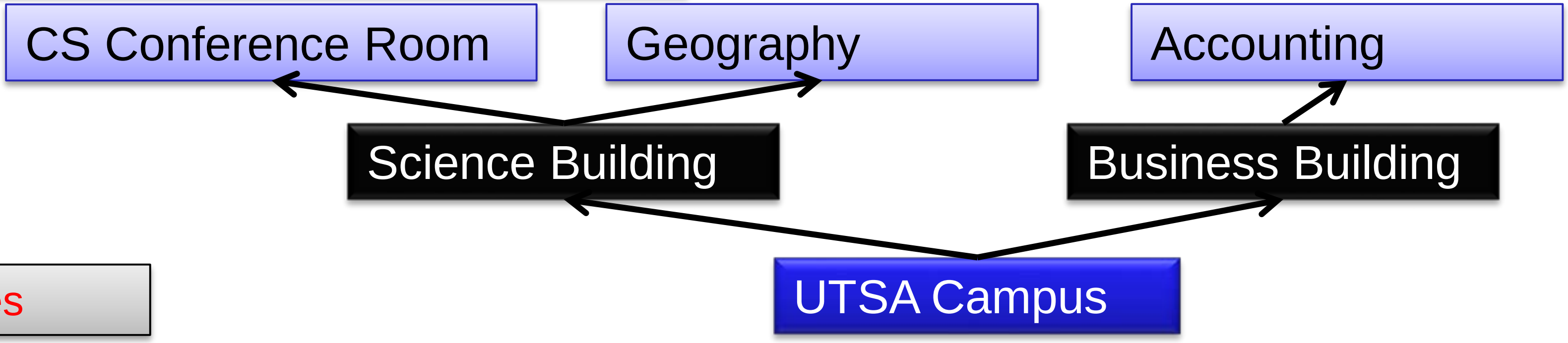
2. Subject attribute constraints policy are different at creation and modification time.

OASIS-RBAC

- Prerequisite role
- Initial role assignment constraints
- Other role assignment constraints

3. Subject attributes constraints by attributes of subjects created by the same user.

Ubi-RBAC



CS Conference Room

Geography

Accounting

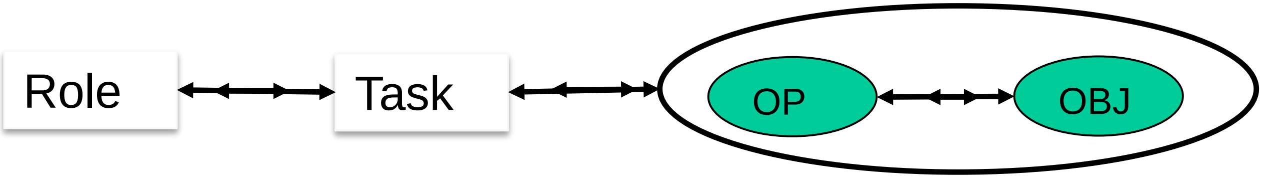
Science Building

Business Building

UTSA Campus

5. Meta-Attributes

Task-RBAC



Role

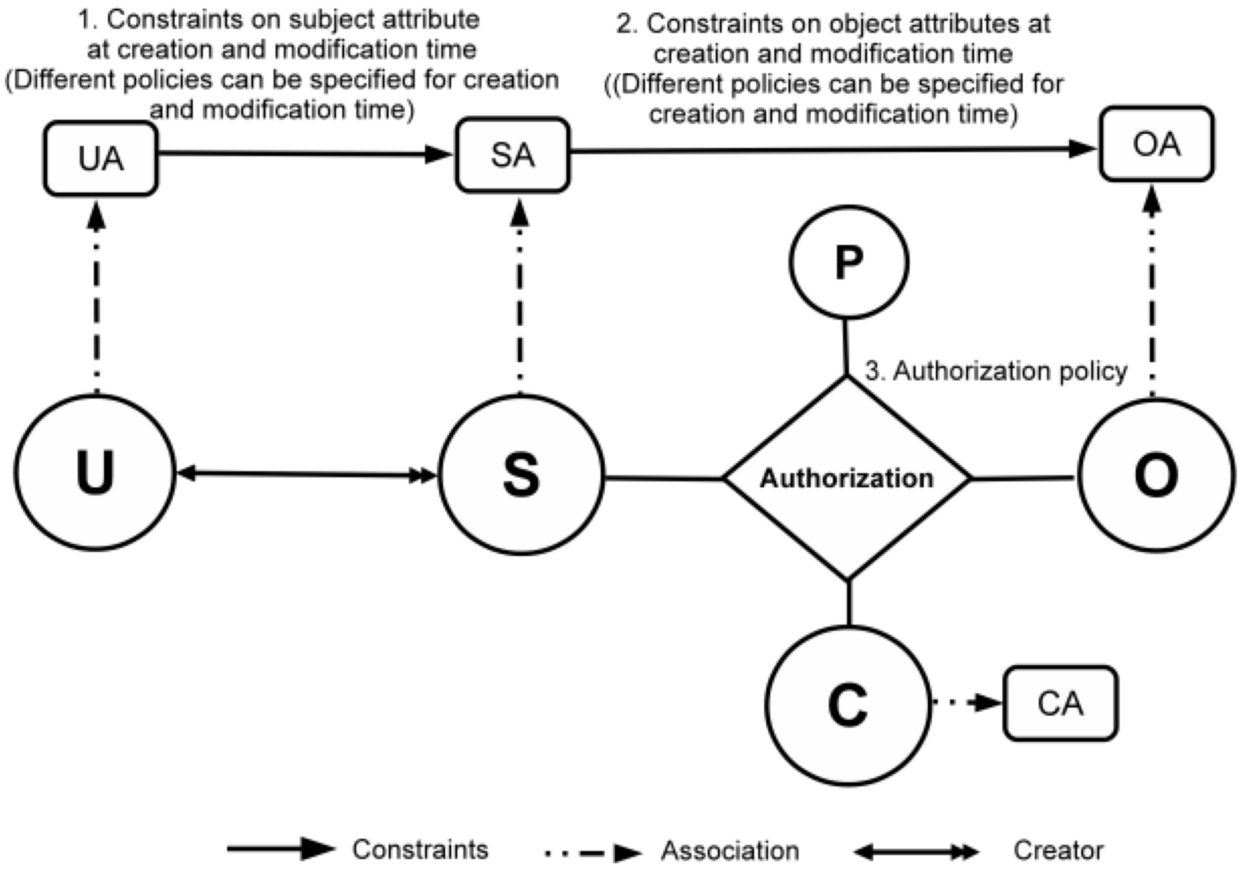
Task

OP

OBJ

task(r1) = {t1, t2}
readtask(o1) = {t1, t2, t3}
urole(u) = {r1, r2}

ABAC-beta Model


1. Constraints on subject attribute at creation and modification time (Different policies can be specified for creation and modification time)
2. Constraints on object attributes at creation and modification time ((Different policies can be specified for creation and modification time)
UA
SA
OA
P
3. Authorization policy
U
S
Authorization
O
C
CA
Constraints
Association
Creator

Summary of ABAC Models

- ABAC-alpha: “Least” features to configure DAC, MAC and RBAC
- ABAC-beta: extension of ABAC-alpha for the purpose of unifying operational RBAC and its extended models
- Future Work
  - Theoretical analysis of enforcement complexity, RBAC compared with ABAC instance of RBAC
  - Policy specification language. For example, to be able to detect misconfiguration, compliance with privacy expectation

Presentation Outline

- Introduction
- ABAC Operational Models
- ABAC Administrative Model
- ABAC In IaaS Cloud
- Conclusion and Future Work

GURA Model

- The generalized User-Role Assignment Model (GURA) deals with user-attribute administration.
  - It is an extension of URA component in ARBAC97

- Although subject and object are also associated with attributes, this mode is not suitable
  - Subject and object attributes are modified by regular users
  - This model is useful as long as this style of attribute administration is involved

- Advantage
  - Well-documented advantage of RBAC inherited

GURA Model

- Administrators request to modify attributes of users
  - add, delete, assign
- Policy
  - Administrative users with [administrative roles] can [modify] value [value] to [attribute name] attribute of a user if [condition]
- GURA$_0$

can_add$_{project}$ = { (manager, windows in project(u) and linux in project(u), security) }

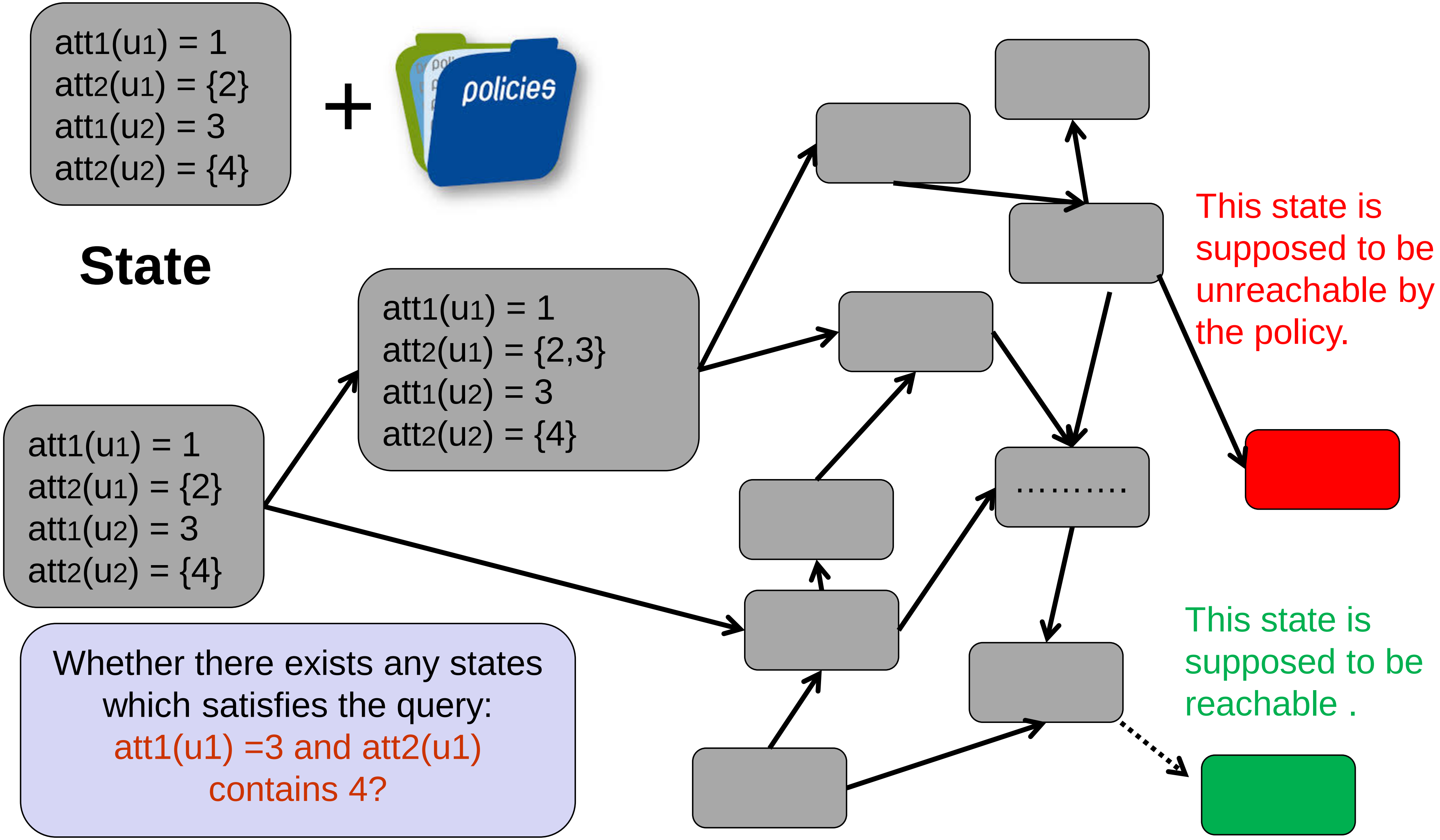
add(Alice, Bob, project, security) where adrole(Alice) = manager
add(Carol, Bob, project, security) where adrole(Carol) is not manager

- GURA$_1$

can_assign$_{approved}$ = { (director, true, {true, false}) }
can_add$_{project}$ = { (manager, windows in project(u) and linux in project(u) and clearance(u) > c and phd in degree(u) and approved(u)= true, security) }

assign(Alice, Bob , approved, true) where adrole(Alice) = director
assign(Carol, Alice, approved, true) where adrole(Carol) is not director

Attribute Reachability Problem



att1(u1) = 1
att2(u1) = {2}
att1(u2) = 3
att2(u2) = {4}

+

State

att1(u1) = 1
att2(u1) = {2}
att1(u2) = 3
att2(u2) = {4}

att1(u1) = 1
att2(u1) = {2,3}
att1(u2) = 3
att2(u2) = {4}

This state is supposed to be unreachable by the policy.

This state is supposed to be reachable .

Whether there exists any states which satisfies the query:
att1(u1) =3 and att2(u1) contains 4?

Reachability Query and Types

- We define two query types. RP-equal ($RP_{=}$) and RP-super ($RP_{\supseteq}$)

Query Type: RP-same

Clr(Alice) = classified
Proj(Alice) = {search, cloud}

Clr(Alice) = classified
Dept(Alice) = finance
Proj(Alice) = {search, cloud}
Skill(Alice) = {web, server}

Clr(Alice) = classified
Dept(Alice) = finance
Proj(Alice) = {search, game, cloud}
Skill(Alice) = {web, server}

Query Type: RP-super

Clr(Alice) = classified
Dept(Alice) = market
Proj(Alice) = {search, cloud}

Clr(Alice) = classified
Dept(Alice) = finance
Proj(Alice) = {search, cloud}
Skill(Alice) = {web, server}

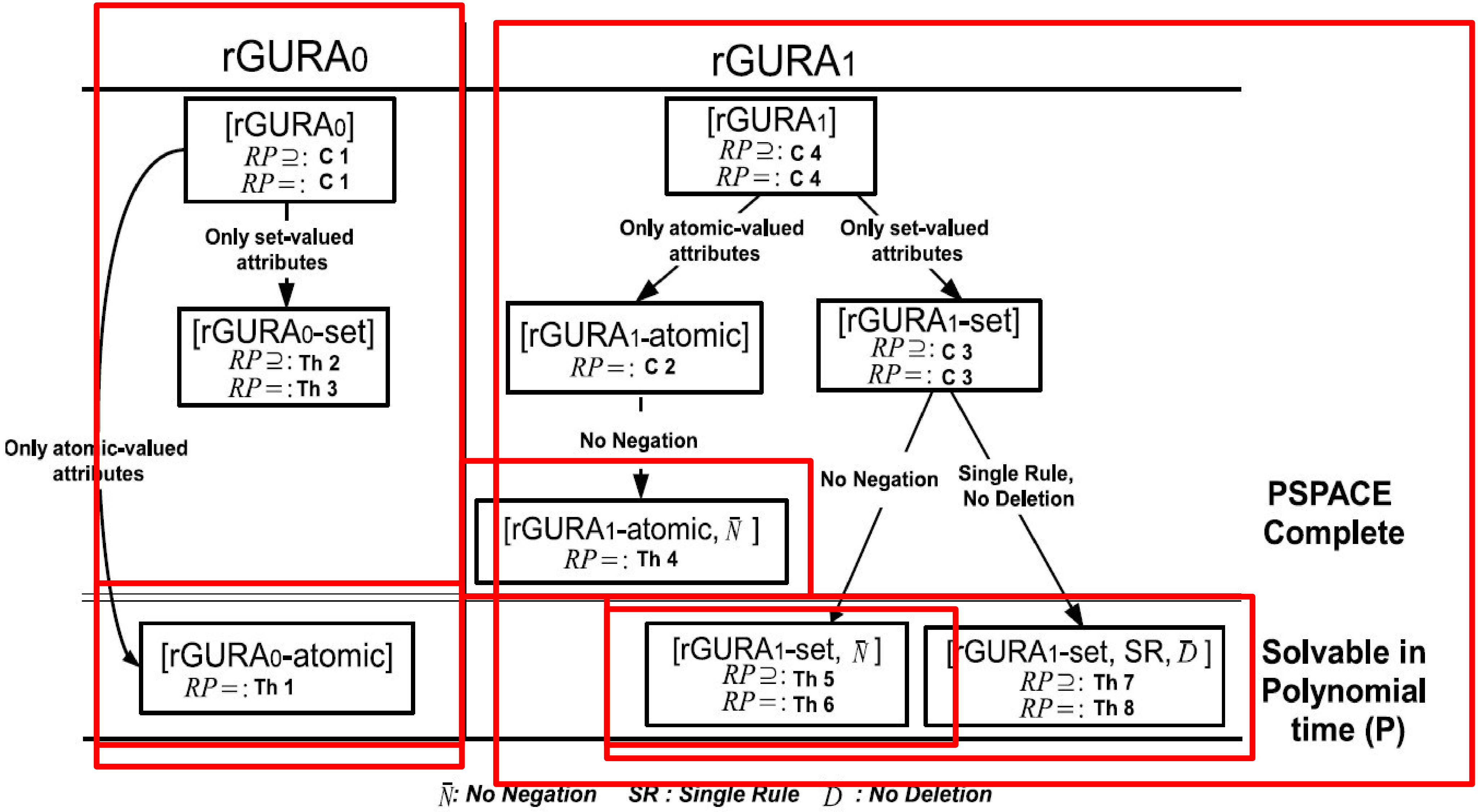
Clr(Alice) = classified
Dept(Alice) = market
Proj(Alice) = {search, game, cloud}
Skill(Alice) = {web, server}

- rGURA is different from GURA model only in the [condition] specification languages for administrative rules
  - Only conjunction and negation is allowed

$$\varphi ::= \neg\varphi \mid \varphi \wedge \varphi \mid \mathrm{aua(u)} = \mathrm{avalue}$$
$$\mathrm{avalue} ::= \mathrm{aval}_1 \mid \mathrm{aval}_2 \; \ldots \mid \mathrm{aval}_n$$

$$\varphi ::= \neg\varphi \mid \varphi \wedge \varphi \mid \mathrm{aua(u)} = \mathrm{avalue} \mid \mathrm{svalue} \in \mathrm{sua(u)}$$
$$\mathrm{avalue} ::= \mathrm{aval}_1 \mid \mathrm{aval}_2 \; \ldots \mid \mathrm{aval}_n$$
$$\mathrm{svalue} ::= \mathrm{sval}_1 \mid \mathrm{sval}_2 \; \ldots \mid \mathrm{sval}_n$$

Analysis Results Summary

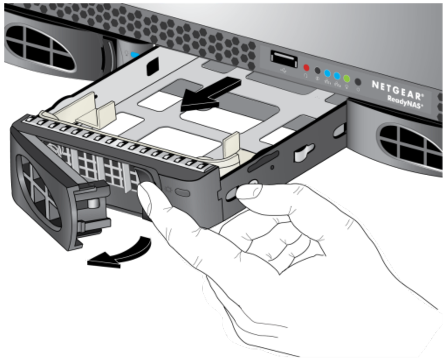


World-Leading Research with Real-World Impact!

Presentation Outline

- Introduction
- ABAC Operational Models
- ABAC Administrative Model
- ABAC In IaaS Cloud
- Conclusion and Future Work

Operations in Local Data Center



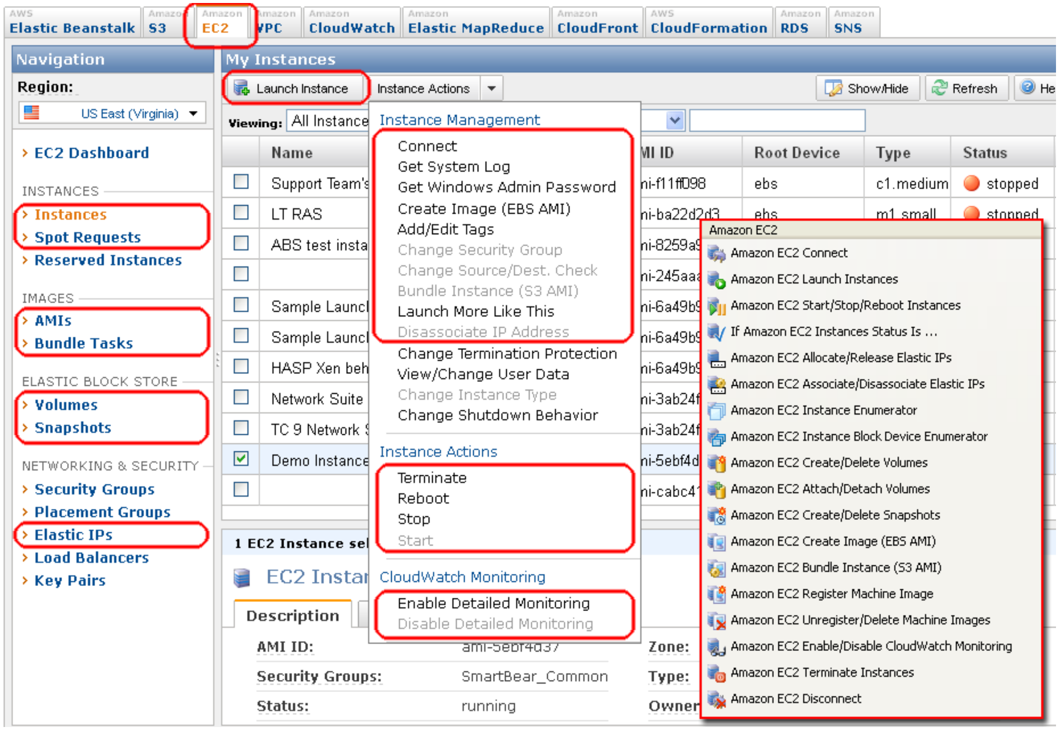
iOS

Access Control

Access Control in IaaS Cloud

Equivalent policy in physical world should be able to be configured using cloud access control service

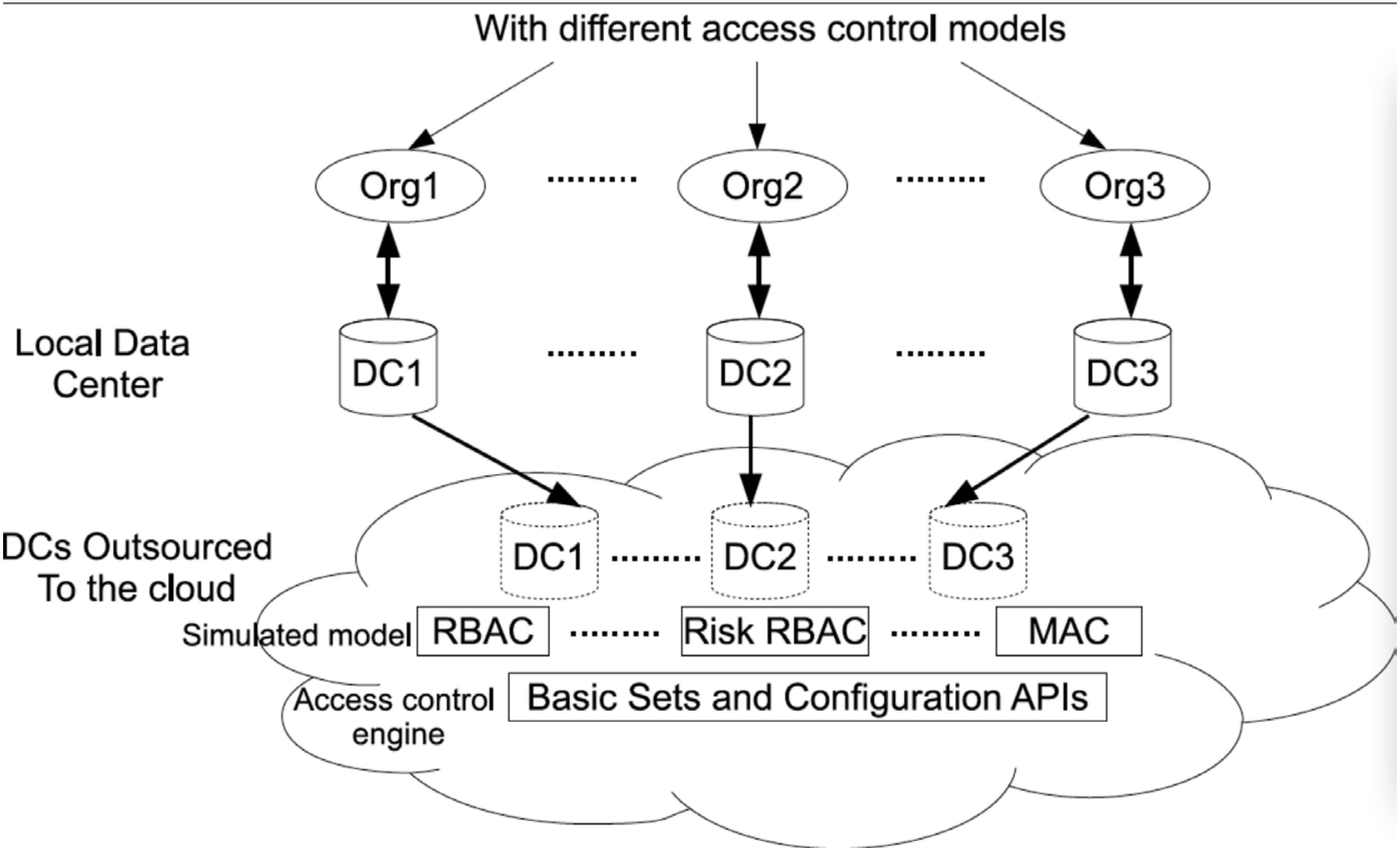
With virtualization, cloud may provide more fine-grained access control



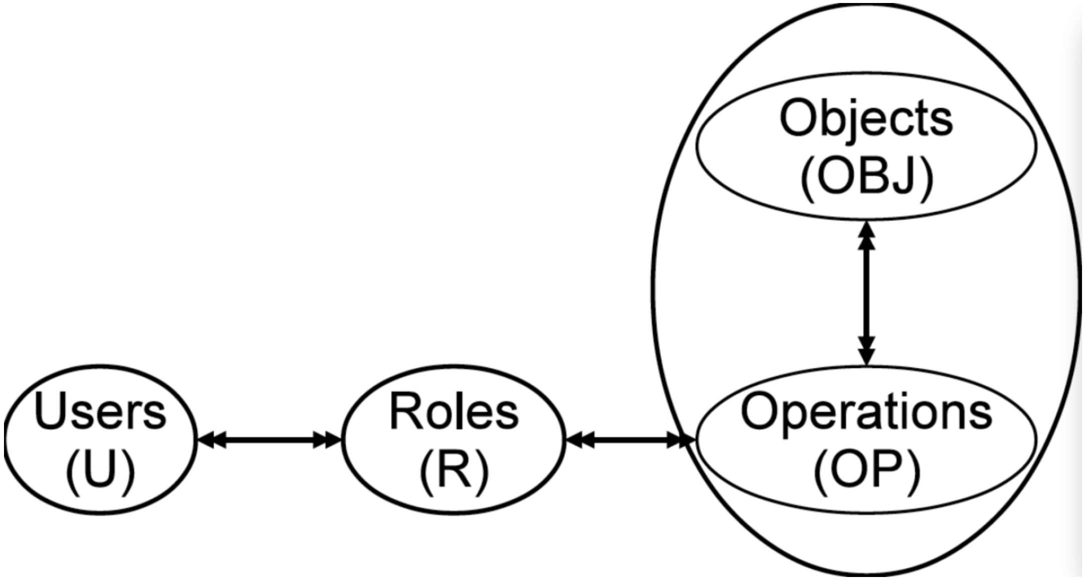
	Name	MI ID	Root Device	Type	Status
□	Support Team's	ni-f11ff098	ebs	c1.medium	● stopped
□	LT RAS	ni-ba22d2d3	ebs	m1.small	● stopped
□	ABS test insta	ni-8259a9			
□		ni-245aaa			
□	Sample Launcl	ni-6a49b9			
□	Sample Launcl	ni-6a49b9			
□	HASP Xen beh	ni-6a49b9			
□	Network Suite	ni-3ab24f			
□	TC 9 Network S	ni-3ab24f			
☑	Demo Instance	ni-5ebf4d			
□		ni-cabc41			

AMI ID: ami-5ebf4d37
Security Groups: SmartBear_Common
Status: running
Zone:
Type:
Owner

Problem

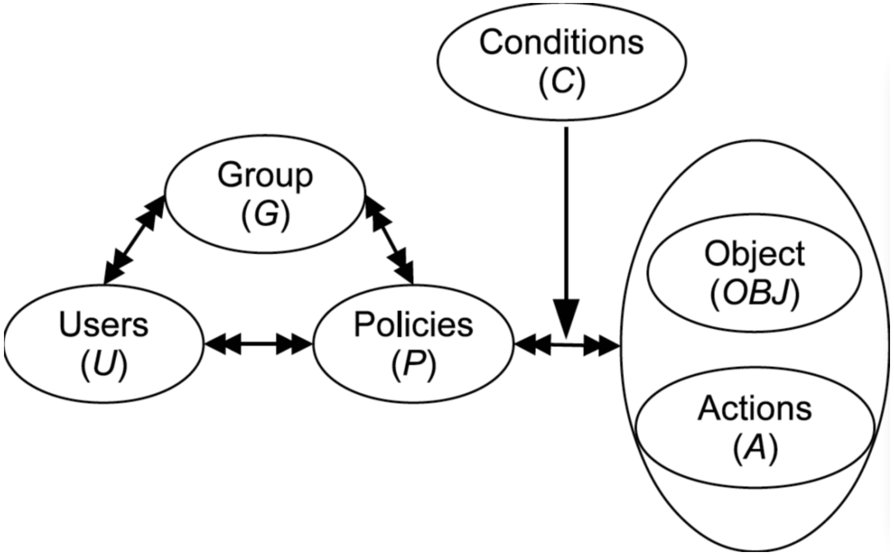

With different access control models
Org1
Org2
Org3
Local Data Center
DC1
DC2
DC3
DCs Outsourced To the cloud
DC1
DC2
DC3
Simulated model
RBAC
Risk RBAC
MAC
Access control engine
Basic Sets and Configuration APIs

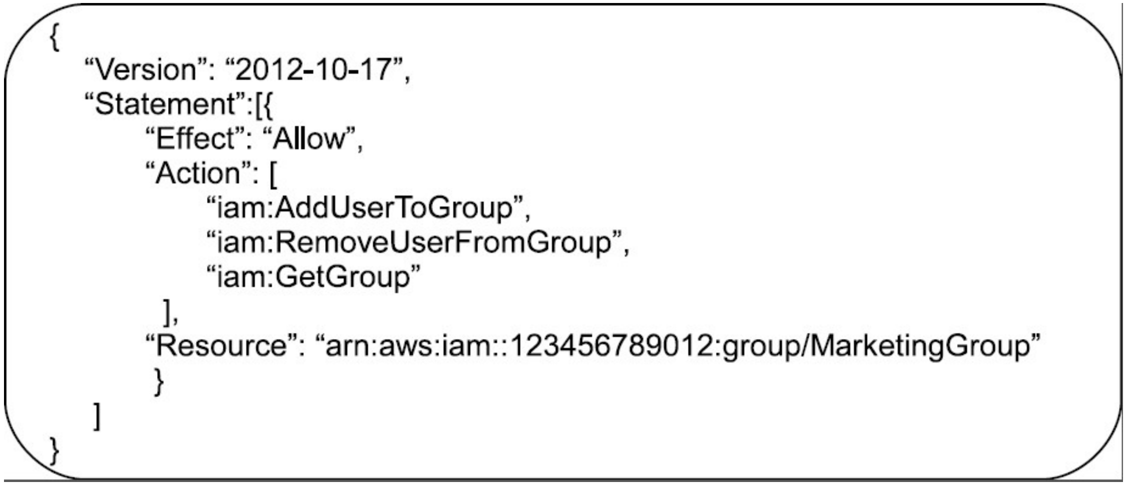
OpenStack (Grizzly Release)



- Limitations
  - Tenant can not configure their own policy, uses cloud role instead
  - Not able to configure tenant administrator
  - Access control on operation level, no control on object level
    - Give *identity:createUser* permission to role r1, then r1 can create users in any tenant
    - Give *nova:stop* permission to role r1, r1 can stop any machine in the tenant
  - Access control only based on role

AWS Access Control





```
{
    “Version”: “2012-10-17”,
    “Statement”:[{
          “Effect”: “Allow”,
          “Action”: [
                “iam:AddUserToGroup”,
                “iam:RemoveUserFromGroup”,
                “iam:GetGroup”
           ],
          “Resource”: “arn:aws:iam::123456789012:group/MarketingGroup”
           }
     ]
}
```

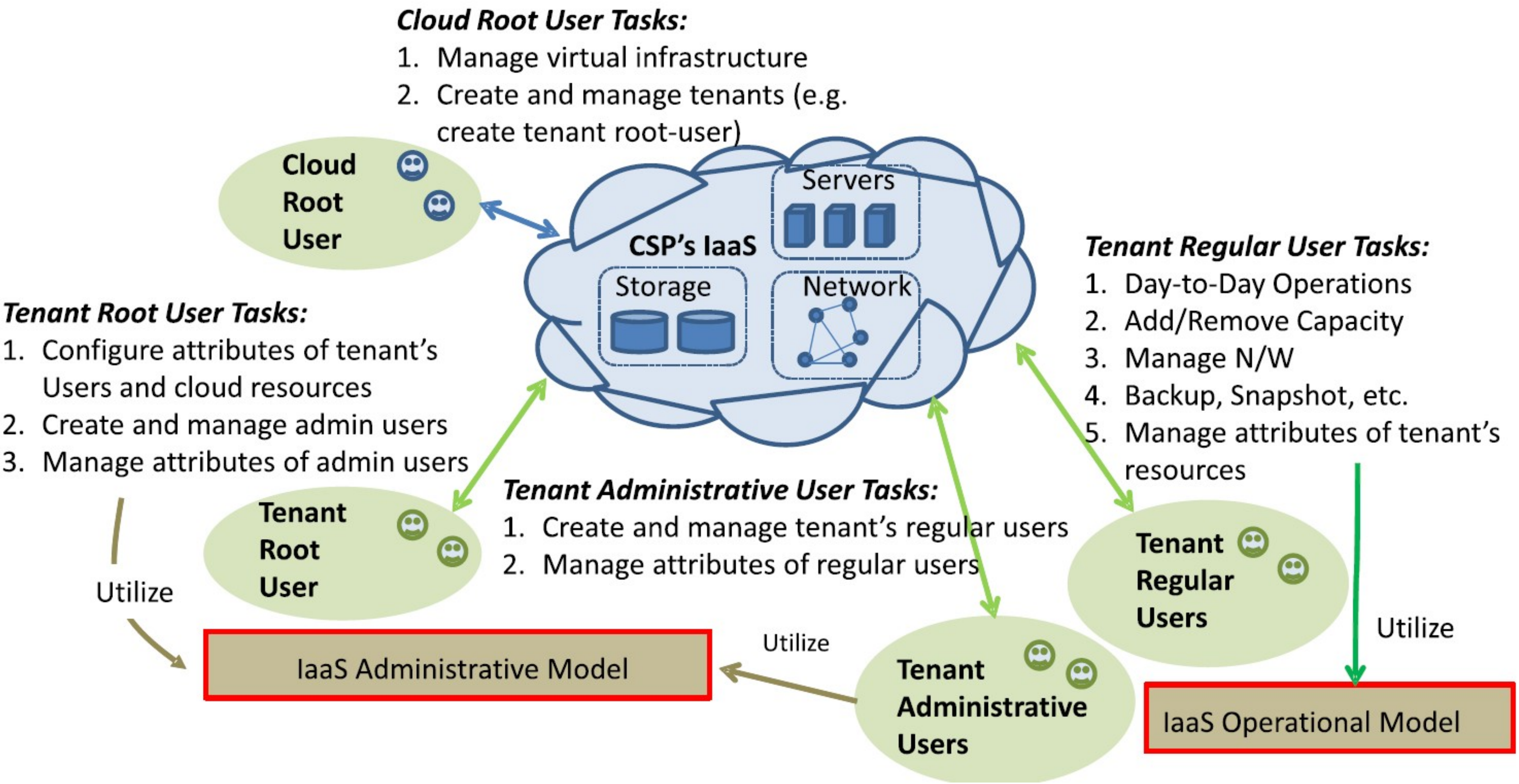
AWS Access Control

- Advantages over OpenStack
  - Tenant has full control over their own policy, by account root user
  - Flexible policy : groups, user id, time, address.
  - Control over resources and operations
- Limitations
  - No automation
  - Restricted set of attributes
  - Not flexible enough, group explosion (e.g., can not configure DAC, cumbersome to configure MAC)
  - No extension available (e.g., can not include customized attributes)
  - No subject and user distinction

ABAC Solving Problems

- Flexibility
  - Covers DAC, MAC and RBAC
  - Covers RBAC extensions
  - Resource-level fine-grained access control
- Automation
  - User attributes inherited by subject and further object, access control automatically added for newly created objects
- Ease in policy specification
  - Attributes defined to reflect semantic meaning and policy specified with certain level of relationship to natural language

Access Control in IaaS



Cloud Root User Tasks:

1. Manage virtual infrastructure
2. Create and manage tenants (e.g. create tenant root-user)

Cloud Root User

CSP's IaaS

Servers

Storage

Network

Tenant Root User Tasks:

1. Configure attributes of tenant's Users and cloud resources
2. Create and manage admin users
3. Manage attributes of admin users

Tenant Regular User Tasks:

1. Day-to-Day Operations
2. Add/Remove Capacity
3. Manage N/W
4. Backup, Snapshot, etc.
5. Manage attributes of tenant's resources

Tenant Administrative User Tasks:

1. Create and manage tenant's regular users
2. Manage attributes of regular users

Tenant Root User

Tenant Regular Users

Tenant Administrative Users

Utilize

IaaS Administrative Model

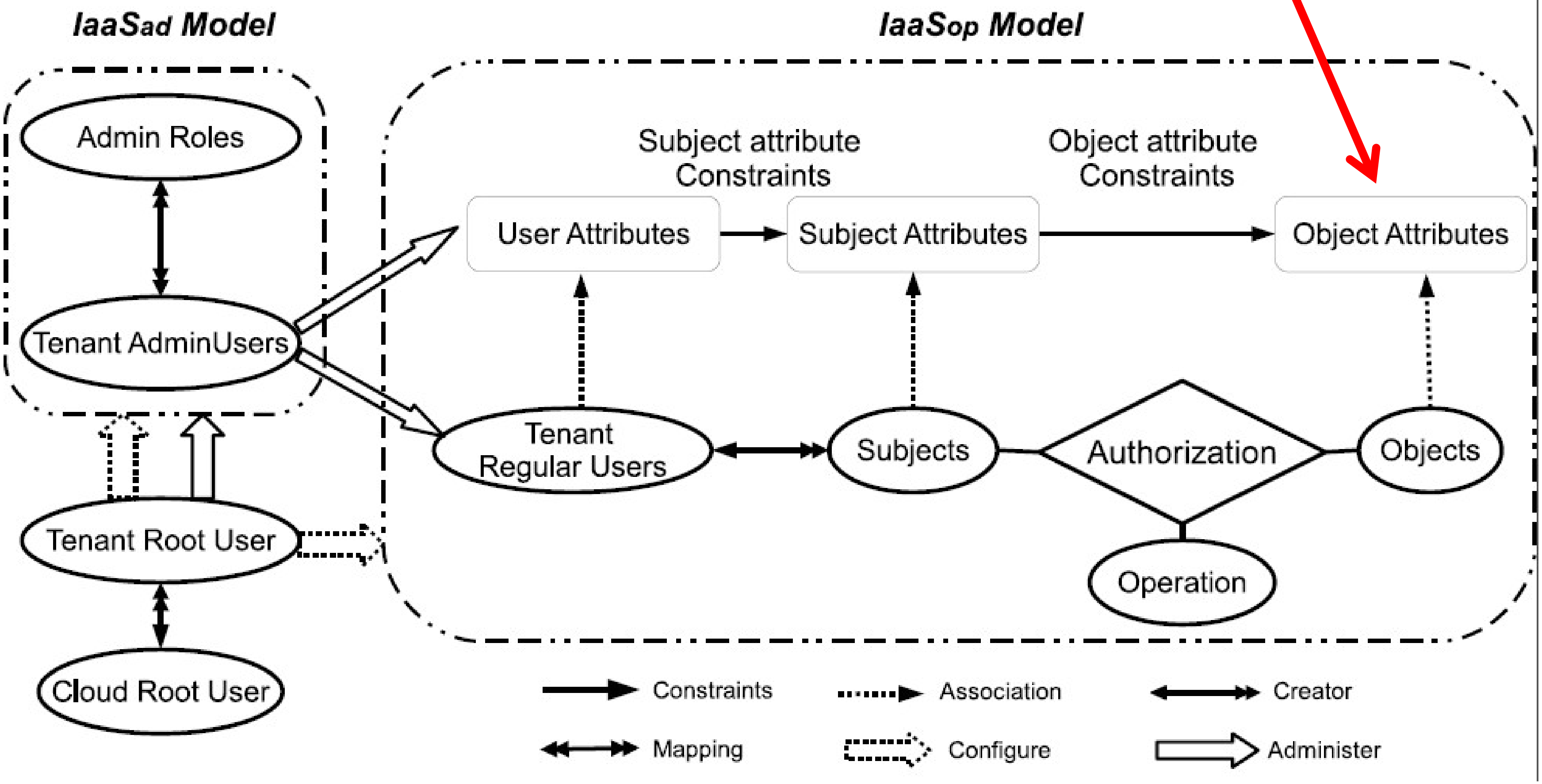
Utilize

Utilize

IaaS Operational Model

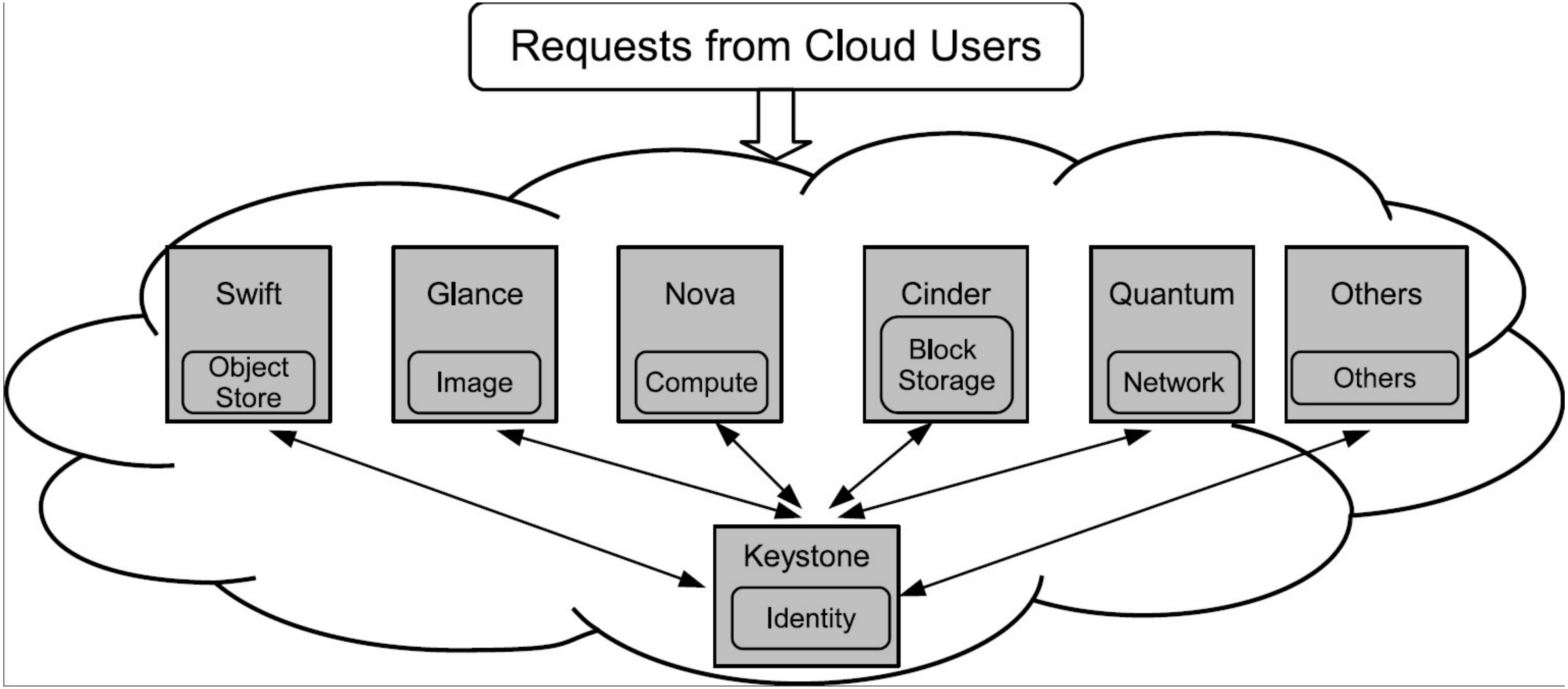
IaaS$_{ad}$ and IaaS$_{op}$ Model

Different types of object may have different sets of attributes.

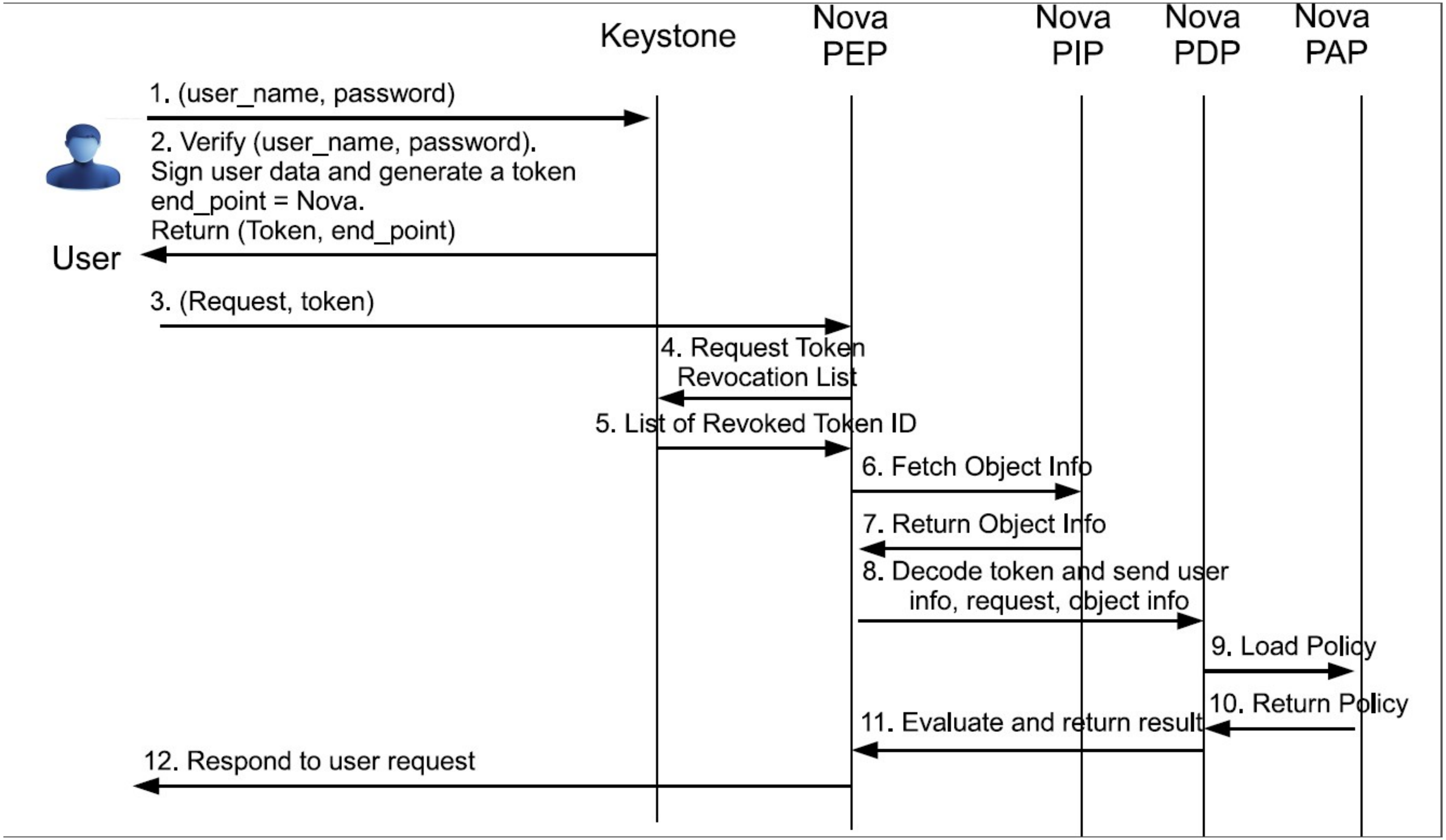


World-Leading Research with Real-World Impact!

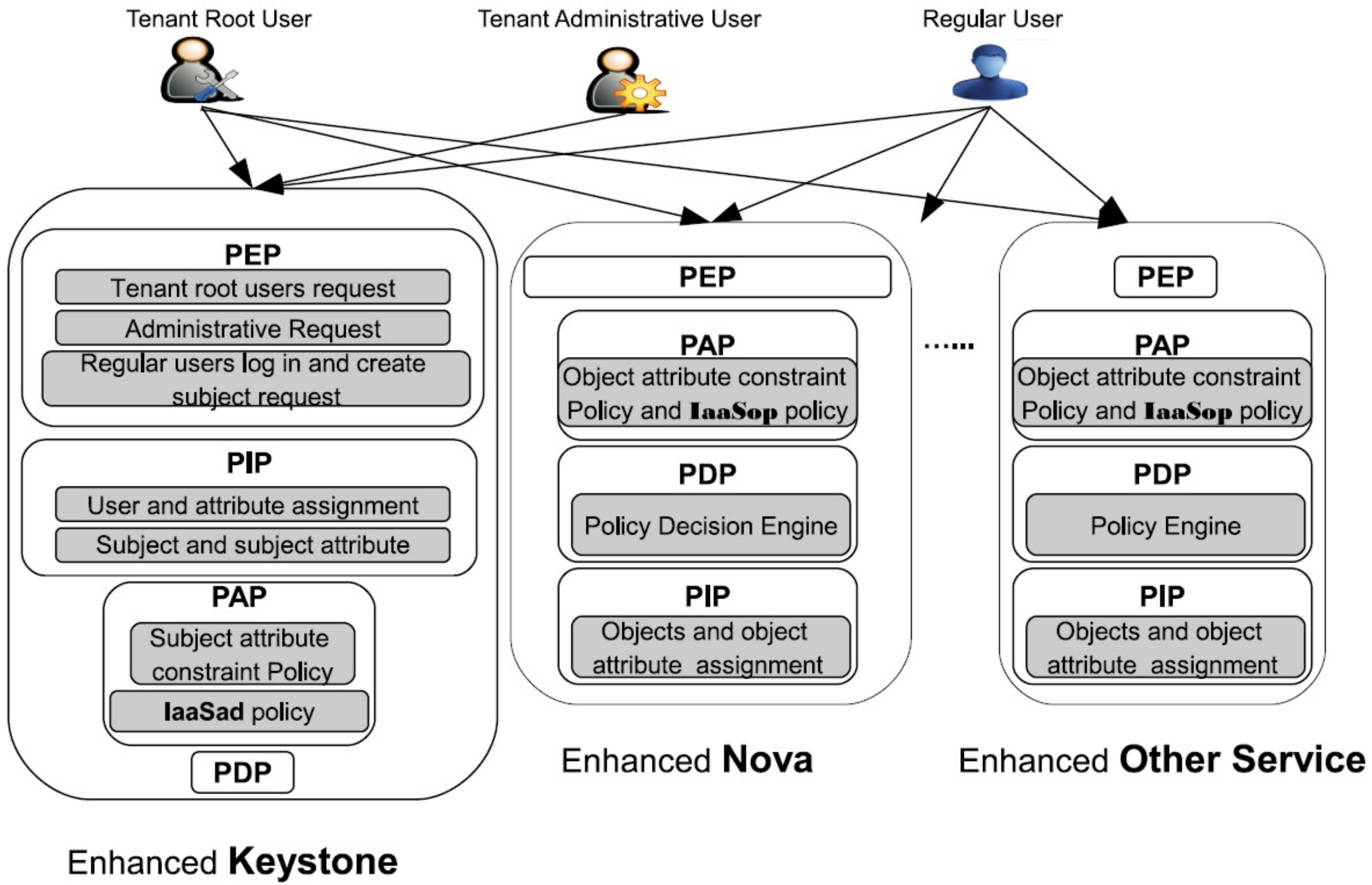
OpenStack


Requests from Cloud Users
Swift
Object Store
Glance
Image
Nova
Compute
Cinder
Block Storage
Quantum
Network
Others
Others
Keystone
Identity

OpenStack Authorization for Nova


Keystone
Nova PEP
Nova PIP
Nova PDP
Nova PAP
User
1. (user_name, password)
2. Verify (user_name, password).
Sign user data and generate a token
end_point = Nova.
Return (Token, end_point)
3. (Request, token)
4. Request Token Revocation List
5. List of Revoked Token ID
6. Fetch Object Info
7. Return Object Info
8. Decode token and send user info, request, object info
9. Load Policy
10. Return Policy
11. Evaluate and return result
12. Respond to user request

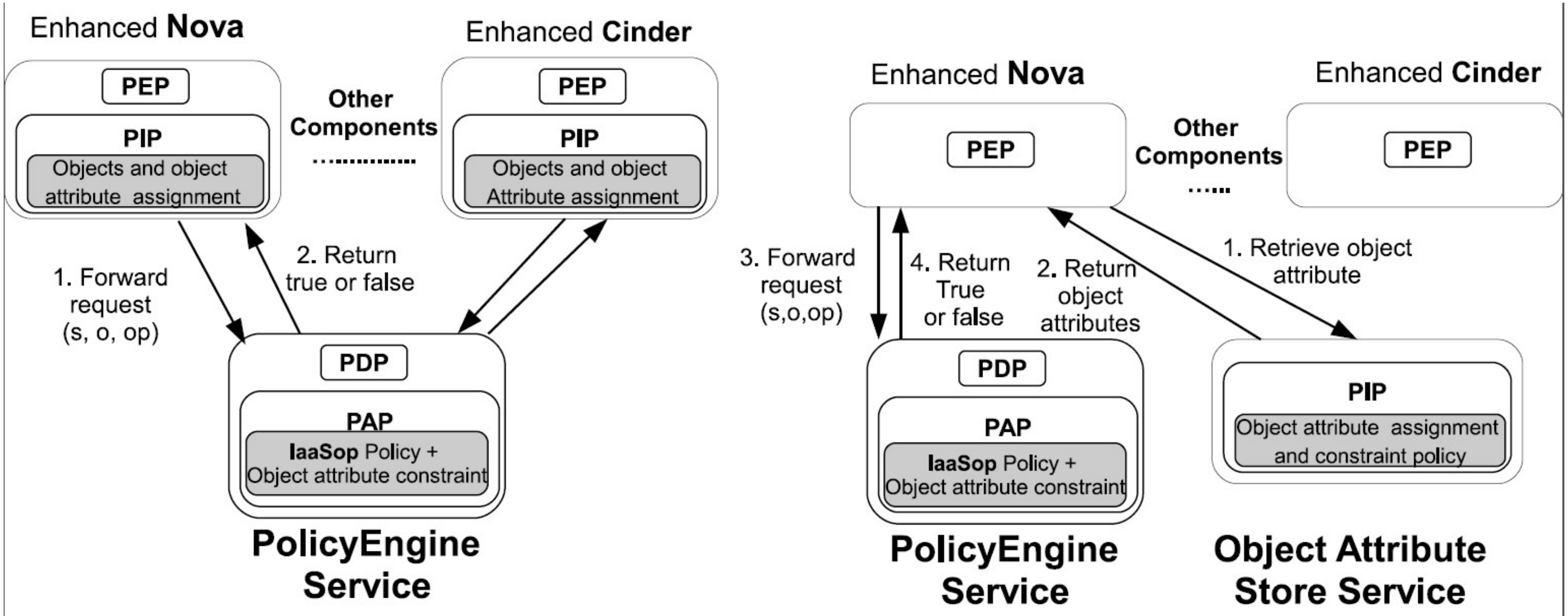
Enforcement Models


Tenant Root User
Tenant Administrative User
Regular User
PEP
Tenant root users request
Administrative Request
Regular users log in and create subject request
PIP
User and attribute assignment
Subject and subject attribute
PAP
Subject attribute constraint Policy
IaaSad policy
PDP
Enhanced Keystone
PEP
PAP
Object attribute constraint Policy and IaaSop policy
PDP
Policy Decision Engine
PIP
Objects and object attribute assignment
Enhanced Nova
……
PEP
PAP
Object attribute constraint Policy and IaaSop policy
PDP
Policy Engine
PIP
Objects and object attribute assignment
Enhanced Other Service

Enforcement Model I

World-Leading Research with Real-World Impact!

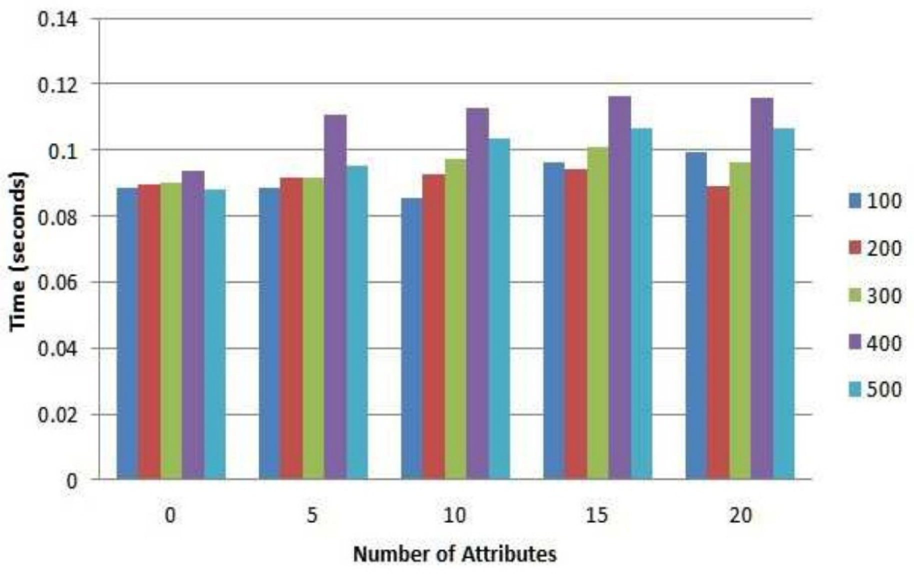
Enforcement Models

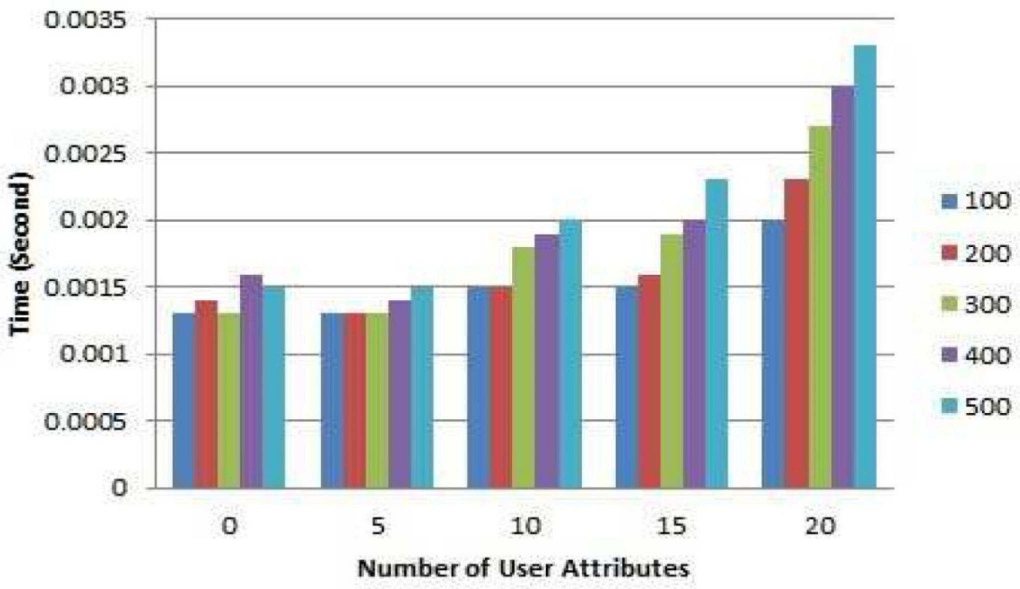

Enhanced Nova
PEP
PIP
Objects and object attribute assignment
Other Components
Enhanced Cinder
PEP
PIP
Objects and object Attribute assignment
1. Forward request (s, o, op)
2. Return true or false
PDP
PAP
IaaSop Policy + Object attribute constraint
PolicyEngine Service
Enhanced Nova
PEP
Other Components
Enhanced Cinder
PEP
3. Forward request (s,o,op)
4. Return True or false
2. Return object attributes
1. Retrieve object attribute
PDP
PAP
IaaSop Policy + Object attribute constraint
PIP
Object attribute assignment and constraint policy
PolicyEngine Service
Object Attribute Store Service

(a) Enforcement Model II

(b) Enforcement Model III

Experiment Result





Time for generating token from Keystone
(Enforcement Model 1)

Time for receiving request from PolicyEngine
(Enforcement Model 2)

Conclusion

- Policy
  - Formal Operational Model. ABAC-alpha to cover classical models DAC, MAC and RBAC; ABAC-beta extends ABAC-alpha to cover extensions to RBAC model which is dominant in recent decades
  - Formal administration Model GURA. Straight forward extension to Administrative RBAC model, easy extension to attribute based model
  - Formal reachability analysis on GURA model, future analysis on extended models subsumes our results
- Enforcement
  - ABAC designed for single tenant access control in IaaS
- Implementation
  - Implement ABAC on selected components in OpenStack and evaluate performance

Publications

[1] Xin Jin, Ram Krishnan, and Ravi Sandhu. A unified attribute-based access control model covering DAC, MAC and RBAC. *Data and Applications Security and Privacy XXVI, pages 41–55, 2012 (cited by 32)*

[2] Xin Jin, Ram Krishnan, and Ravi Sandhu. A role-based administration model for attributes. *In Proceedings of the First International Workshop on Secure and Resilient Architectures and Systems, pages 7–12. ACM, 2012.*

[3] Xin Jin, Ram Krishnan, Ravi Sandhu, Reachability analysis for role-based administration of attributes. *ACM DIM Workshop , held In Conjunction with ACM CCS , 2013.*

[4] Xin Jin, Ram Krishnan, Ravi Sandhu, Unified attribute based access control model covering RBAC and its extensions. *To be submitted to journal.*

[5] Xin Jin, Ram Krishnan, Ravi Sandhu, Attribute-Based Access Control for Cloud Infrastructure as a Service. *To be submitted to conference.*

Others:

[6] Xin Jin, Ravi Sandhu, and Ram Krishnan. RABAC: Role-centric attribute-based access control. *In 6th International Conference, on Mathematical Methods, Models, and Architectures for Computer Network Security, MMM-ACNS 2012.*

[7] Ravi Sandhu, Khalid Zaman Bijon, Xin Jin, and Ram Krishnan. RT-based administrative models for community cyber security information sharing. *In Collaborative Computing: Networking, Applications and Work sharing (CollaborateCom), 2011 7th International Conference on, pages 473–478. IEEE, 2011.*

Thanks
Questions?