Object-Tagged RBAC Model for the Hadoop Ecosystem

Maanak Gupta, Farhan Patwa, and Ravi Sandhu

Institute for Cyber Security and Department of Computer Science
University of Texas at San Antonio

31st Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy (DBSec 2017), Philadelphia, Pennsylvania, July 19-21, 2017

Outline

- Introduction and Motivation
- Multi-layer Access Control
- Authorization Architecture
- Hadoop Ecosystem Access Control Model
- OT-RBAC Model
- Proposed Implementation
- Attribute Based Extensions to OT-RBAC
- Conclusion

Big Data and Big Challenges

- IDC 2025 :
  - global “datasphere” – 163 zettabytes
  - 10x than 2016
- Opportunities: 21st century gold for data miners
- Big Data require “Big Systems”

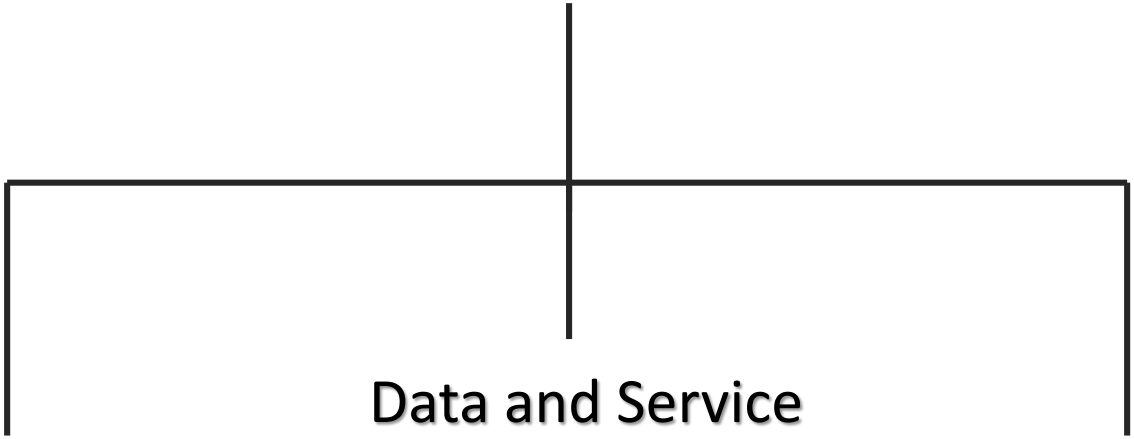
Security:

- Secure Storage
- Privacy Concerns (eg: HIPPA)
- Fine granular access requirements

Hadoop Ecosystem

- Hadoop: resilient, cost efficient distributed storage (HDFS) and processing framework (MapReduce) and YARN
- Ecosystem = Hadoop core + Open-Source Projects
- Hadoop Data Lake
- Security Concerns

Multi-Layer Access Control



Services

HDFS NameNode,
YARN ResourceManager
Apache Hive

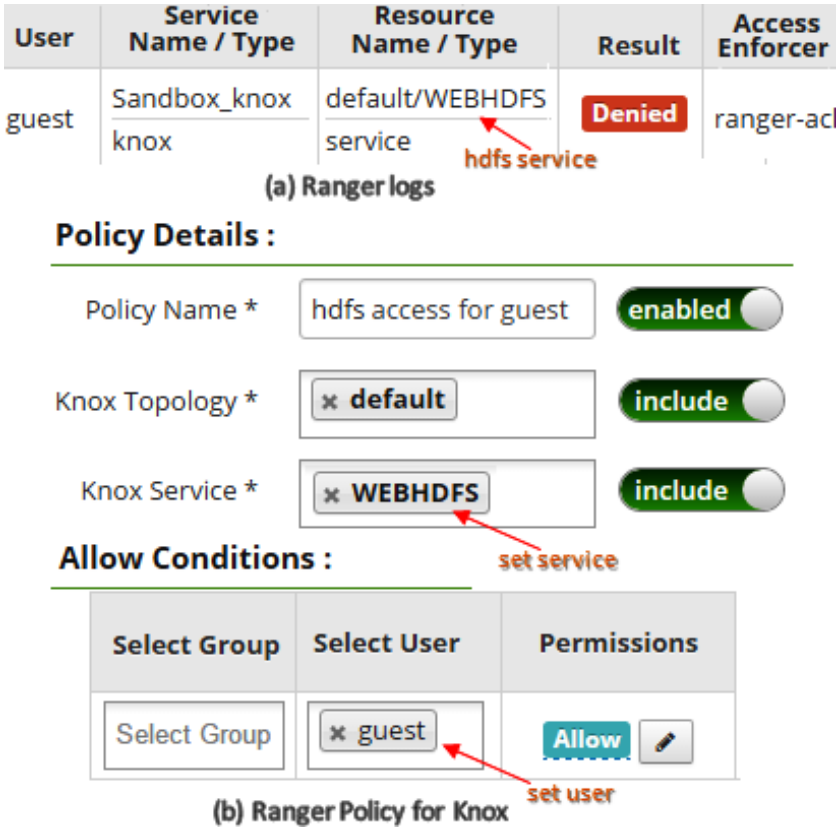
Data and Service Objects

HDFS Files,
Hive Tables
Kafka Topics

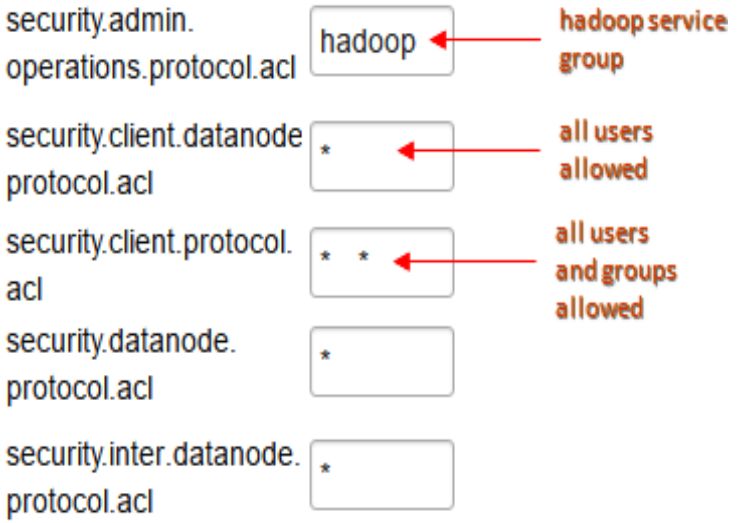
Cluster Resources and Applications

YARN Queues,
Cluster Nodes

Hadoop and Data Services Access

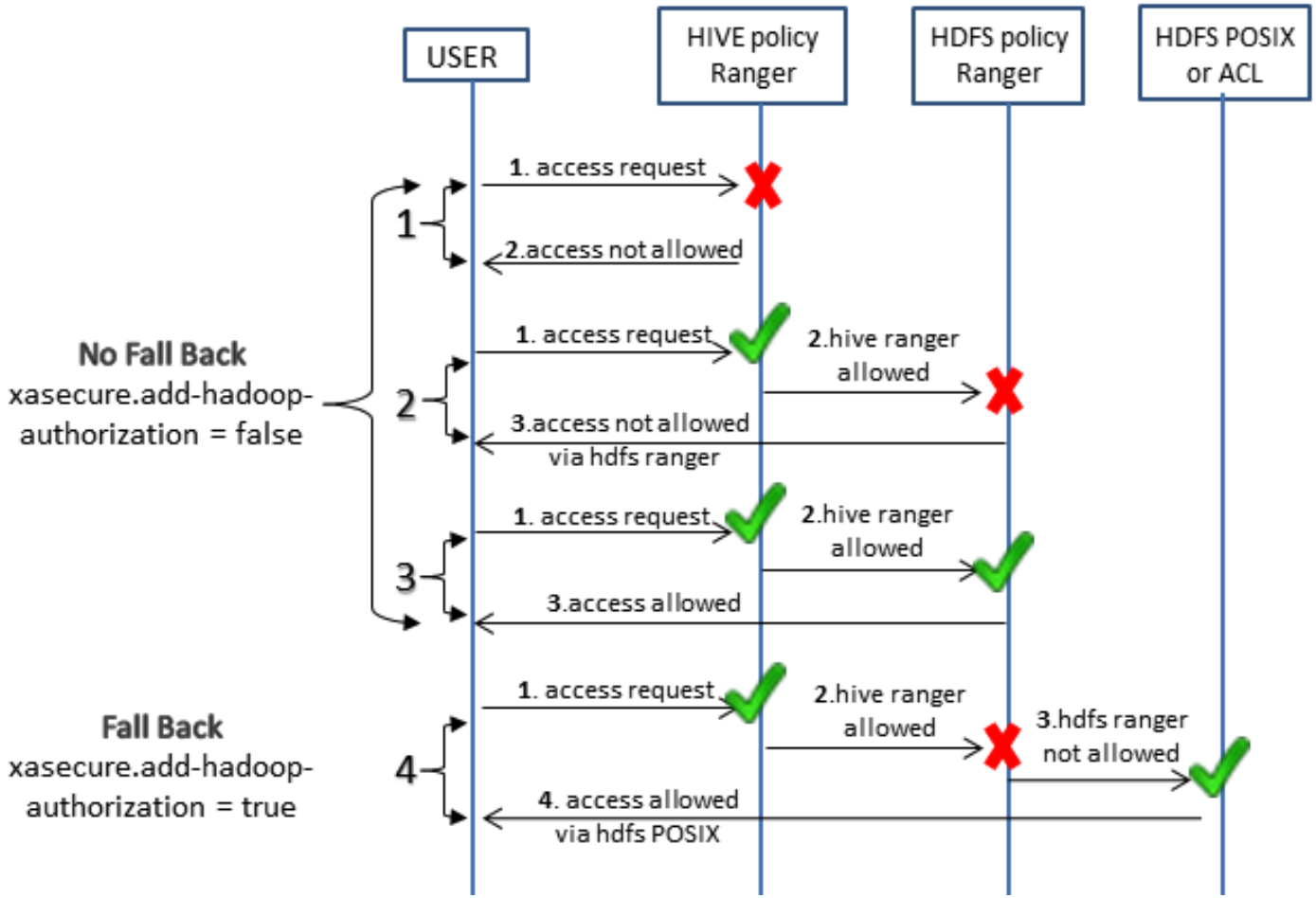


WebHDFS Access via Apache Knox



Hadoop Daemons Access Configuration

Data Objects Access

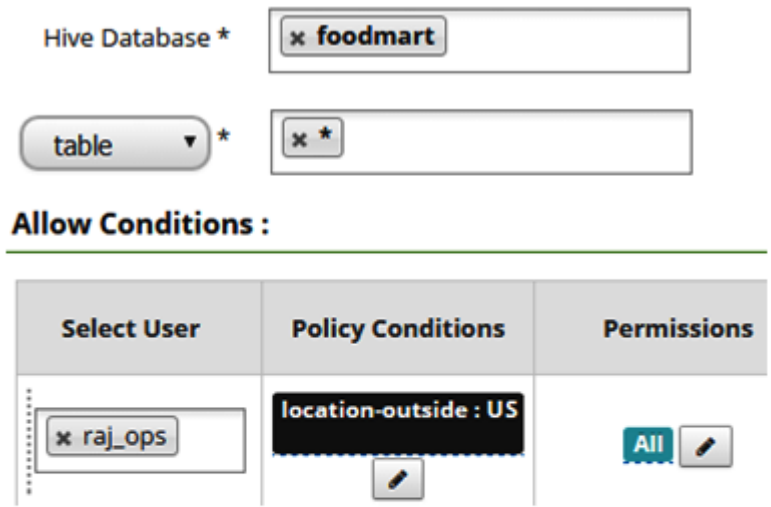


Hive and HDFS Access Configurations

Tag Based Objects Access

Tag Based Policy Configuration

Context Enricher and Policy Conditions



(a) Ranger Policy

```
IP_ADDRESS_FROM, IP_ADDRESS_TO,  COUNTRY_CODE
10.245.121.X,    10.245.124.X,       US
19.145.123.X,    19.145.124.X,       CN
21.245.25.X,     21.245.25.X,        IN
```

(b) Text File

```
1  "id": 25,
2  "service": "Hive",
3  "resources": {
4    "database": {
5      "values": [
6        "foodmart"
7      ],
8    },
9    "table": {
10     "values": [
11       "*"
12     ],
13   }
14 "policyItems": [
15   {
16     "accesses": [
17       {
18         "type": "all",
19         "isAllowed": true
20       }
21     ],
22     "users": [
23       "raj_ops"
24     ],
25     "groups": [],
26     "conditions": [
27       {
28         "type": "location-outside"
29         "values": [
30           "US"
31         ]
32       }
33     ],
```

condition

(c) JSON Policy

Geo Location Based Policies

Cluster Resource and Application Access

yarn.acl.enable 🔒 true yarn.admin.acl root

```
yarn.scheduler.capacity.root.queues=default,newQueue   ← child queues
yarn.scheduler.capacity.root.acl_administer_queue= root
yarn.scheduler.capacity.root.acl_submit_applications=   ← no user
yarn.scheduler.capacity.root.default.acl_submit_applications= raj_ops
yarn.scheduler.capacity.root.newQueue.acl_administer_queue= maria_dev
yarn.scheduler.capacity.root.newQueue.acl_submit_applications= maria_dev
yarn.scheduler.capacity.queue-mappings=u:maria_dev:newQueue
yarn.scheduler.capacity.queue-mappings-override.enable=false
```

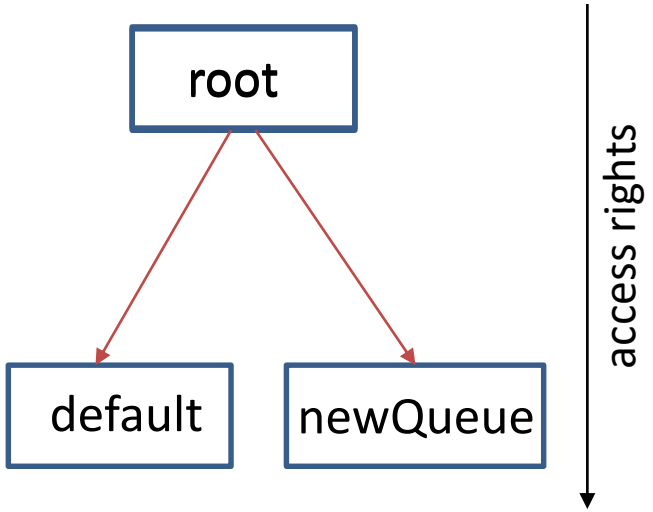
(a) Capacity Scheduler configuration (YARN)

```
Queue acls for user : raj_ops
Queue  Operations
=====================
root
default  SUBMIT_APPLICATIONS
newQueue
```

```
Queue acls for user : root
Queue  Operations
=====================
root   ADMINISTER_QUEUE
default   ADMINISTER_QUEUE
newQueue   ADMINISTER_QUEUE
```

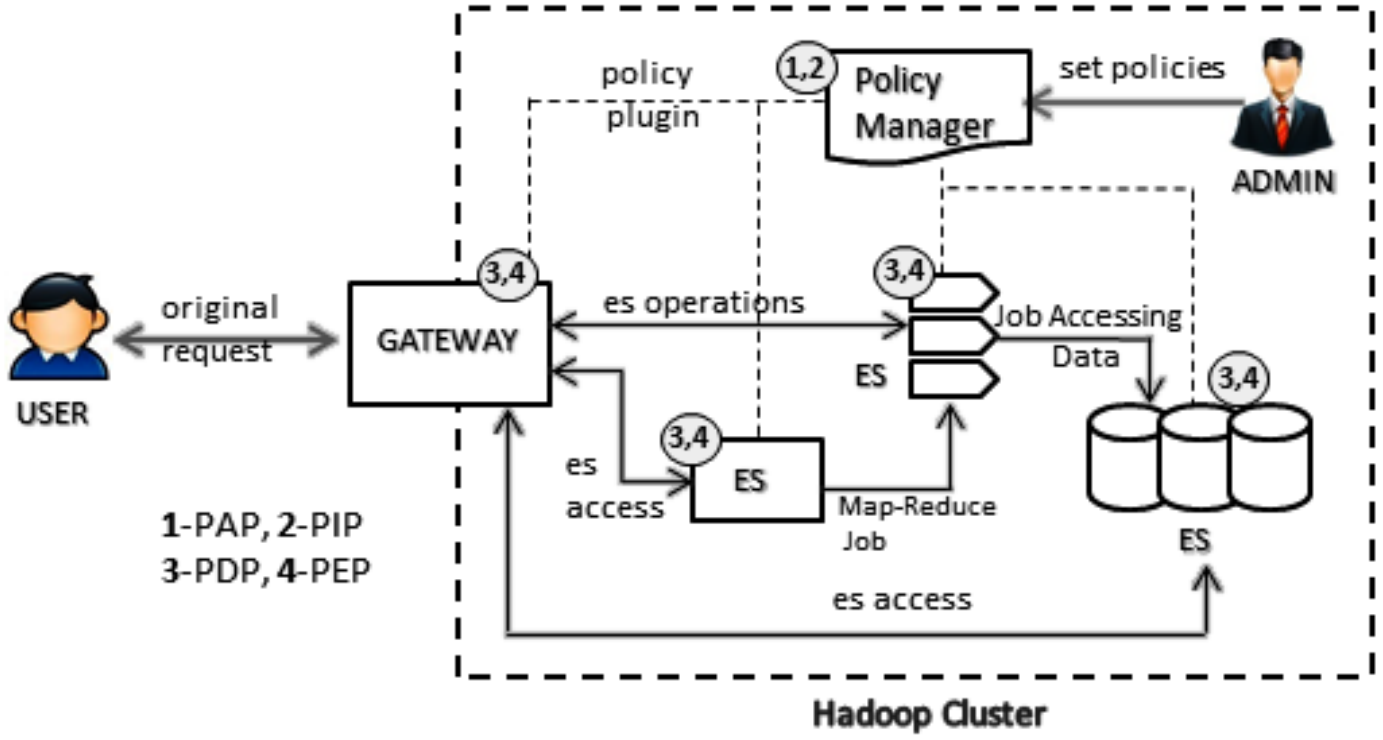
```
Queue acls for user :  maria_dev
Queue  Operations
=====================
root
default
newQueue  ADMINISTER_QUEUE , SUBMIT_APPLICATIONS
```

(b) ACL s for different users



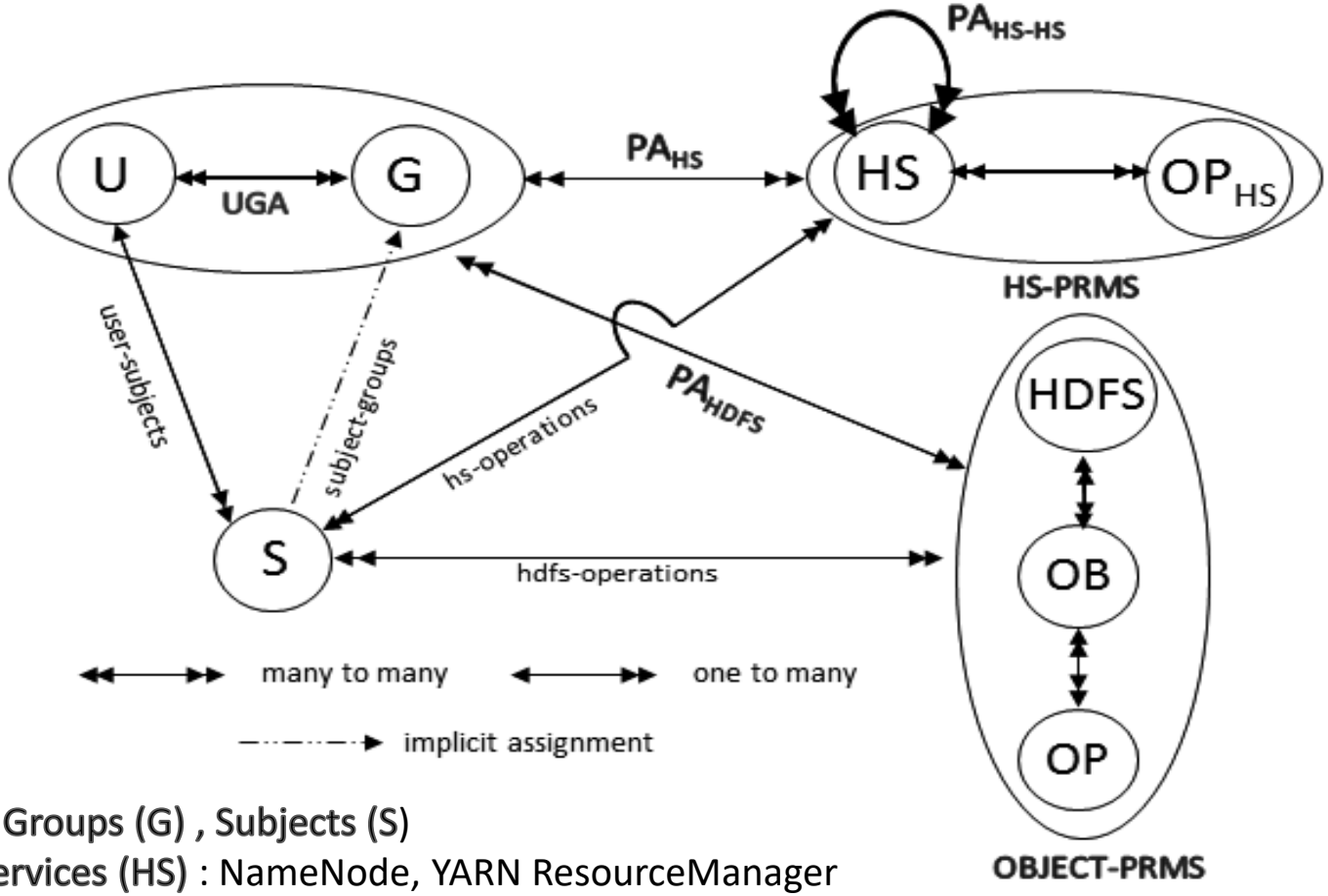
YARN Queue Access Control Configuration

Hadoop Ecosystem Authorization Architecture



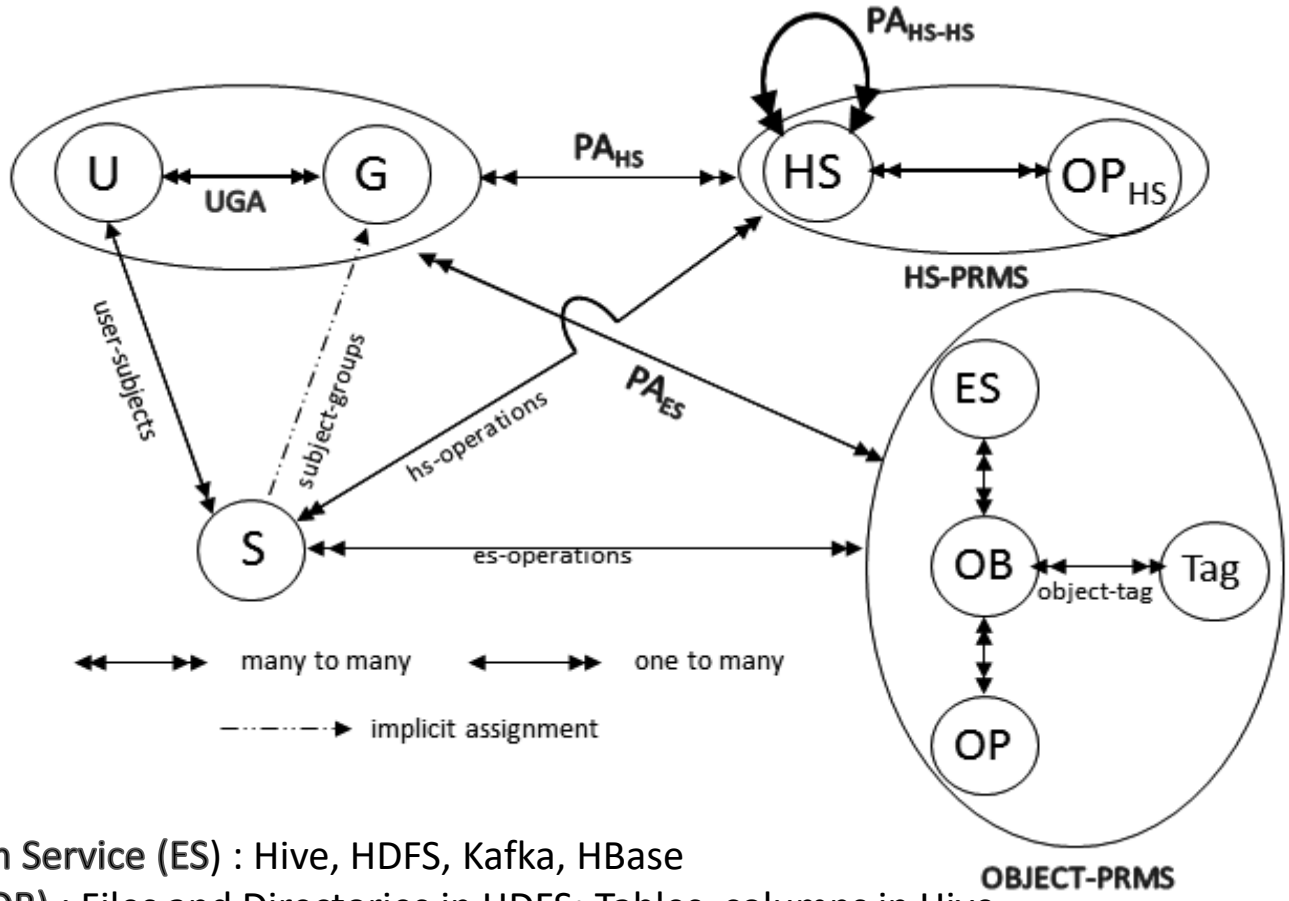
Policy Manager : Apache Ranger, Apache Sentry
Gateway : Apache Knox
Ecosystem Service (ES) : Apache Hive, HDFS, Apache Storm, Apache Kafka, YARN

AC Model: Hadoop View



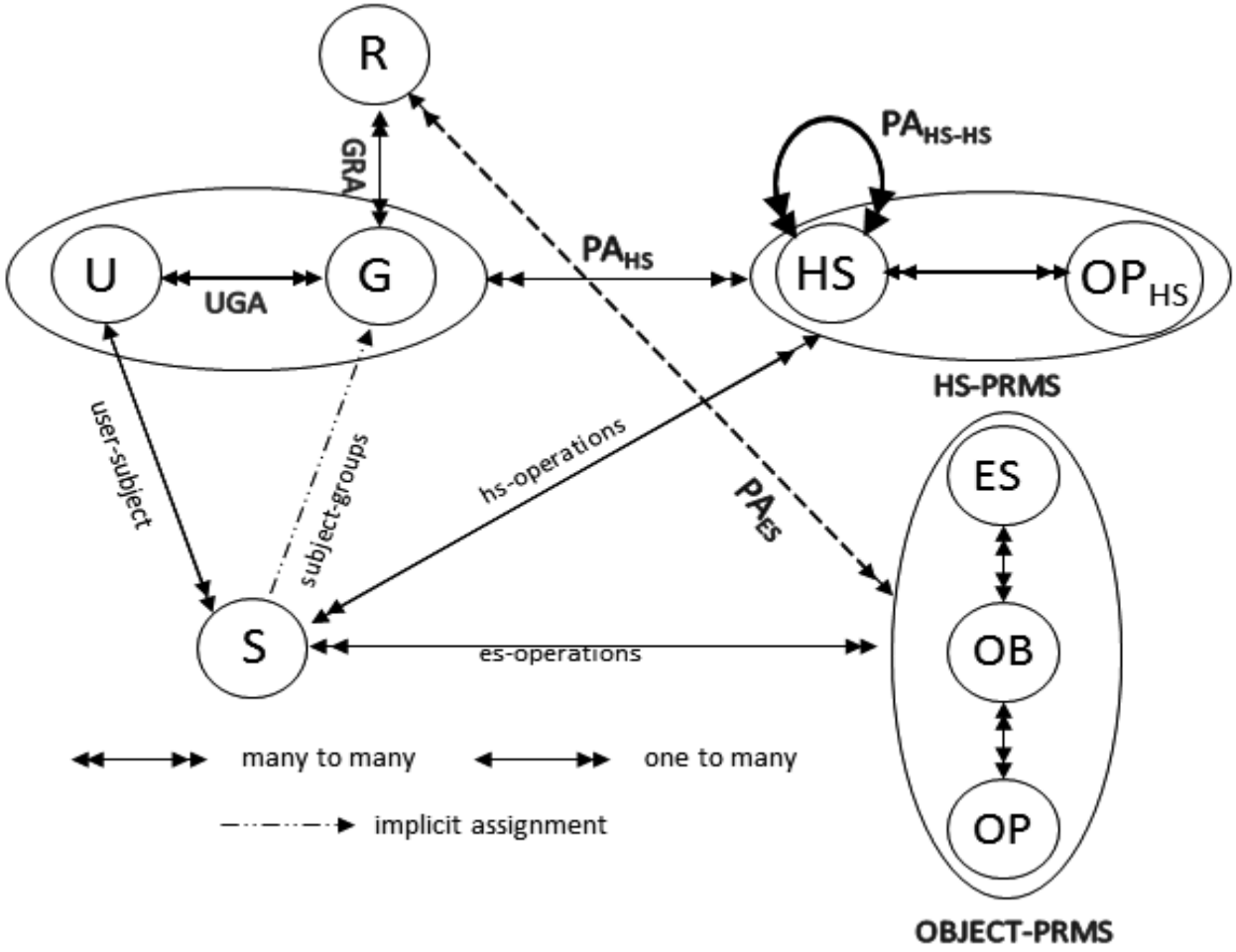
Users (U), Groups (G) , Subjects (S)
Hadoop Services (HS) : NameNode, YARN ResourceManager
Hadoop Service Operations (OP_{HS}) : access / communicate
Objects (OB) : Files and Directories in HDFS
Operations (OP) : read, write, execute

AC Model: Ranger View



Ecosystem Service (ES) : Hive, HDFS, Kafka, HBase
Objects (OB) : Files and Directories in HDFS; Tables, columns in Hive
Operations (OP) : read, write, execute, select, create
Tag : PII, top-secret

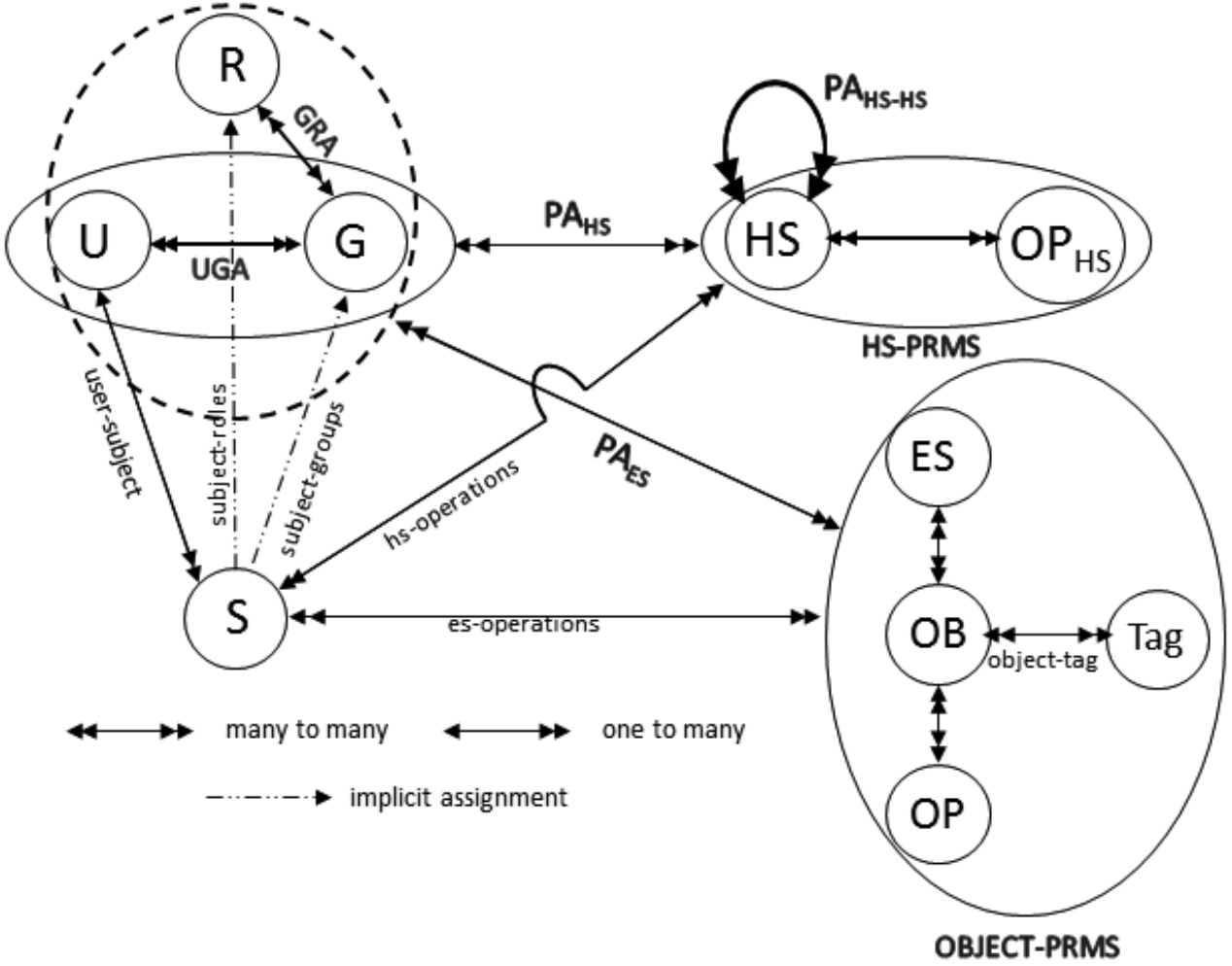
AC Model: Sentry View



Roles (R)

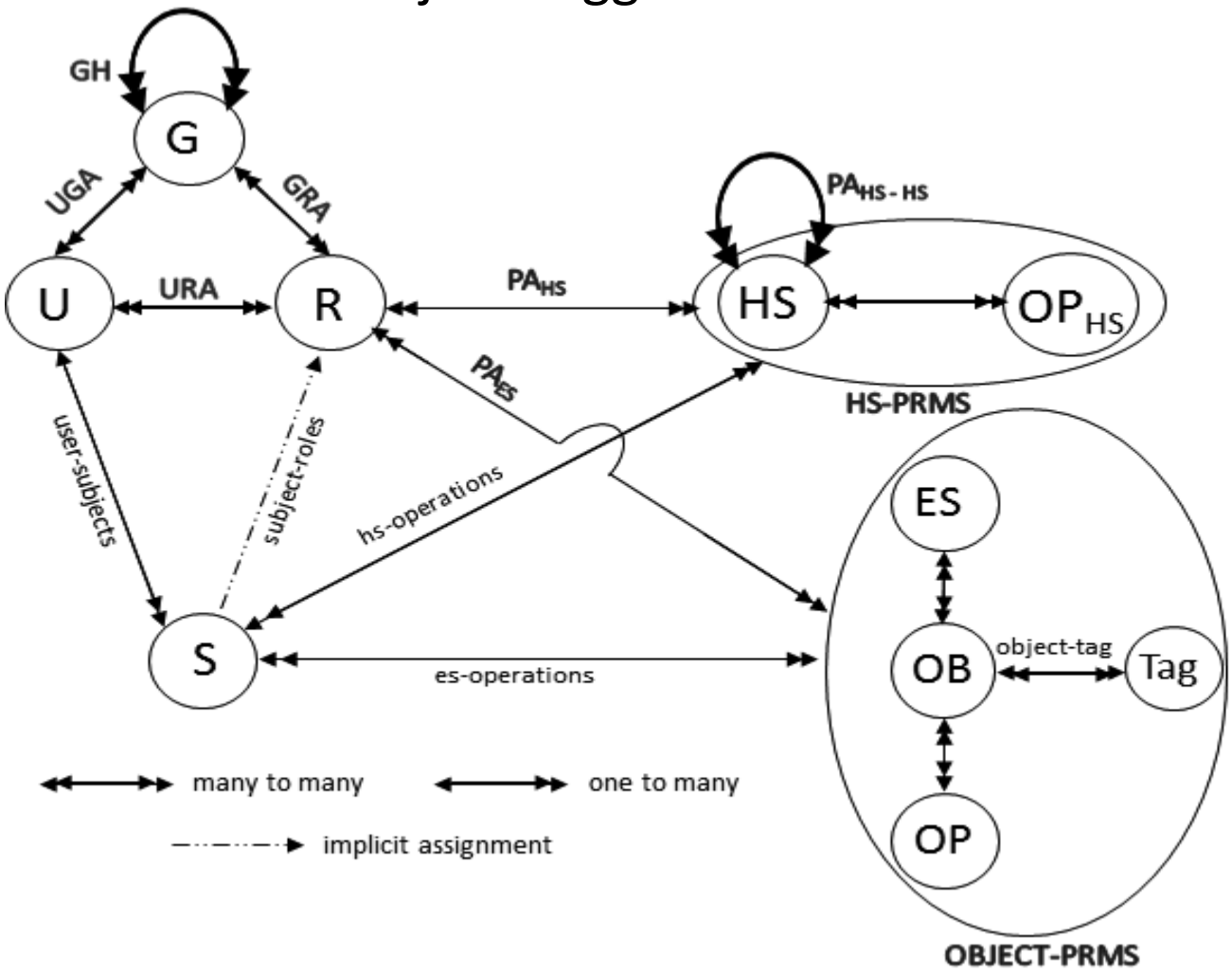
HeAC Model: Consolidated View

Hadoop Ecosystem Access Control Model

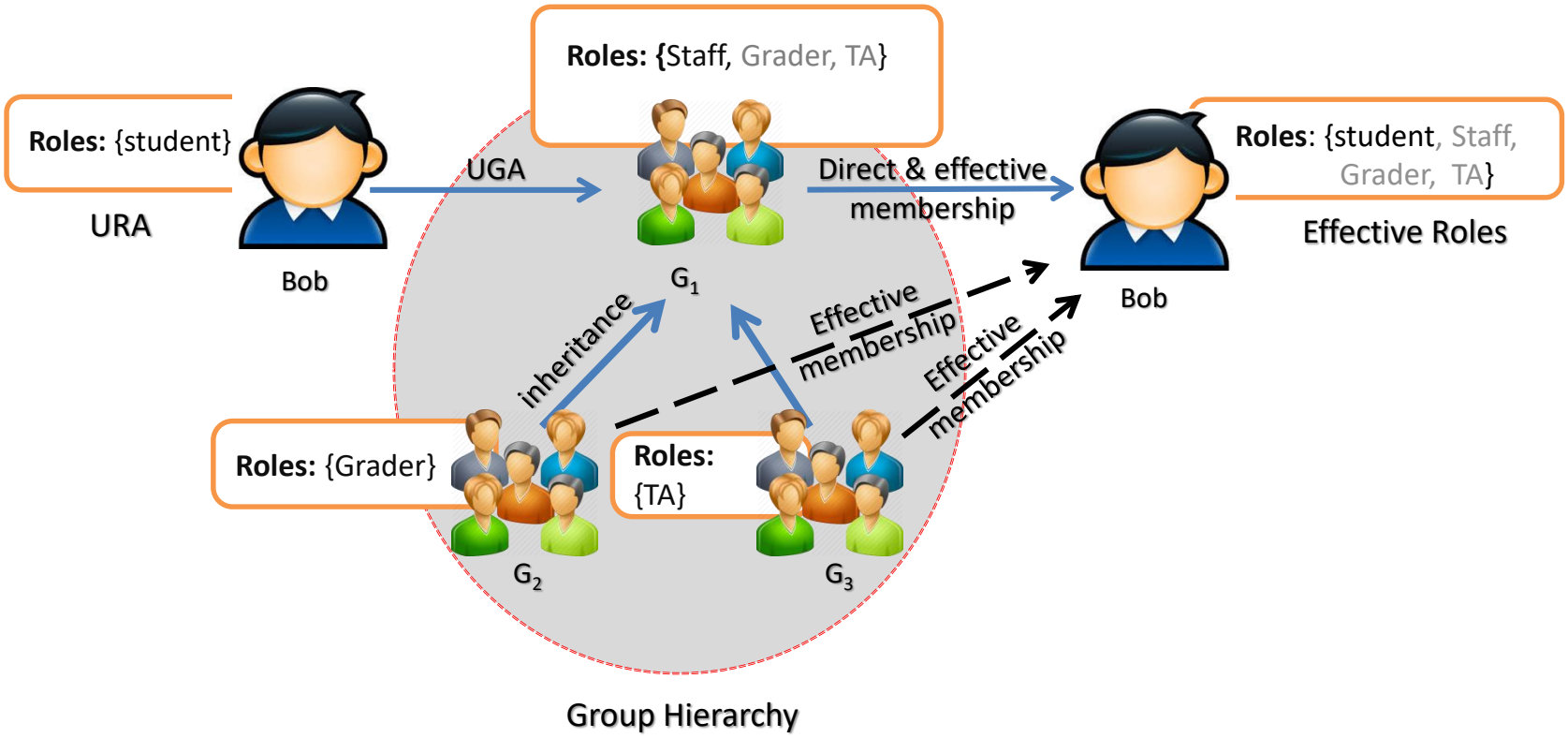


OT-RBAC Model

Object-Tagged RBAC

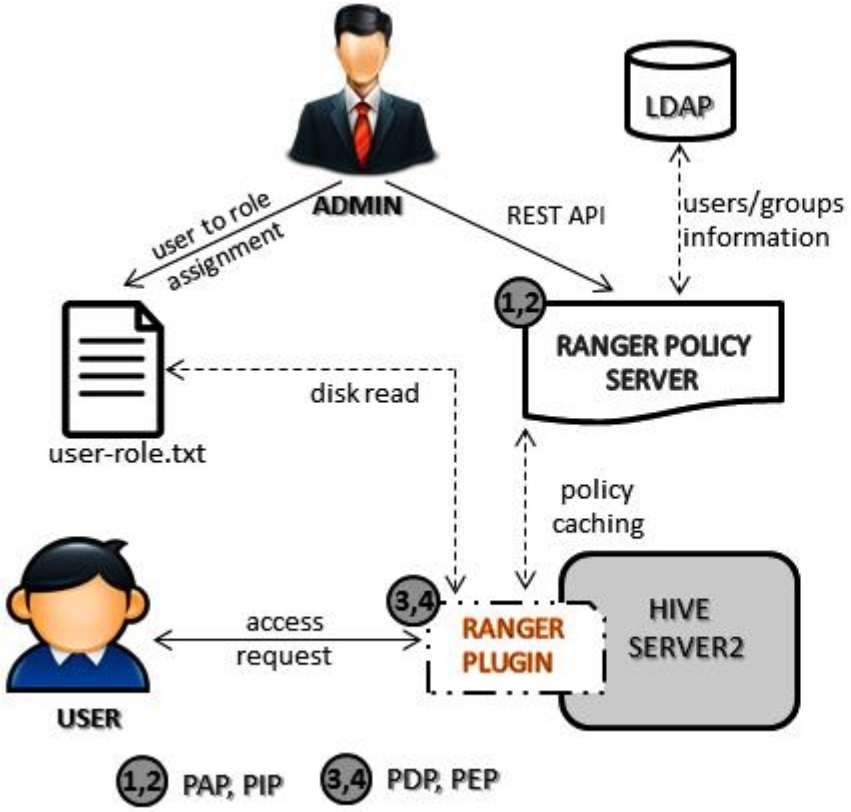


Group Based Role Inheritance



Major Benefits: Easy Administration where multiple roles can be assigned to user with single administrative operation.

Implementation Approach




```
{
  "id": 25,
  "service": "Hive",
  "name": "Sample Policy",
  "resources": {
  "database": {
      "values": [
        "foodmart"
      ],
    }
  },
  "policyItems": [
    {
      "accesses": [
        {
          "type": "select",
          "isAllowed": true
        }
      ],
      "users": [
        "user1"
      ],
      "conditions": [
        {
          "type": "Roles",
          "values": [
            "customer,owner"
          ]
        }
      }
```

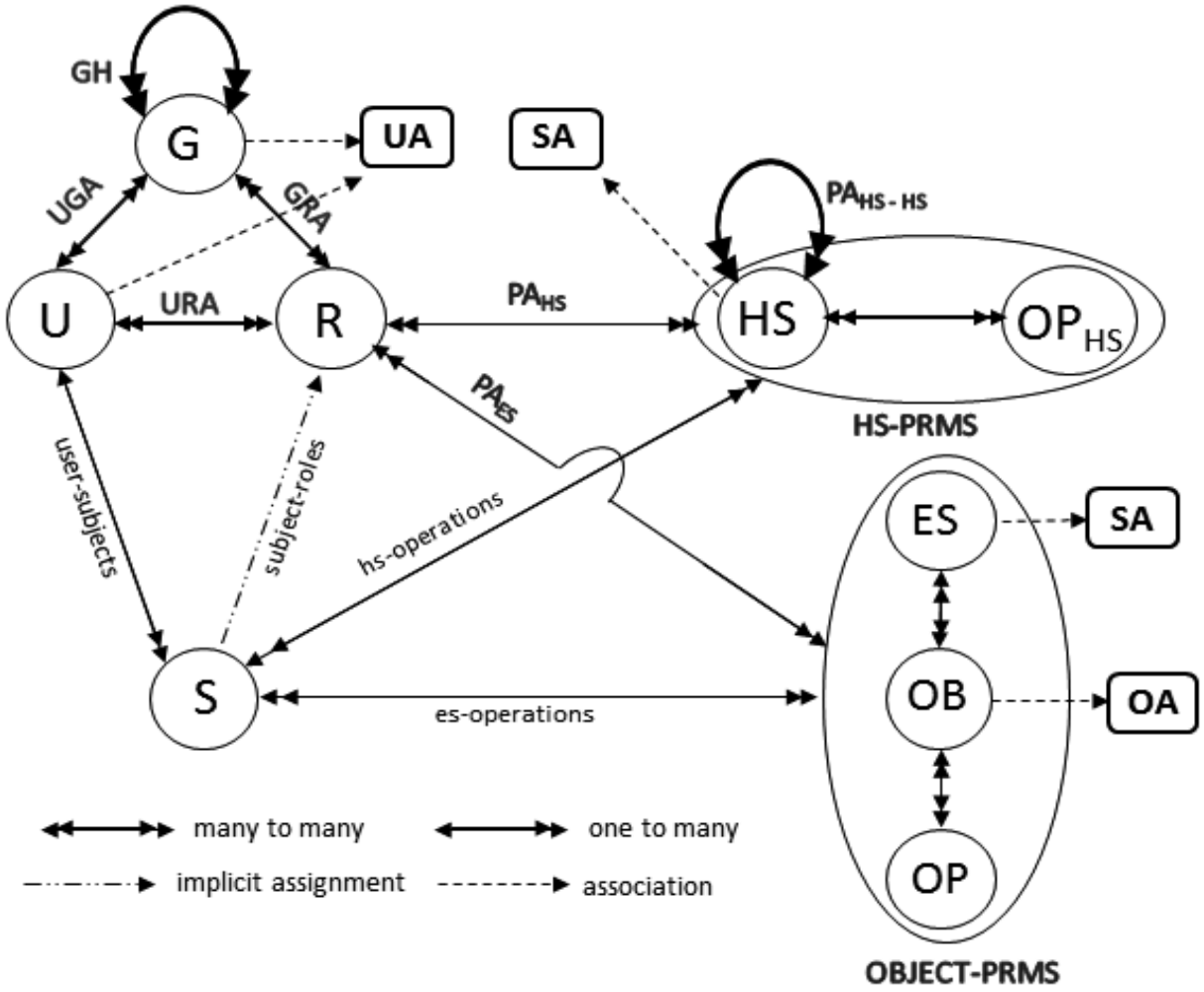
object

operation

users

roles

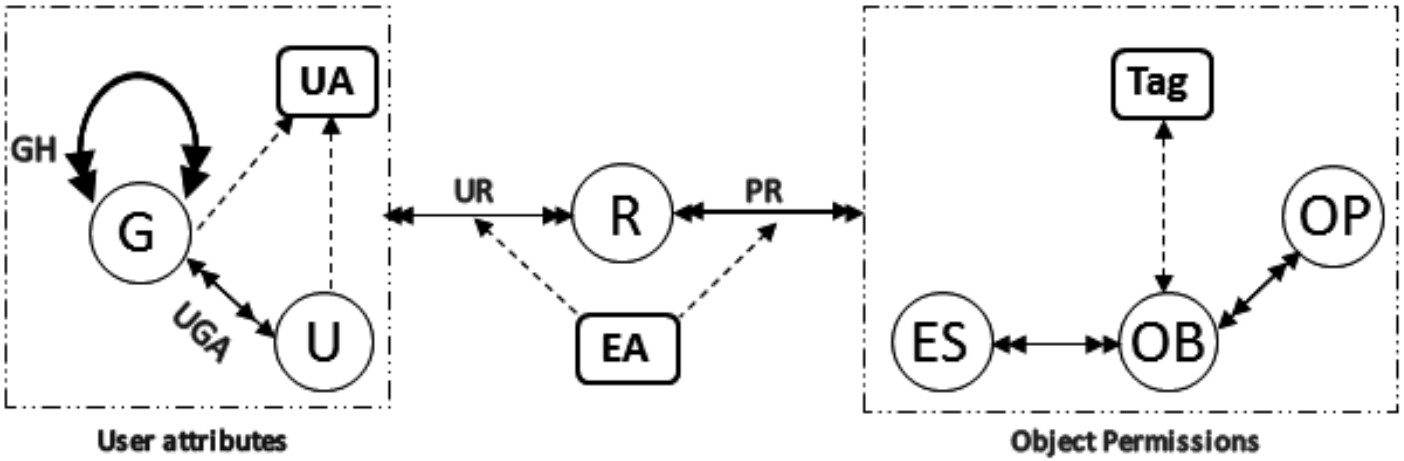
Adding Attributes to OT-RBAC



User Attributes (UA)
Service Attributes (SA)
Object Attributes (OA)

NIST Strategies

➢ Dynamic Roles



User to Role Assignment:

jobTitle(u_1) = director ∧ optMode = normal → Admin

jobTitle(u_1) = director ∧ optMode = emergency → Faculty

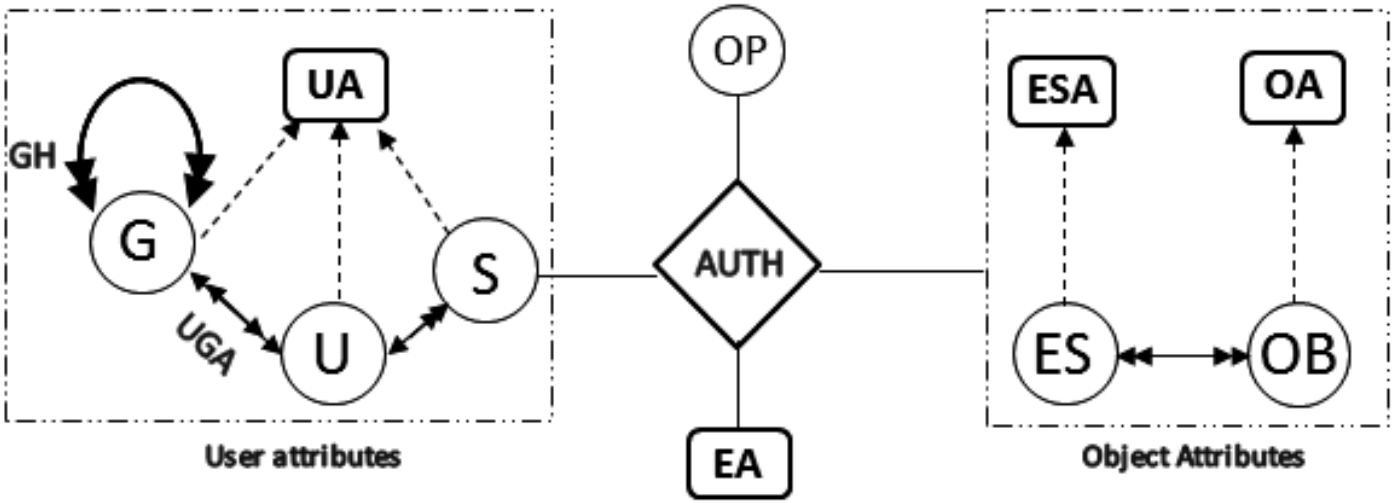
Permission to Role Assignment: Permission P_1 = (es,op,tag), (ob,tag) ∈ object-tag

tag= PII ∧ op = write:: P_1 → Admin

tag= PCI ∧ op = write:: P_1 → Faculty

NIST Strategies

➢ Attribute Centric

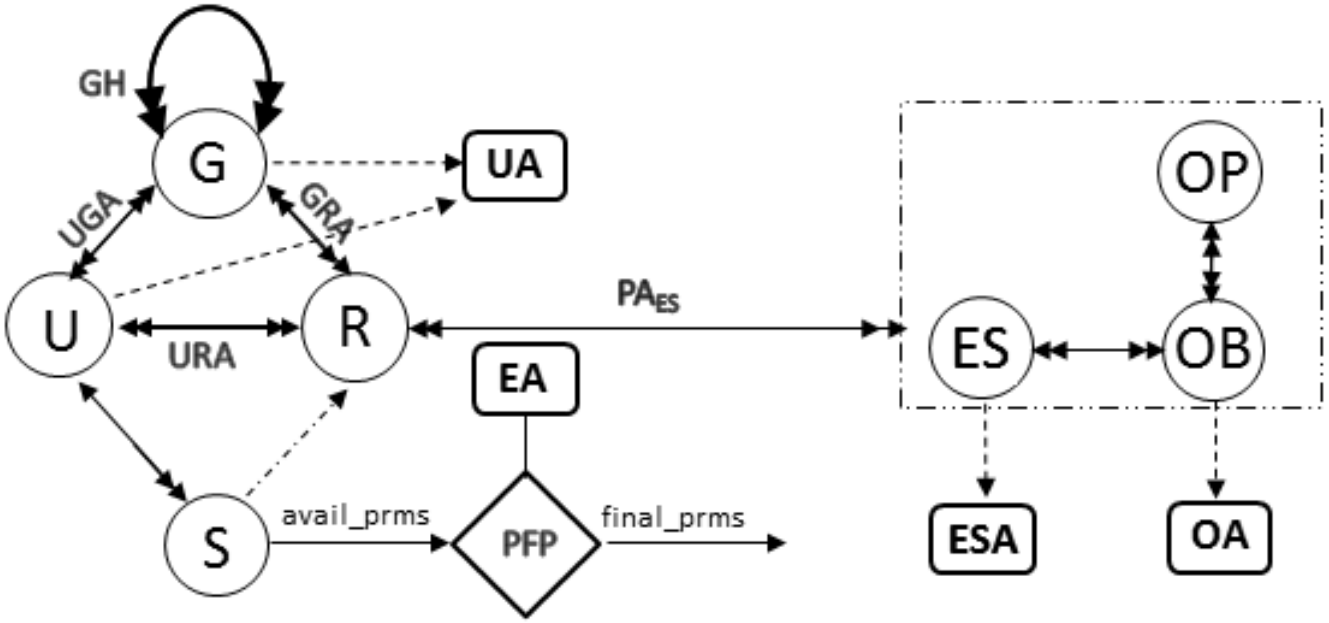


$Authorization_{write}$ (s:S, es:ES, ob:OB) :: $effective_{jobTitle}(s)$ = director ∧ access(s,es) = True ∧ name(es) = hdfs ∧ tag(ob)= PII ∧ optMode = normal

$Authorization_{read}$ (s:S, es:ES, ob:OB) :: $effective_{jobTitle}(s)$ = professor ∧ access(s,es) = True ∧ name(es) = hdfs ∧ tag(ob)= PCI ∧ optMode = emergency.

NIST Strategies

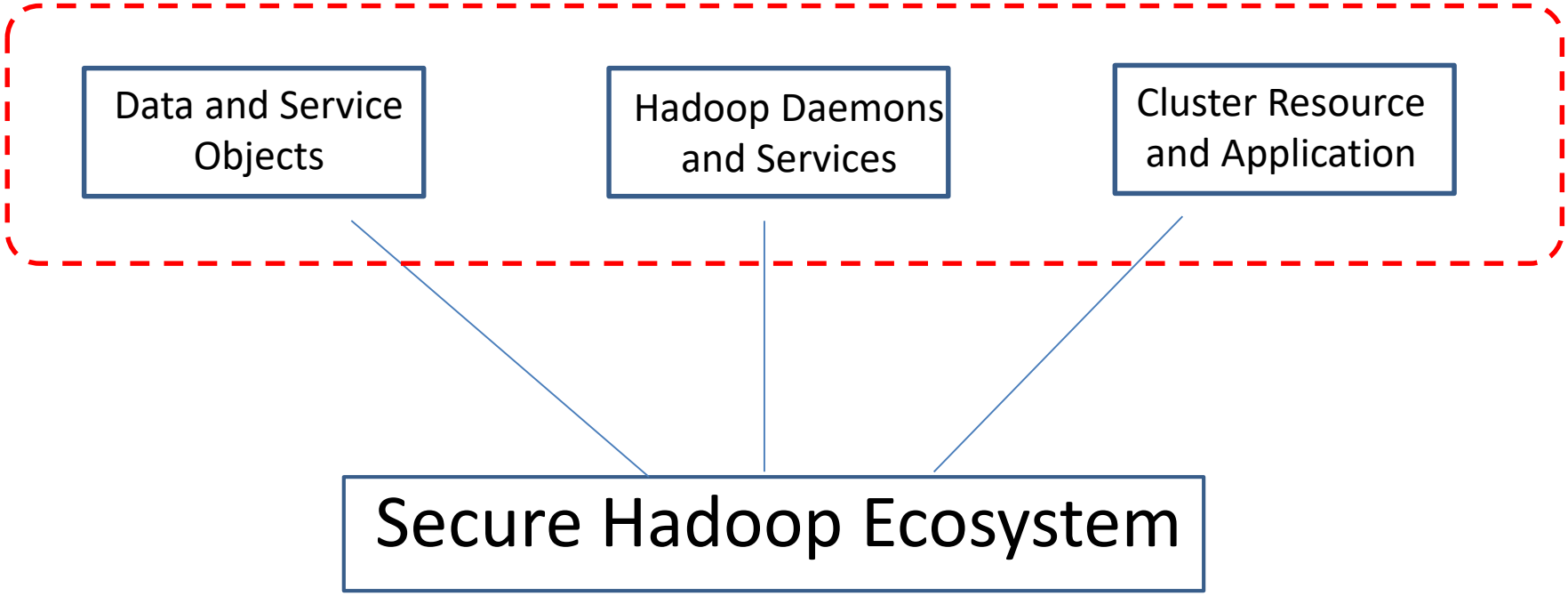
- Role Centric



FAdmin1(s:S, es:ES, ob:OB, write) ::
jobTitle(subUser(s)) = director ∧ optMode = normal

FAdmin2(s:S, es:ES, ob:OB, read) ::
jobTitle(subUser(s)) = faculty ∧ tag(ob) = PCI

Defense in Depth



Conclusion and Future Work

- Formalized Conceptual HeAC Model
- Object-Tagged-RBAC Model
- Attributes based extensions

Some Future Goals:

- Introduce Data ingestion security
- Privacy concerns and finer grained approaches in Multi-Tenant Hadoop Lake