Policy Machine

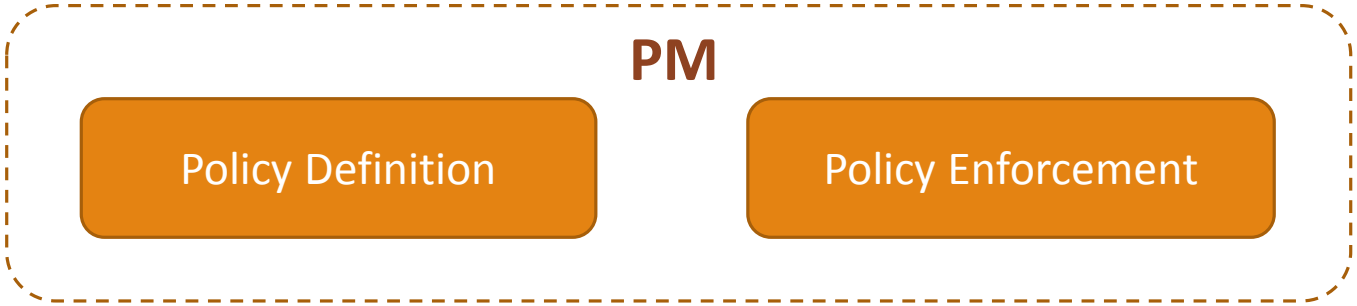
PRESENTED BY:

SMRITI BHATT

Overview

- Many policies and access control models –

  DAC, MAC, RBAC, ABAC, LaBAC, ReBAC, …
- Policy Machine – immense concept and capabilities
- PM vs ABAC
  - Attributes, relationships, etc.



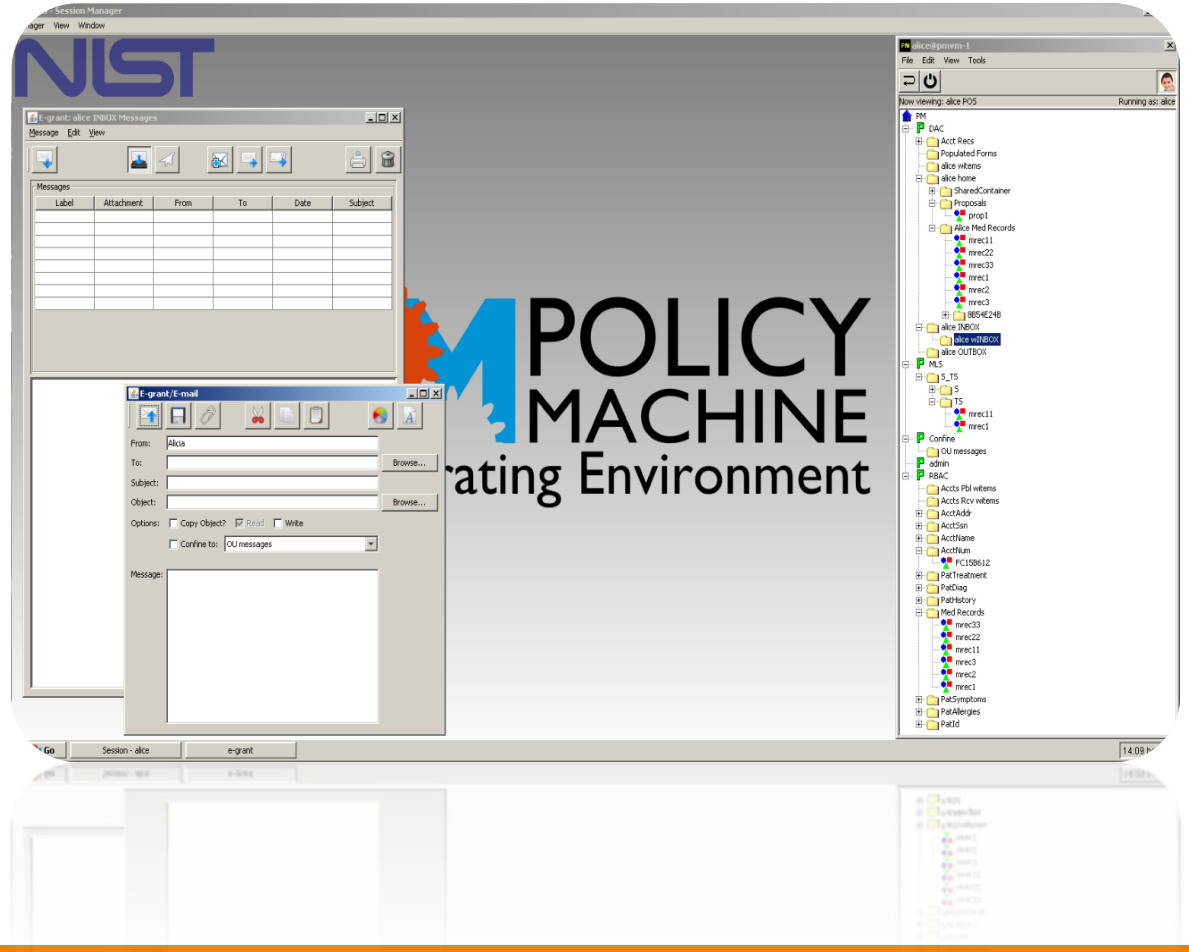
Introduction

Policy Machine (PM):

- Unified access control framework
- Express and enforce arbitrary access control policies
- Attribute-based access control policies

Goal & Objective:

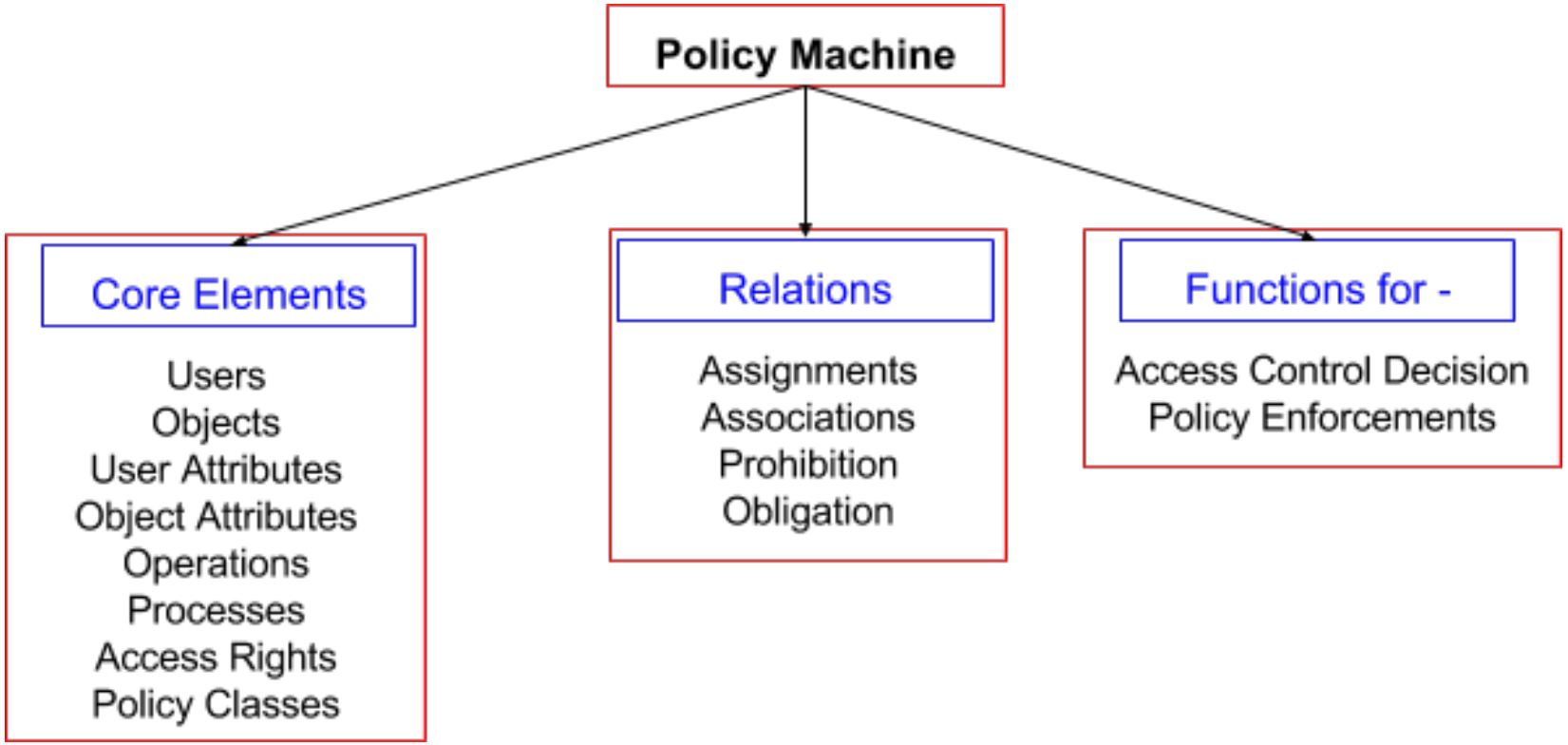
- To provide a generic platform that supports
  - Commonly known and implemented access control policies
  - Combinations of policies
  - Policies for which no access control mechanism presently exists



Characteristics of PM

- Comprehensive enforcement of many policies across both centralized and distributed systems
- Single administrative domain
- Comprises of a family of standards that recognizes
  - Policy Administration Point (PAP),
  - Policy enforcement points (PEP),
  - Policy Decision Point(s) (PDP),
  - A policy database

PM Framework- a logical ‘machine’



PM Core Elements

User Attributes (UA) and Object Attributes (OA):containers for users and objects respectively

Policy Classes (PC): containers for each access control policies

Processes (p): similar to a subject, issues access request

PM vs ABAC Attributes

ABAC attributes

- Characteristics/properties of users and objects
- Finite domain
- Set or atomic valued
- Example:
- User attribute –
  - Title = { manager, employee}
  - Age = {18, 19}
- Object attribute –
  - Owner = Alice
  - Label = {private, public}

PM attributes

- Containers for users and objects
- No defined domain
- Set of members in the container
- Example:
- User attribute –
  - Manager = manager_1, manager 2
  - Employee = employee_1, employee 2
- Object attribute –
  - Medical records = mrec1, mrec2

Attributes (User/Object)

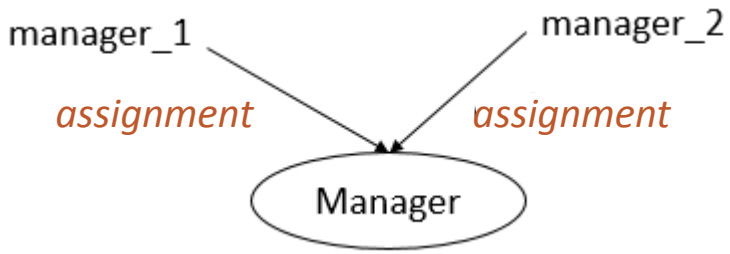


Fig1: PM user attribute and assigned users

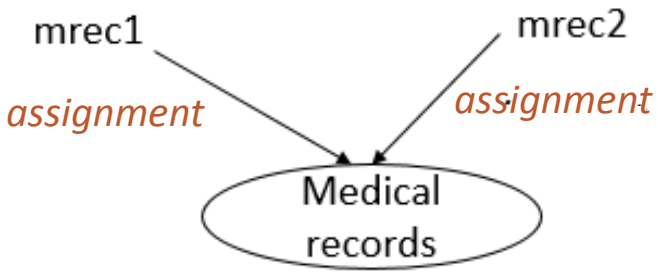


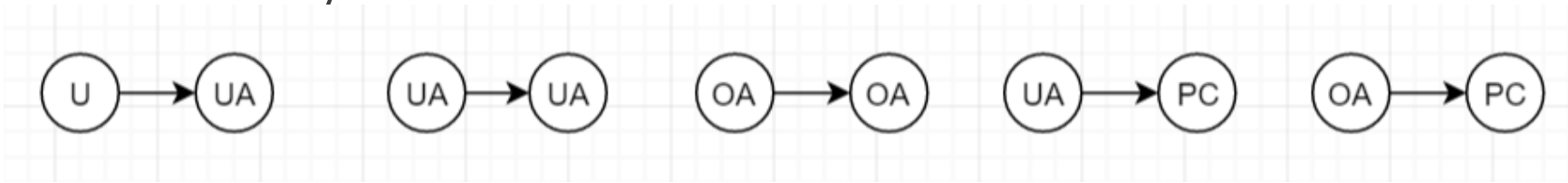
Fig2: PM object attribute and assigned objects

Fig3: ABAC user and attribute-value pair

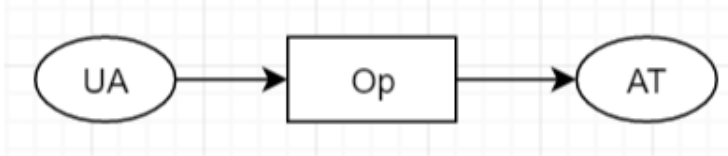
Fig4: ABAC object and object-value pair

PM Relations

Assignment: x ASSIGN y



Association: relationship between user attributes (UA) and user attributes and objects attributes (AT = UA U OA) via operations/actions, policies



Prohibition: constraints and restrictions on access rights on users, processes or user attributes, SSD

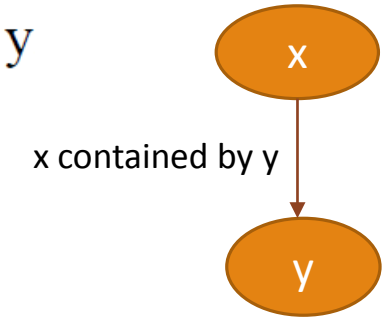
Obligations: dynamically change the policy state in response to user/process event, DSD

When *pattern* ***do*** *response*

PM Assignments

- Containment Property:

$x \ \text{ASSIGN}^{+} \ y$



- Hierarchical relationships
- Inheritance shown downwards

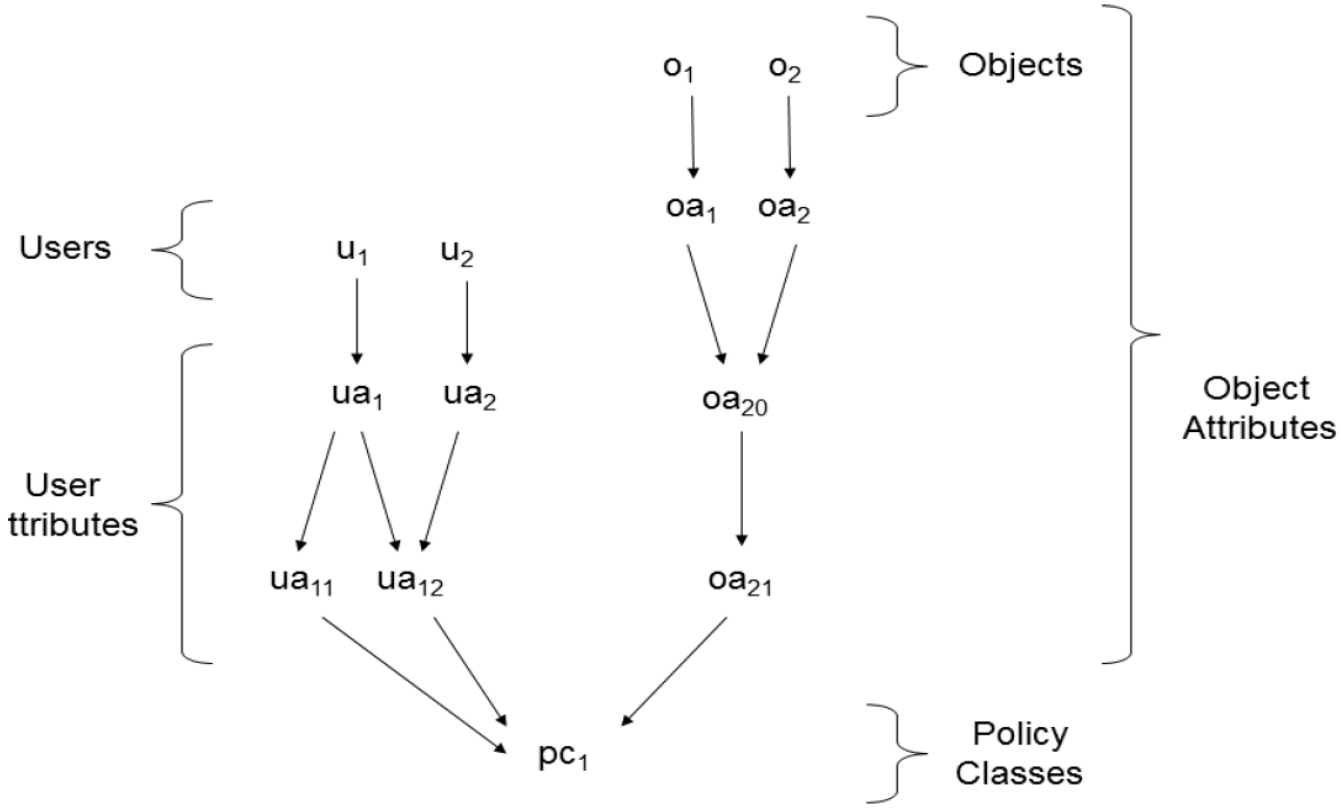
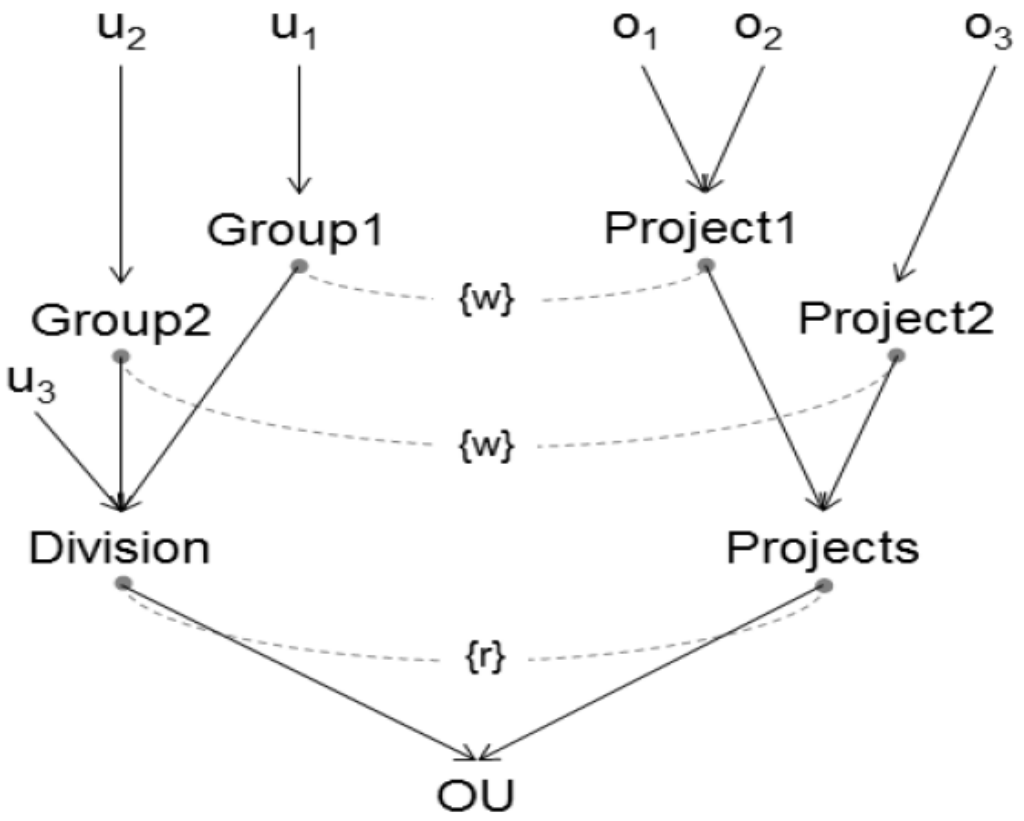


Figure 2: Simplified Policy Element Diagram

PM Authorization Graph



Simple Authorization Graph

Assignments

Associations

{r}, {w} - Operations

Derived Privileges - Enumerations

Enumerations all explicit and implicit privileges/accesses

Association →(Group1, {w}, Project1)

- Users(Group1)={u_1}
- Access right={w}
- Elements(Project1)={Project1, o1, o2}

Derived privileges for user u_1:

- (u_1, w, Project1), (u_1, w, o1), and (u_1, w, o2)

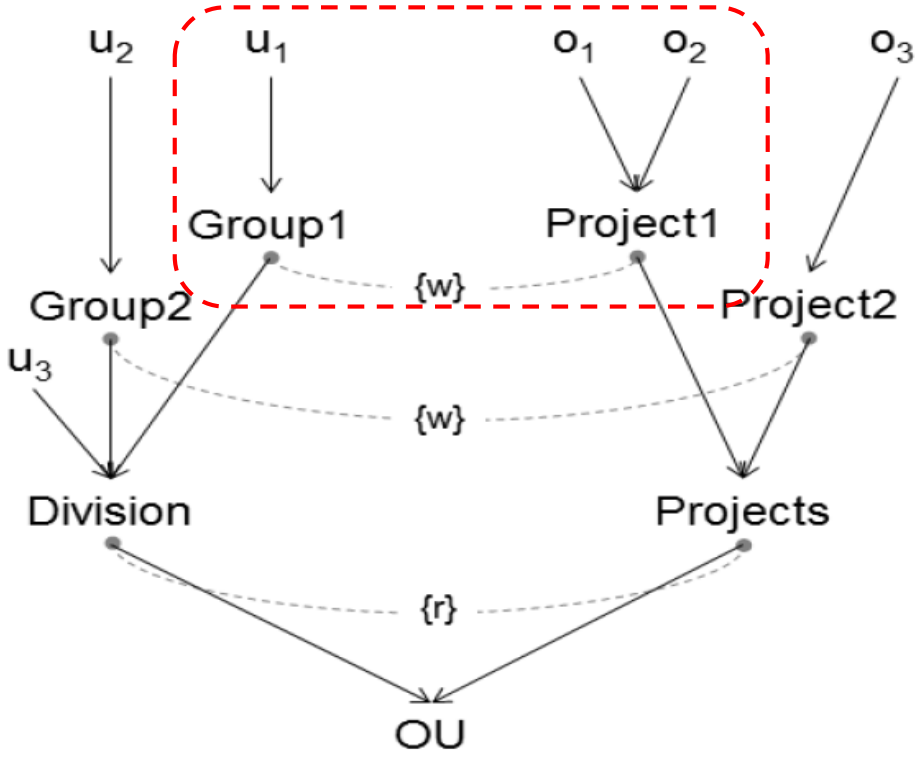
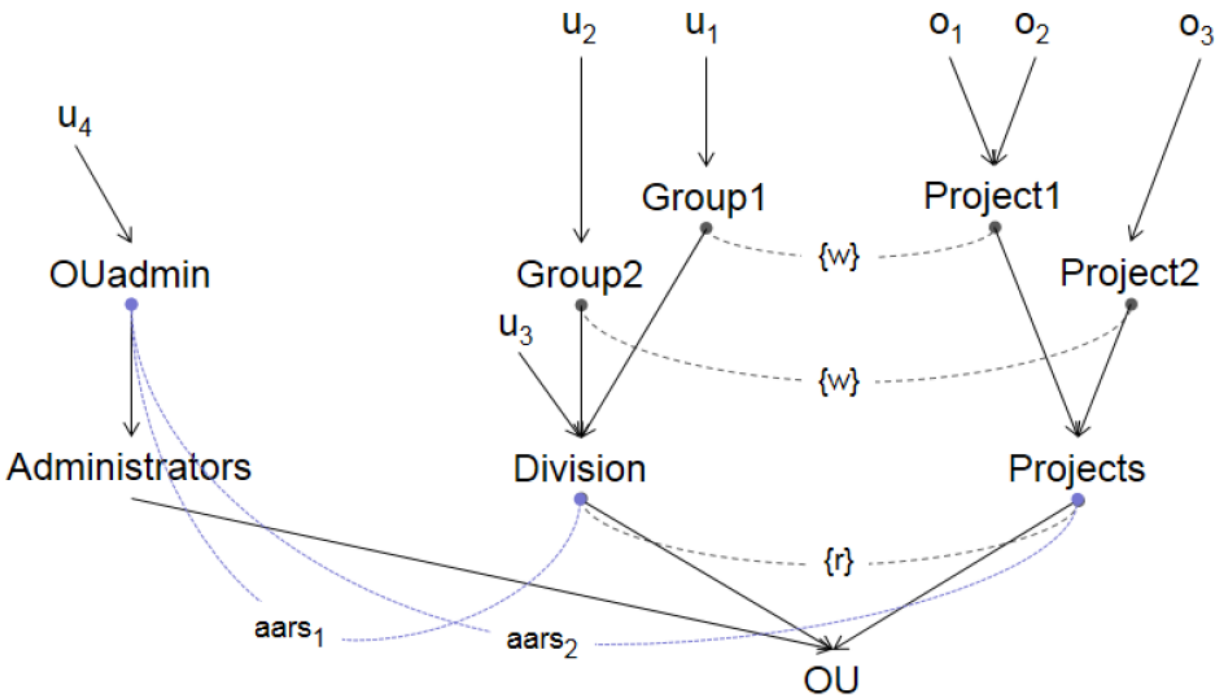


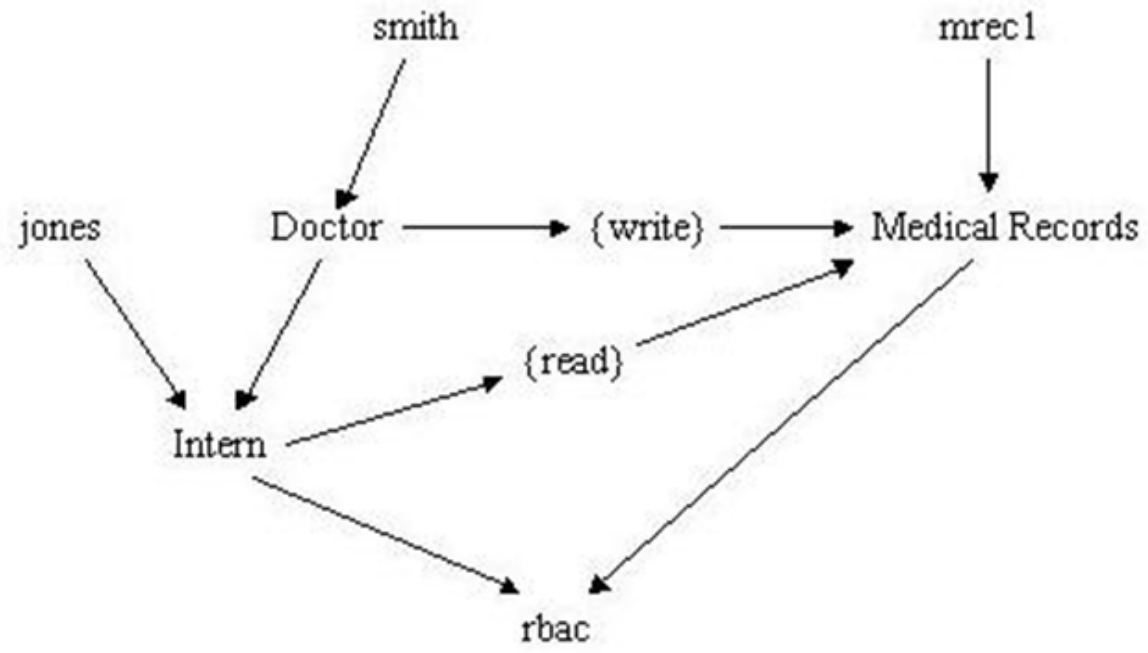
Figure 3: Simple Authorization Graph

PM Administrative Authorization Graph



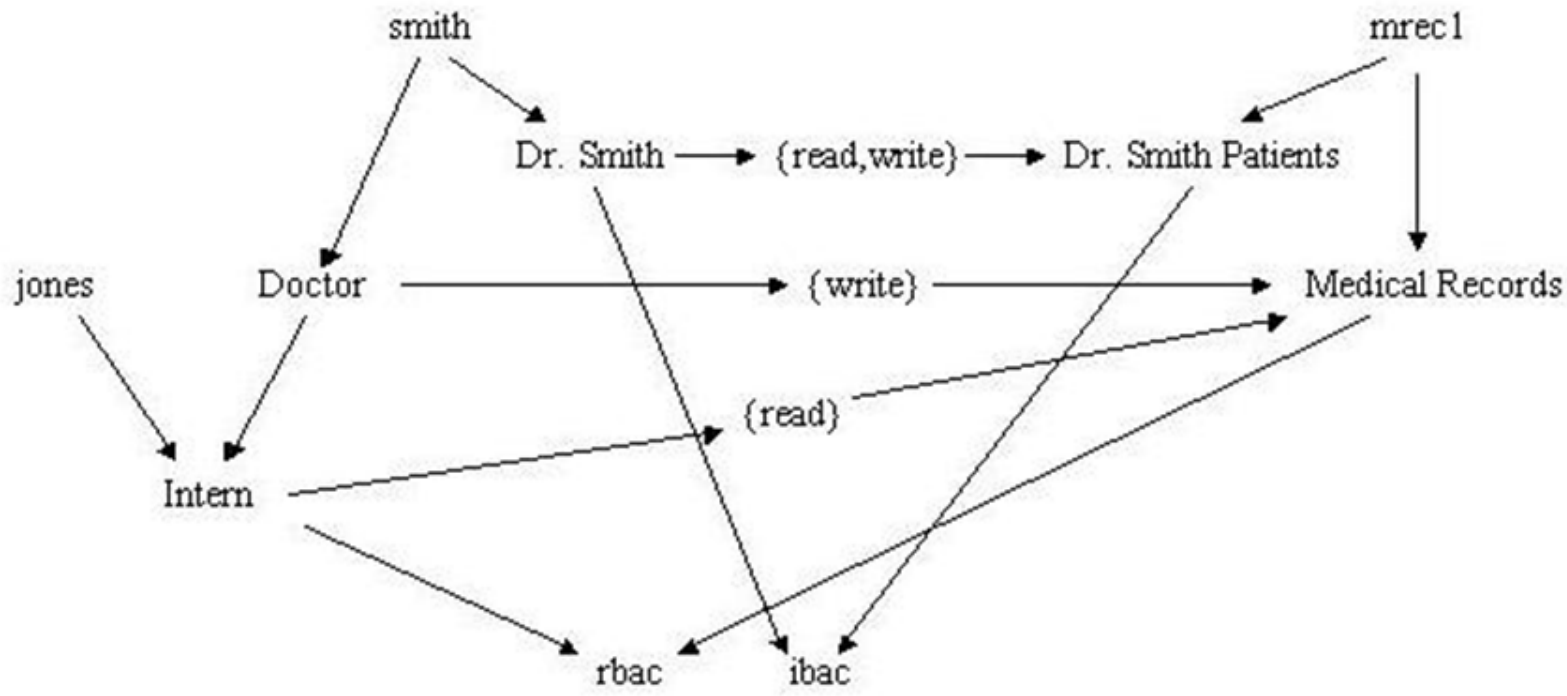
Simple Example Involving Administrative Associations

Example – RBAC Policy



Example of assignments in the Policy Machine

Combination of Policies



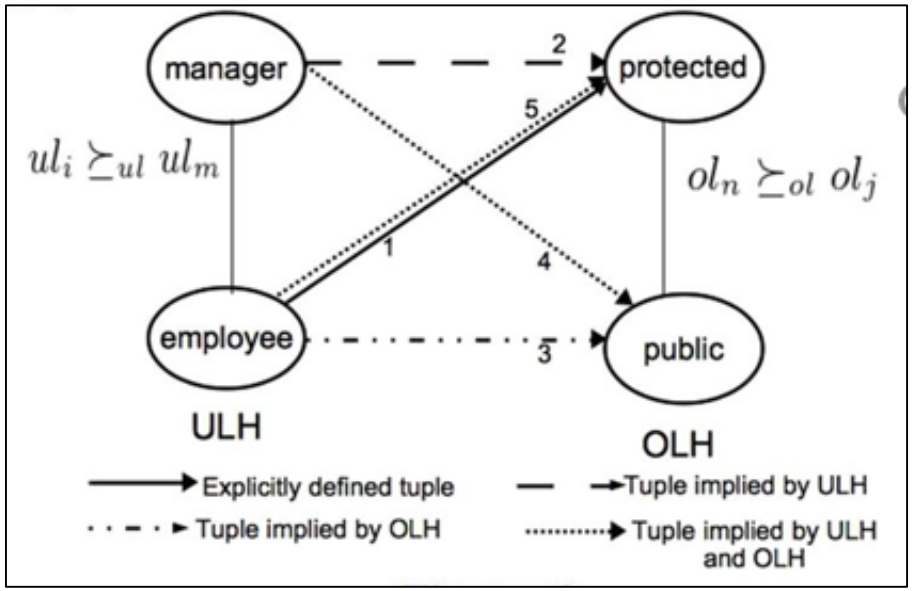
Multiple policies example

LaBAC$_H$

LaBAC$_H$ Characteristics:

- Assignments and Associations relations
- User/object label hierarchy
- Only operations on objects/resources

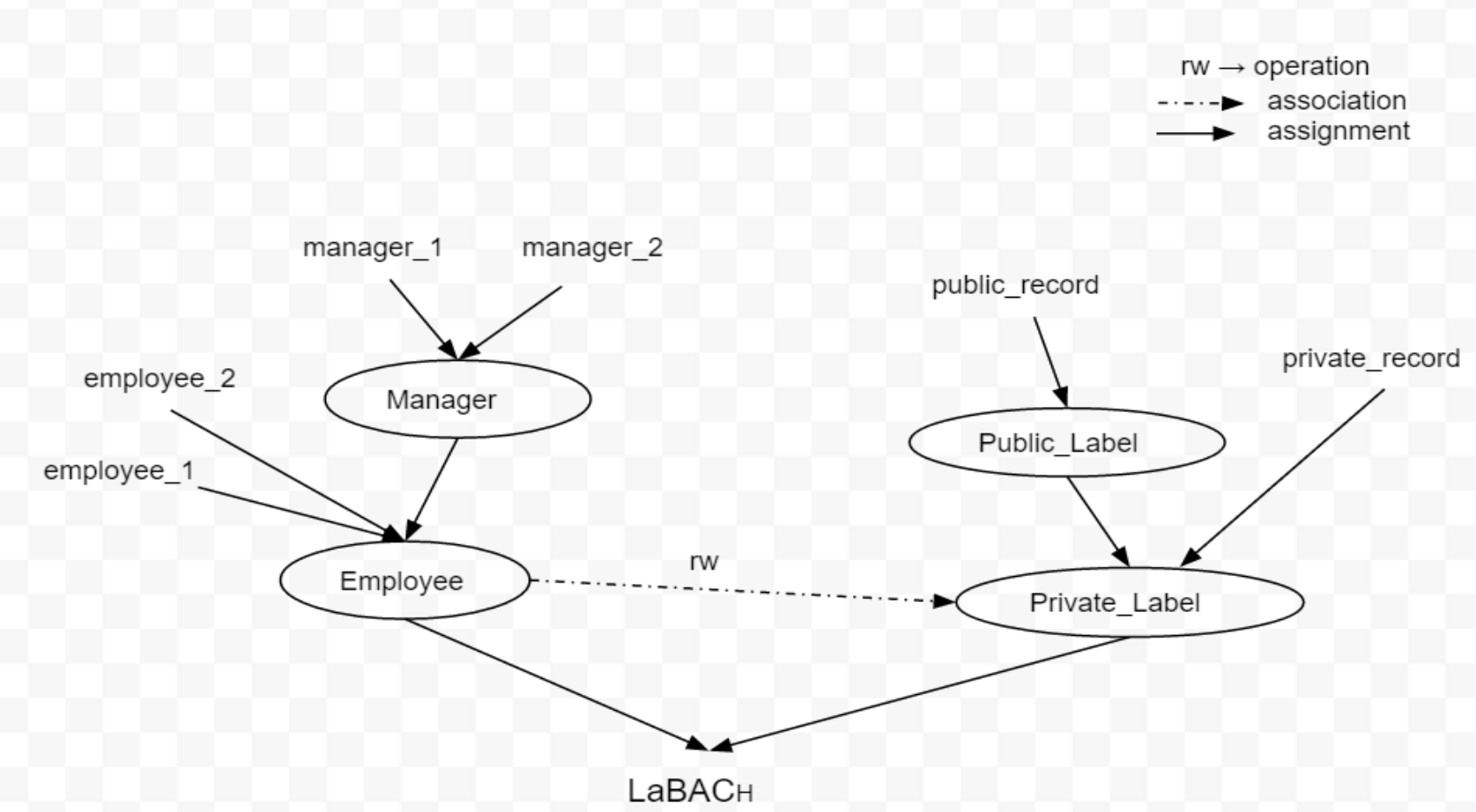
  (no admin operations)
- Enumerated policies



$Policy_a$ = {(employee,protected)}

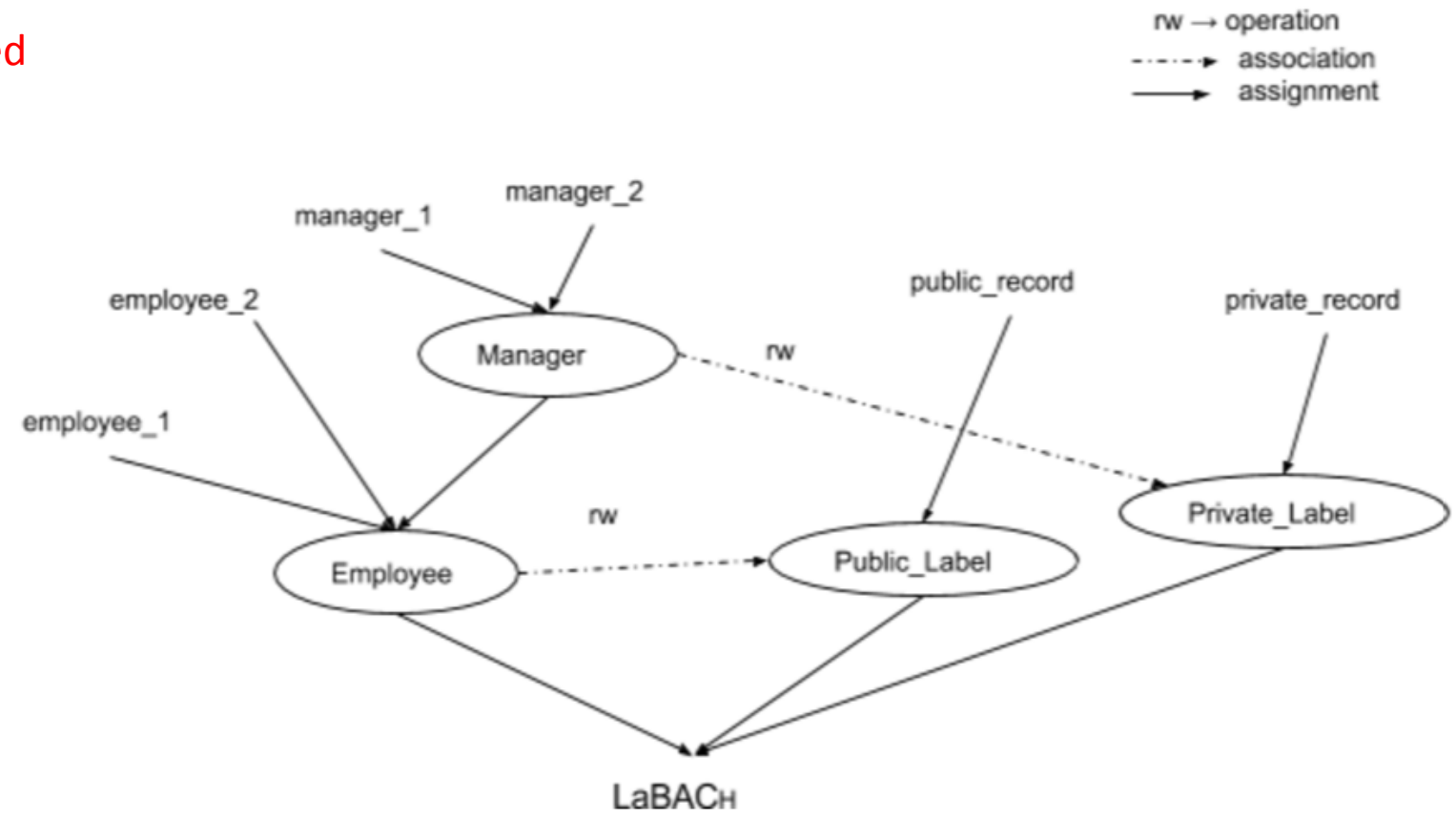
$ImpliedPolicy_a$ = { (employee, protected), (manager, proteced), (employee,public), (manager, public}

LaBAC$_H$ in PM


rw → operation
association
assignment
manager_1
manager_2
employee_2
employee_1
Manager
Employee
public_record
private_record
Public_Label
Private_Label
rw
LaBACH

LaBAC$_H$ in PM

Tweaked



PM Tool

PM Server: Policy Administration Point (PAP), Policy Decision Point (PDP), includes a PM Database, an event processing module

Administrative Client: Admin Tool (PAP)

Database: Active Directory

PM client: PEP, an application programming interfaces (API), and PM-aware applications

User Environment Simulator

- Kernel Simulator: acts as PEP
- Session Manager

PM System Architecture

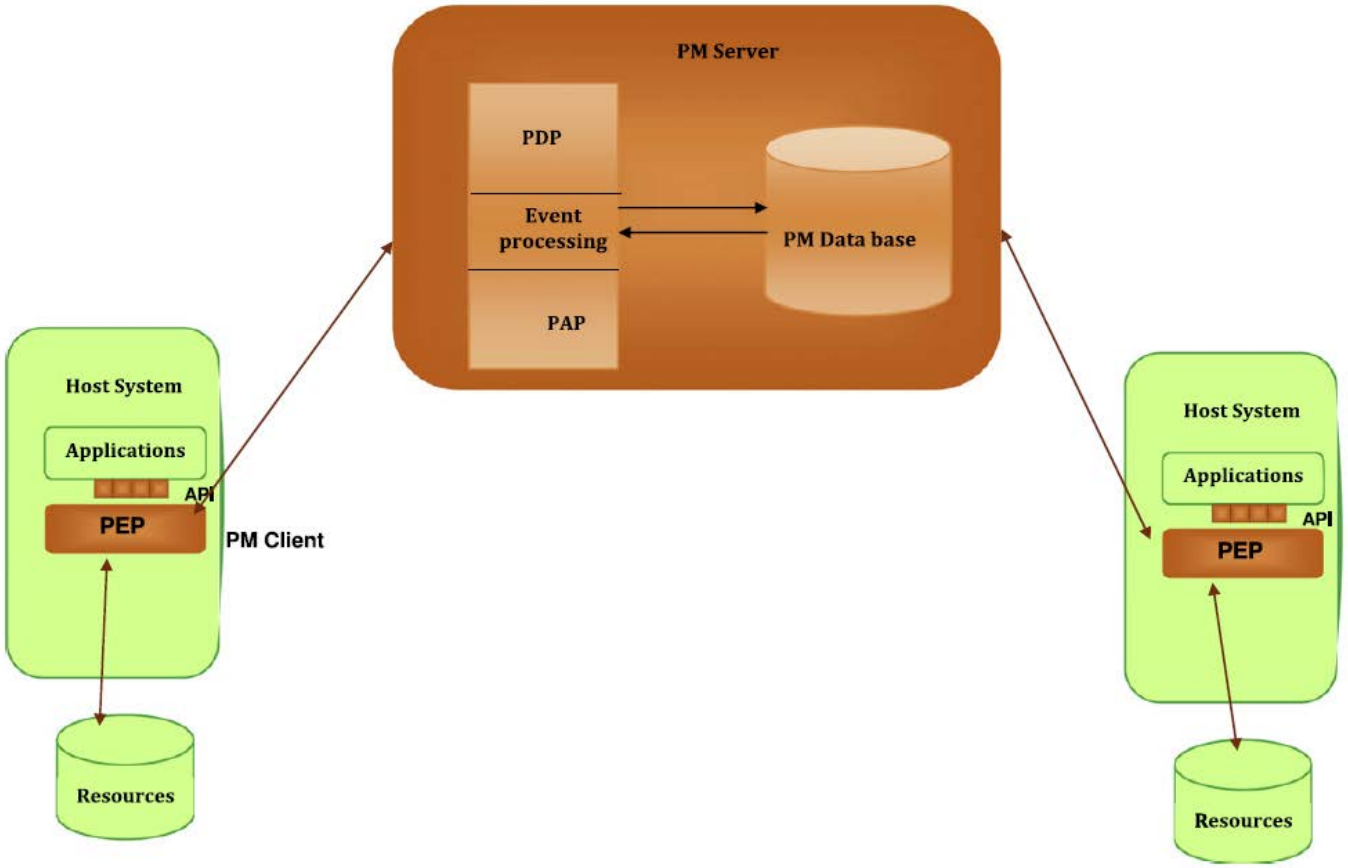


Fig. 1. The system architecture.

LaBAC$_H$ Configuration in PM

Look into PM Tool at the end!!!

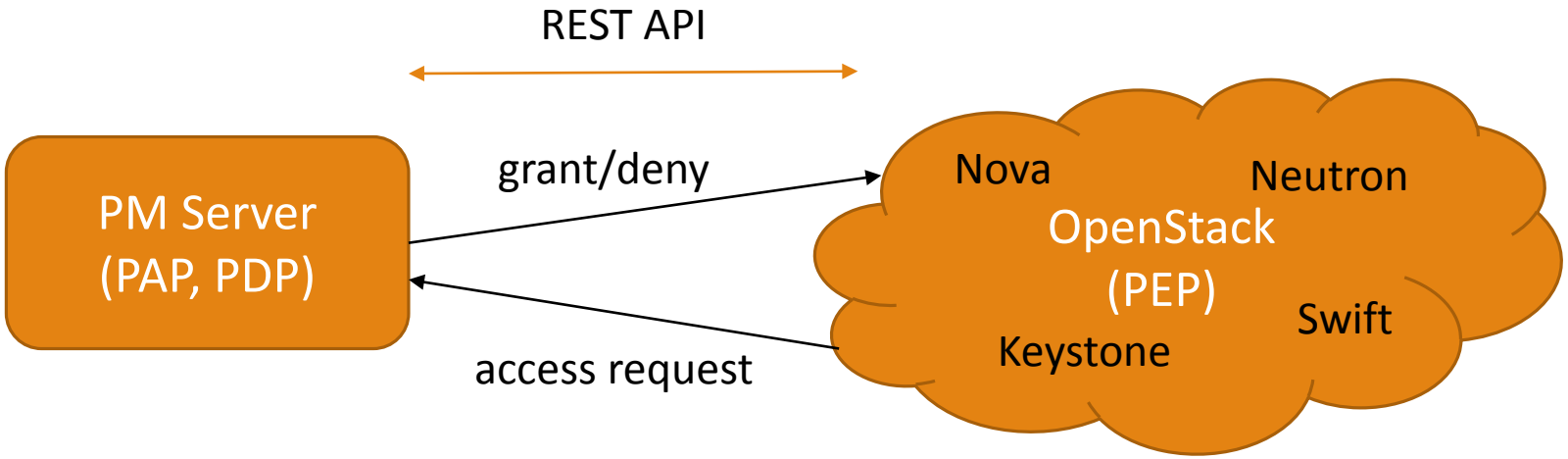
PM and ABAC Analysis

Defining policies in ABAC models:

- Logical formula: ABACα, XACML
  - Pros: simple, easy, and flexible
  - Cons: reviewing, updating policies become NP-complete
- Enumerated policies: LaBAC, 2-sorted RBAC, PM
  - Pros: good for updating and reviewing policies
  - Cons: size might becomes exponential

PM in Cloud

PM & OpenStack



PM & OpenStack

- ✓ Better understand PM and OpenStack
- ✓ Proof of Concept PM-ABAC in cloud
- x Performance overhead
- x PM specific issues

Conclusion & Limitations

- Platform for testing and enforcing policies
- ABAC attributes vs PM attributes
- Configuration overhead
- Many details and complexities
  - Active Directory
  - Certificates and keys, …


Thank You!