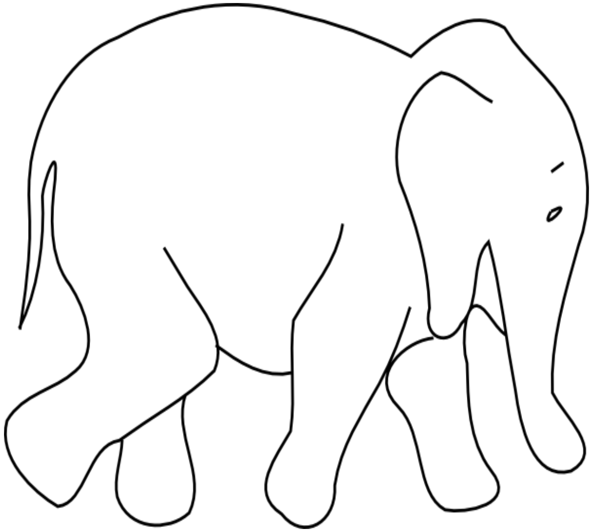
Enumerated Authorization Policy ABAC Models: Expressive Power and Enforcement

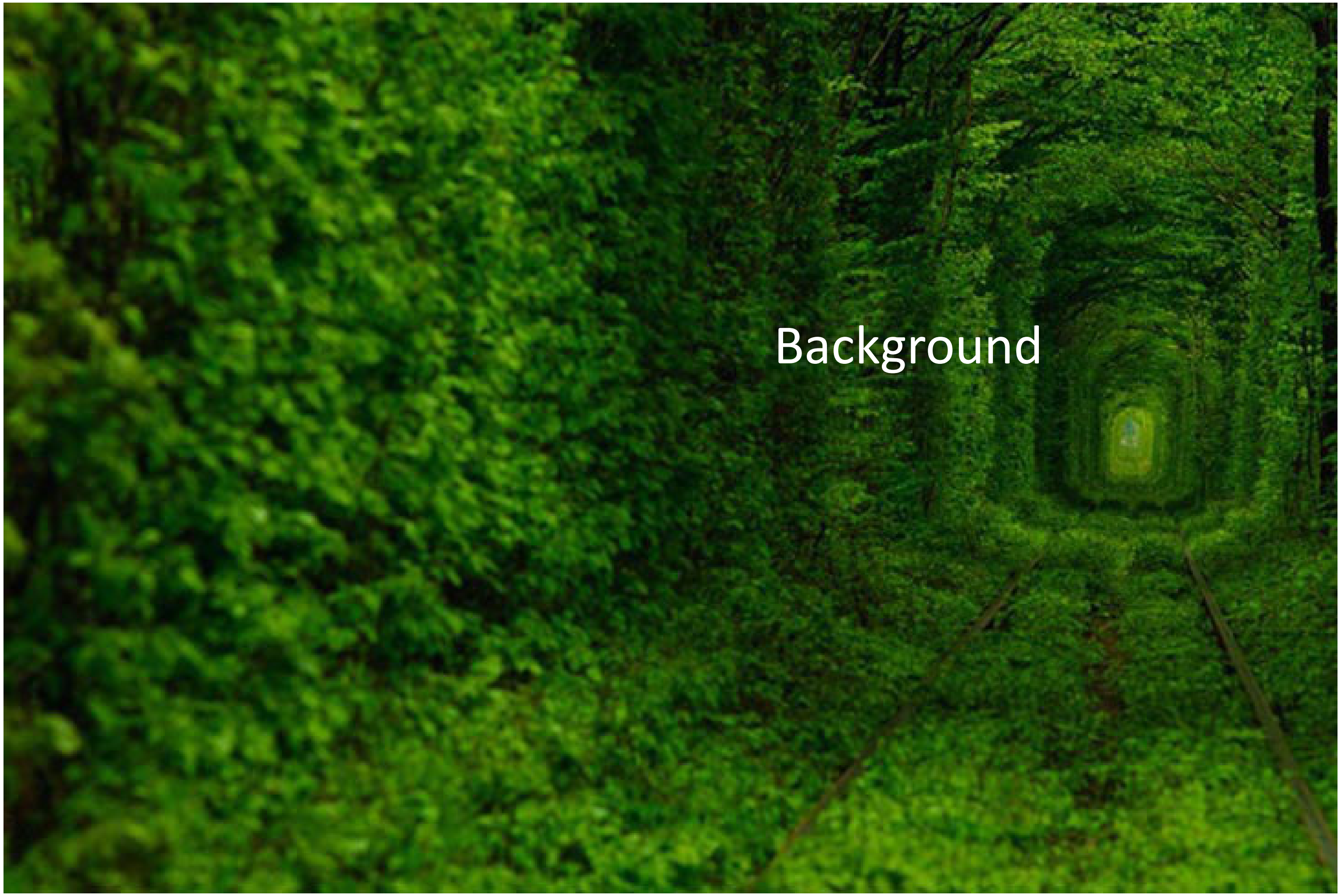
Dissertation Defense
Prosunjit Biswas

Advisor: Prof. Ravi Sandhu
Prof. Gregory White
Prof. Jianwei Niu
Prof. Palden Lama
Prof. Ram Krishnan

Outline

- Background
- Enumerated Authorization-policy Models
- Enumerated vs Logical-formula Authorization-policy models
- Enforcement of Enumerated Authorization-policy models
- Conclusion





Background

Logical-formula as authorization policy

Usually, propositional logic is used to set up authorization policies.

Example:

Can-download ≡ $age(u) > 18 \wedge movie\text{-}rating(o) = R$

Advantages

- easy to set-up
- concise
- very expressive

Logical-formula as authorization policy

Many ways to set up an authorization policy.

e.g. consider a policy, $Auth_{read}$ that allows a manager to read TS objects from home or office

i. $Auth_{read} \equiv role(u) = mng \land (location(u) = office \lor location(u) = home) \land sensitivity(o) = TS$

ii. $Auth_{read} \equiv (role(u) = mng \land location(u) = office \land sensitivity(o) = TS) \lor (role(u) = mng \land location(u) = home \land sensitivity(o) = TS)$

Logical-formula as authorization policy

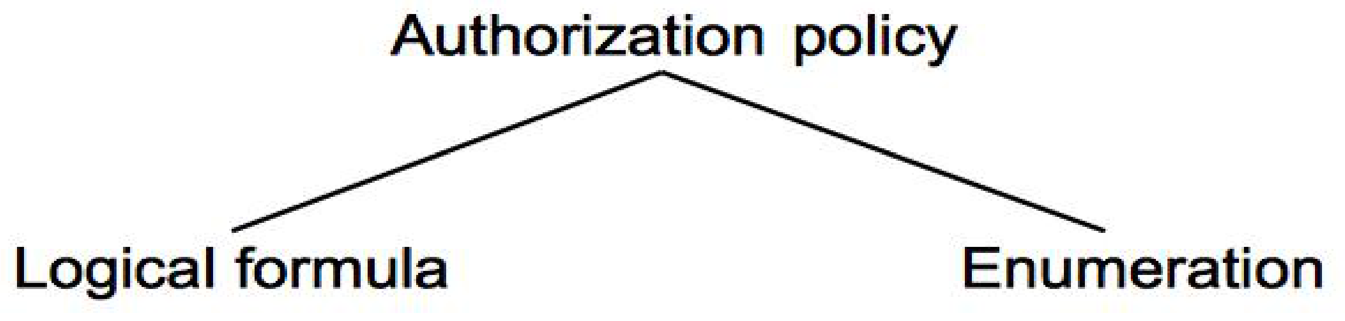
Many ways to administer same changes.

e.g. update $\text{Auth}_{\text{read}}$ policy so that manager can no-longer access from home.

i. role(u) = mng ∧ (location(u) = office ∨ location(u) = home) ∧ sensitivity(o) = TS

ii. (role(u) = mng ∧ location(u) = office ∧ sensitivity(o) = TS) ∨ (role(u) = mng ∧ location(u) = home ∧ sensitivity(o) = TS)

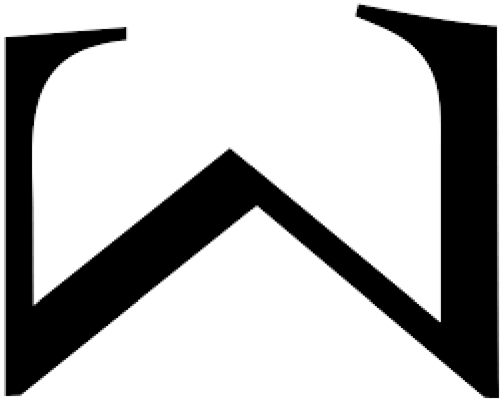
Logical-formula vs enumerated policy


Authorization policy

Logical formula

Enumeration

- Boolean expression
- E.g.: age(u)>18
- Models: ABAC$_\alpha$, HGABAC



- Set of tuples
- {(age(u),19), (age(u),20), …. (age(u),100)} *[assuming range upper bound <=100]*
- *Models: Policy Machine, 2-sorted-RBAC*

Problem and thesis statement

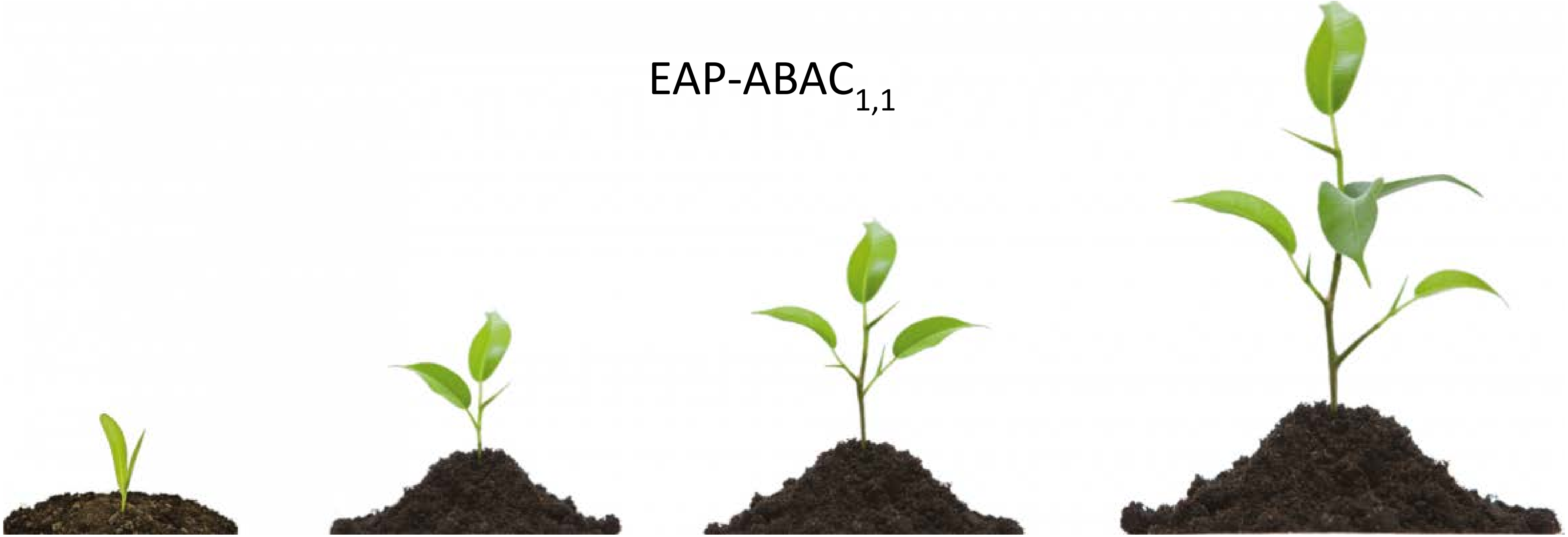
Problem statement

There are two major techniques for specifying authorization policies in Attribute Based Access Control (ABAC). The more conventional approach is to define policies using logical formulas involving attribute values. The alternate technique is by enumeration. While considerable work has been done for the former approach, the later lacks fundamental work from the research community.

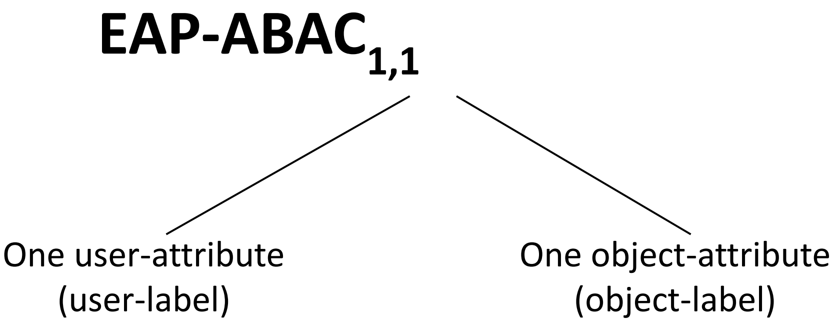
Thesis statement

Enumerated Authorization-Policy ABAC (EAP-ABAC) is a viable alternate to Logical-formula Authorization Policy ABAC (LAP-ABAC). EAP-ABAC is as expressive as LAP-ABAC in the finite domain. EAP-ABAC models can be enforced in different application domains.

EAP-ABAC$_{1,1}$



EAP-ABAC$_{1,1}$



EAP-ABAC$_{1,1}$

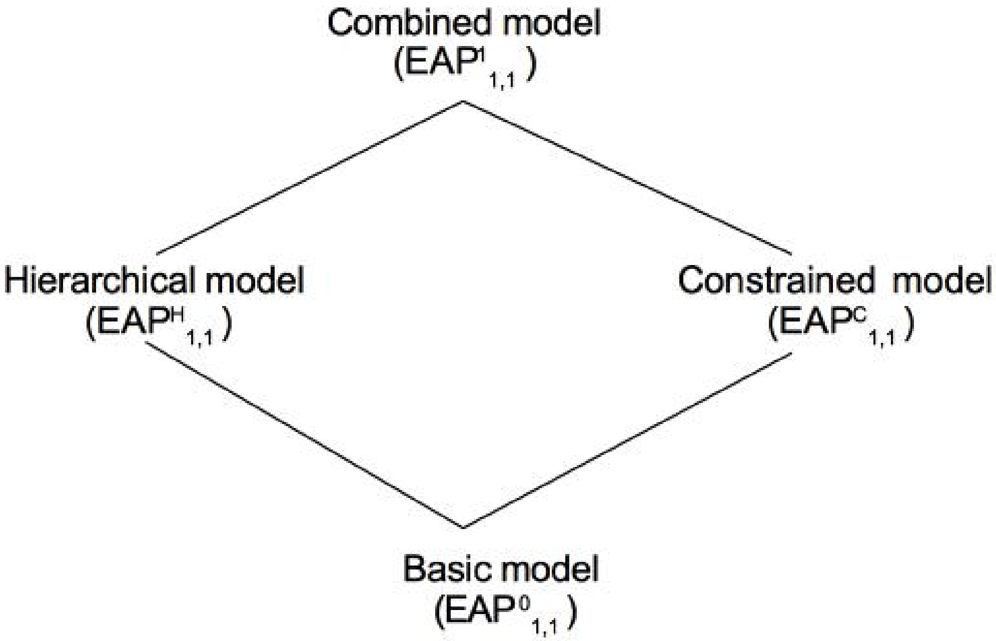
One user-attribute (user-label)

One object-attribute (object-label)

Salient Features:

- Very Simple enumerated ABAC model
- Finite domain ABAC model

EAP-ABAC$_{1,1}$ Family



Combined model
($EAP^{1}_{1,1}$)

Hierarchical model
($EAP^{H}_{1,1}$)

Constrained model
($EAP^{C}_{1,1}$)

Basic model
($EAP^{0}_{1,1}$)

$\text{EAP}_{1,1}$: Core model

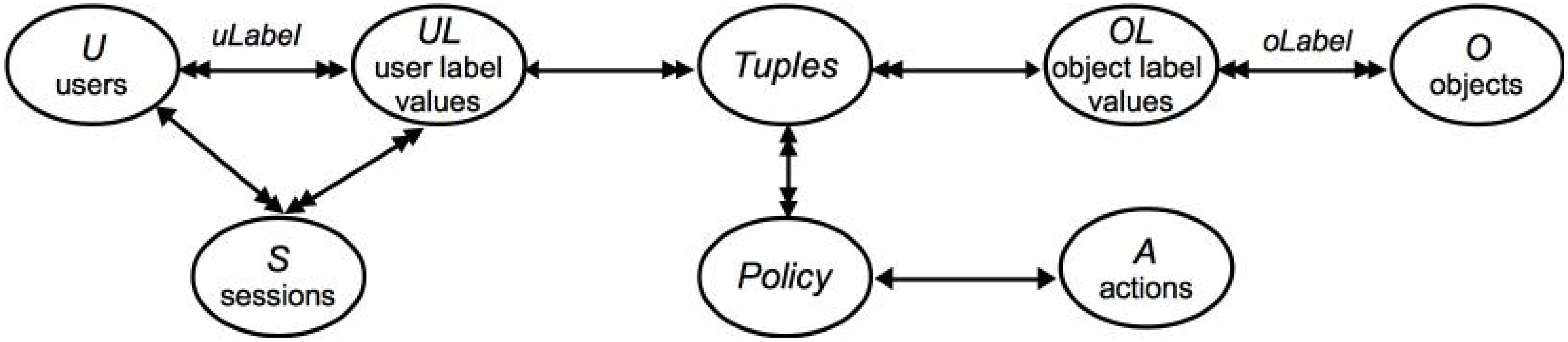


Figure 1: $\text{EAP}_{1,1}$ model

Salient Characteristics:

- One user and object attribute
- Atomic valued tuples
- Tuples represent micro-policies

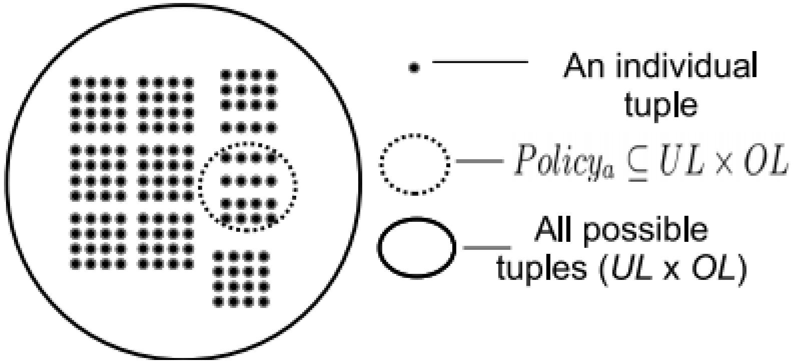

An individual tuple
$Policy_a \subseteq UL \times OL$
All possible tuples (UL x OL)

Figure 2: Policy vs tuples

Examples:

UL={manager,employee}
OL={TS,S}
Tuple1= (manager,TS)
$\text{Policy}_{\text{read}}$ = {tuple1, tuple2…}

$EAP_{1,1}$: Hierarchical model

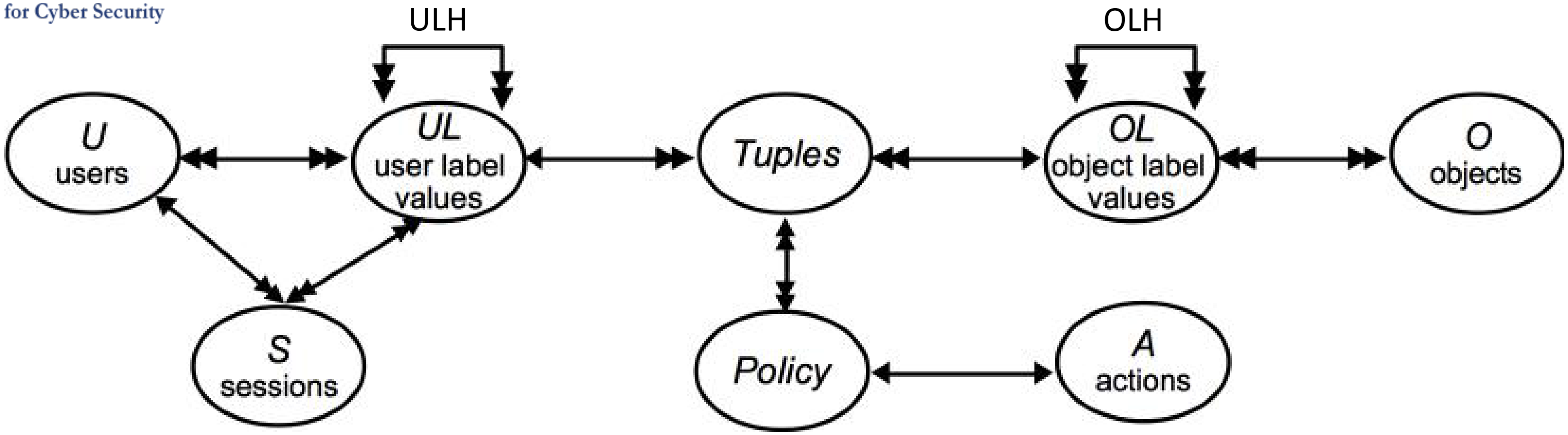


Figure 3: Hierarchical model

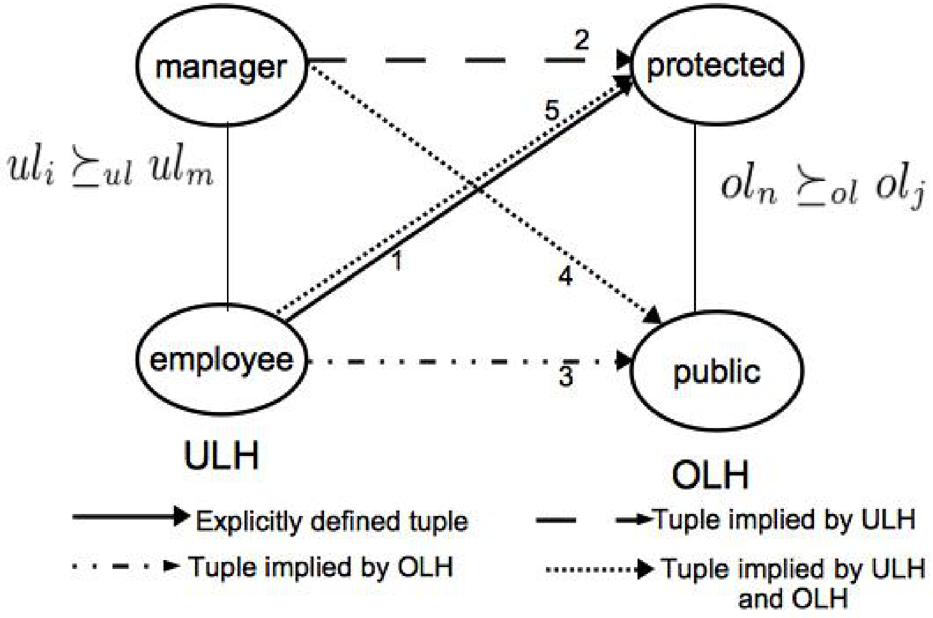


Figure 4: Attribute hierarchy

Examples

ULH={(manager,employee)}

OLH={(protected, public)}

$Policy_a$ = {(employee,protected)}

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

EAP$_{1,1}$: Constrained model

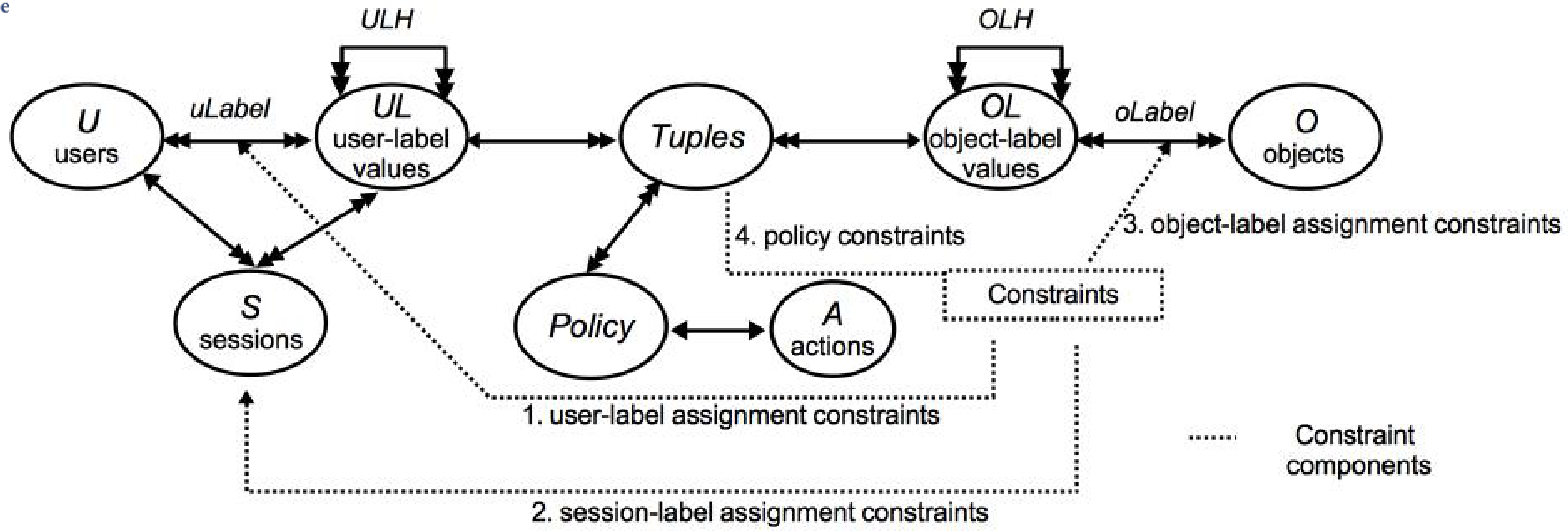


Figure 5: Constrained model

<u>Examples</u>

uLabel assignment constraint: eg. a user cannot be both manager & director.

Session assignment constraint: eg. at most one value can be activated in a session.

oLabel assignment constraint: eg. an object cannot be both private & public

Policy constraints: eg. (employee, TS) can never be used.

Relationship of $\text{EAP}_{1,1}$ with traditional models

Expressiveness of $\text{EAP}_{1,1}$ models

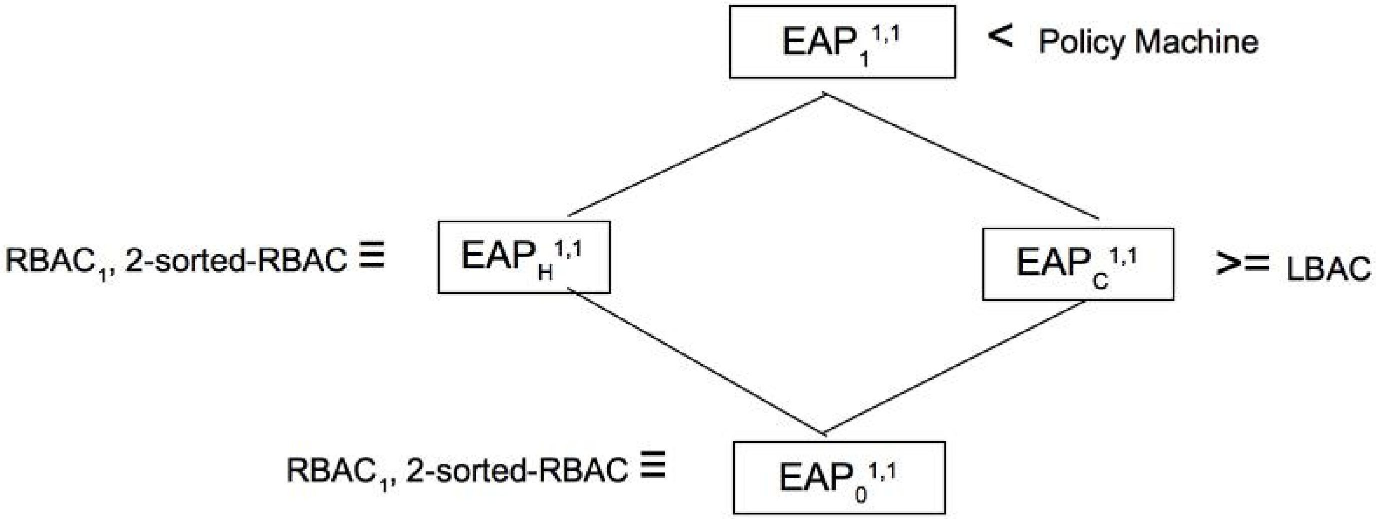


Figure 6: Expressive power of $\text{EAP}_{1,1}$ family

$EAP_{1,1}$ as an instance of Policy Machine

Policy Machine $_{mini}$

- Only ASSIGN and ASSOCIATION relation
- Default policy class

Configuration of $EAP_{1,1}$ in Policy Machine $_{mini}$

$EAP_{1,1}$ equivalent to 2-sorted-RBAC

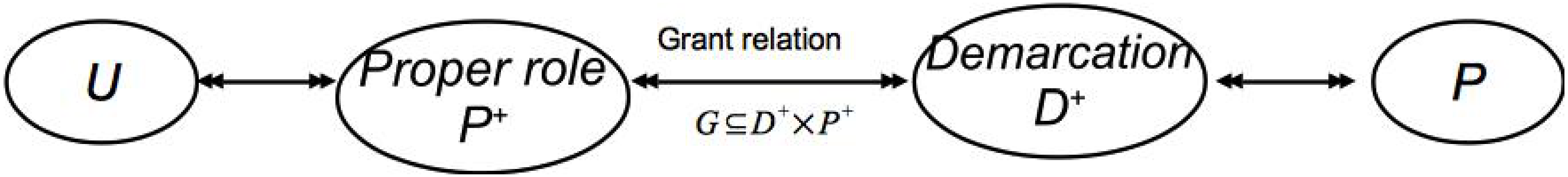


Figure 9: 2-sorted-RBAC

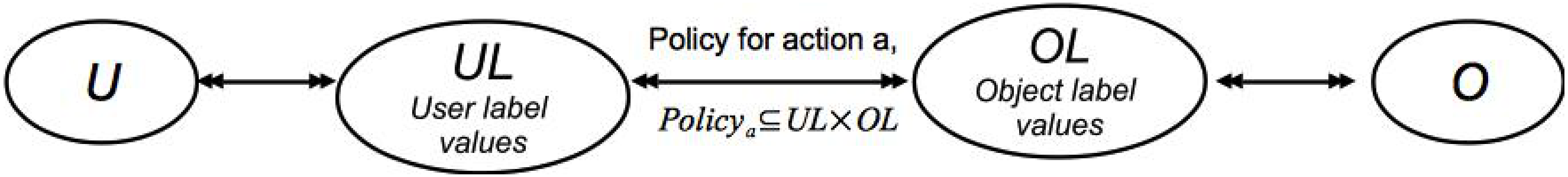


Figure 10: 2-sorted-RBAC in $EAP_{1,1}$

2-sorted-RBAC vs $EAP_{1,1}$:

Use of attributes

Separation of object and action from permission

LBAC in $EAP_{1,1}$

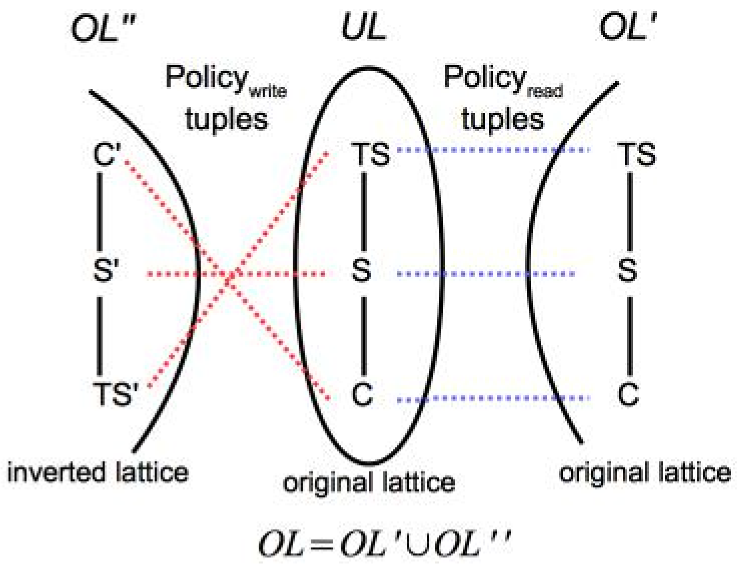
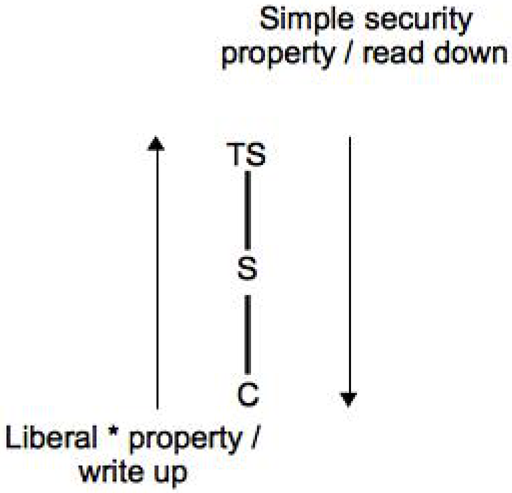


Figure 7: LBAC in EAP $_{1,1}$



Simple security property / read down
TS
S
C
Liberal * property / write up

Figure 8: LBAC properties

LBAC assumptions:

- Tranquility
- Object operation: creation only

EAP-ABAC$_{m,n}$ and LAP-ABAC$_{m,n}$

EAP$_{m,n}$: Multi-attribute EAP model

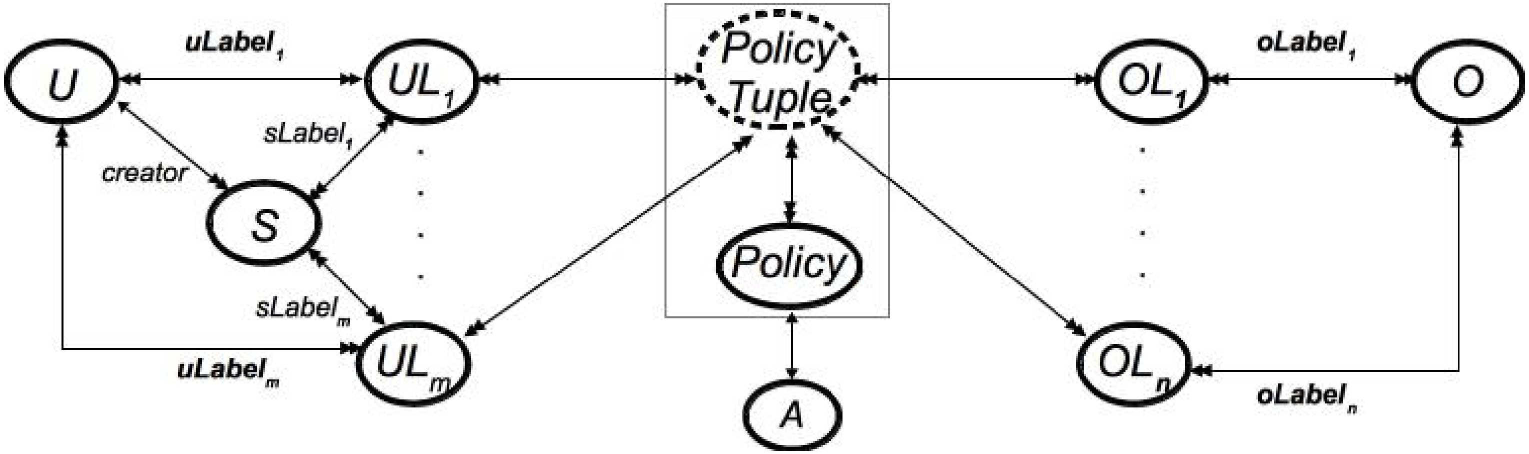


Figure 11: EAP$_{m,n}$ model

Examples:

role={manager,employee}
Clearance = {TS,S}
Resource = {VM, network}
Security-label = {Sensitive, public}

tuple1 = ({manager}, {TS}, {VM}, {Sensitive})
Can-read ≡ {tuple1, tuple2,...}

Salient Characteristics:

- m user and n object attributes
- set valued tuples
- Tuples represent micro-policies

$\text{LAP}_{m,n}$: Multi-attribute LAP model

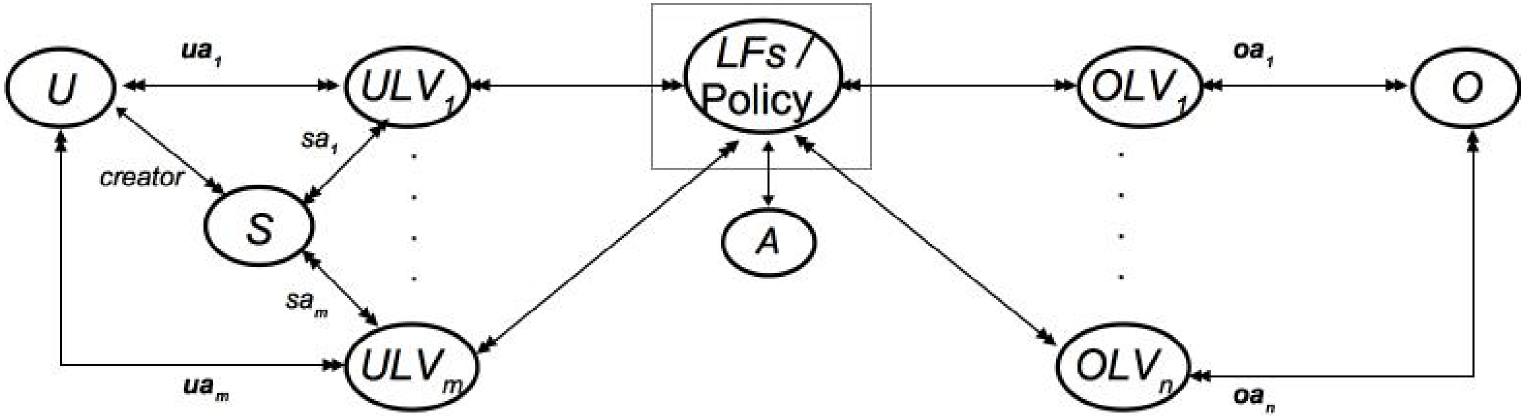


Figure 12: $\text{LAP}_{m,n}$ model

Examples:

role={mng, emp}
Clearance = {TS,S}
Resource = {VM, network}
Security-label = {Sensitive, public}

can-read ≡ role(u)=mng ∧ clearance(u) = TS ∧ resource(o) = VM ∧ security-label(o) = sensitive

Salient Characteristics:

- m user and n object attributes
- logical-formula presents policies

Equivalence of expressive power

Equivalence of expressive power (summary)

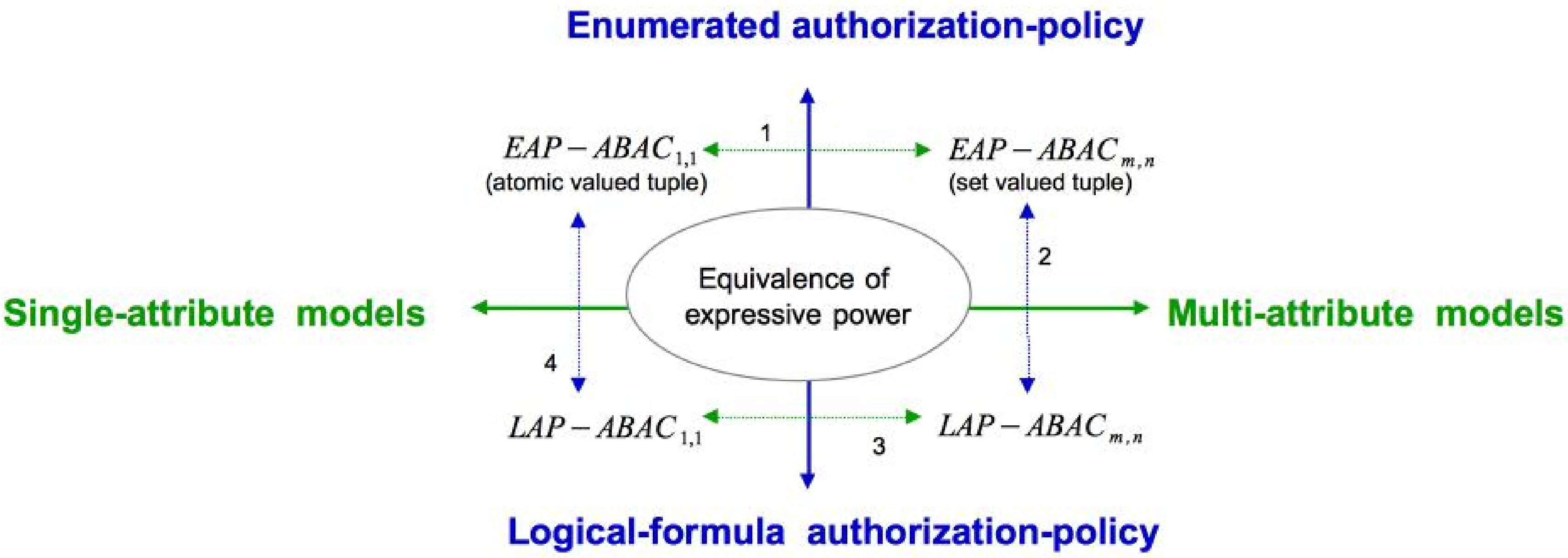


Figure 13: Equivalence of enumerated and logical-formula authorization-policy models

Example: expressing $\text{LAP}_{m,n}$ policy in $\text{LAP}_{1,1}$

Authorization policy in $\text{LAP}_{m,n}$:

Role = {mng, emp}
Location = {campus, home}
Resource = {vm, network}
can-run ≡ role(u)=mng ∧ location(u) = campus ∧ resource(o) = VM

Equivalent policy in $\text{LAP}_{1,1}$:

Role-location = {mng-from-campus, mng-from-home, emp-from-campus, emp-from-home}
Resource = {vm, network}
can-run ≡ Role-location(u) = mng-from-campus ∧ resource(o) = VM

Example: expressing $LAP_{1,1}$ policy in $EAP_{1,1}$

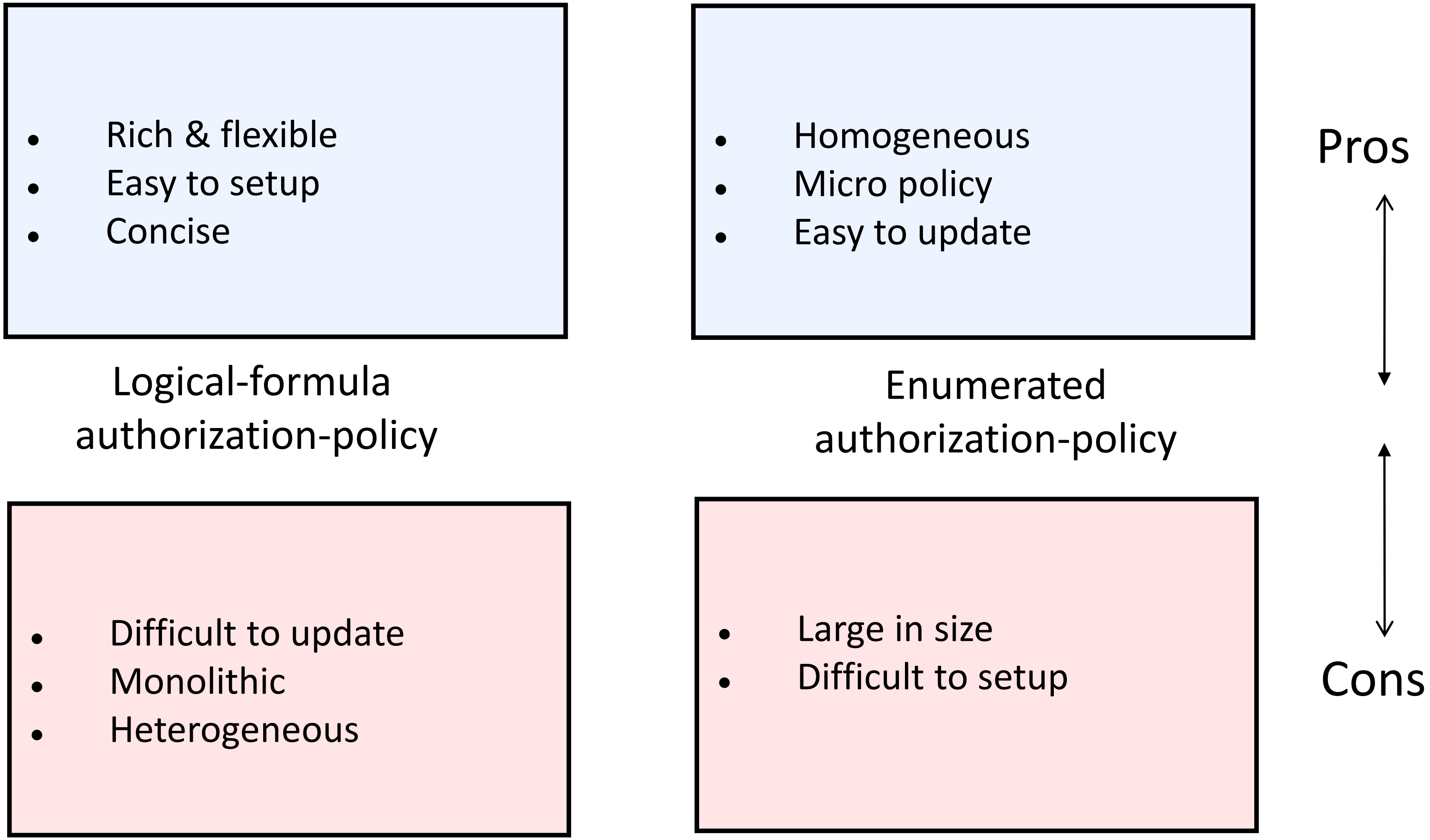
Authorization policy in $LAP_{1,1}$:

Age = {1,2,3,...,100}
Movie-type = {pg, pg-13, R}
can-download ≡ age(u)>=18 ∧ age(u)<25 ∧ movie-type(o) = R

Equivalent policy in $EAP_{1,1}$:

Age = {1,2,3,...,100}
Movie-type = {pg, pg-13, R}
can-download ≡ { (18,R), (19,R), (20,R), (21,R), (22,R), (23,R), (24,R)}

Beyond Expressive power



- Rich & flexible
- Easy to setup
- Concise

Logical-formula authorization-policy

- Difficult to update
- Monolithic
- Heterogeneous

- Homogeneous
- Micro policy
- Easy to update

Enumerated authorization-policy

- Large in size
- Difficult to setup

Pros

Cons

Enforcement

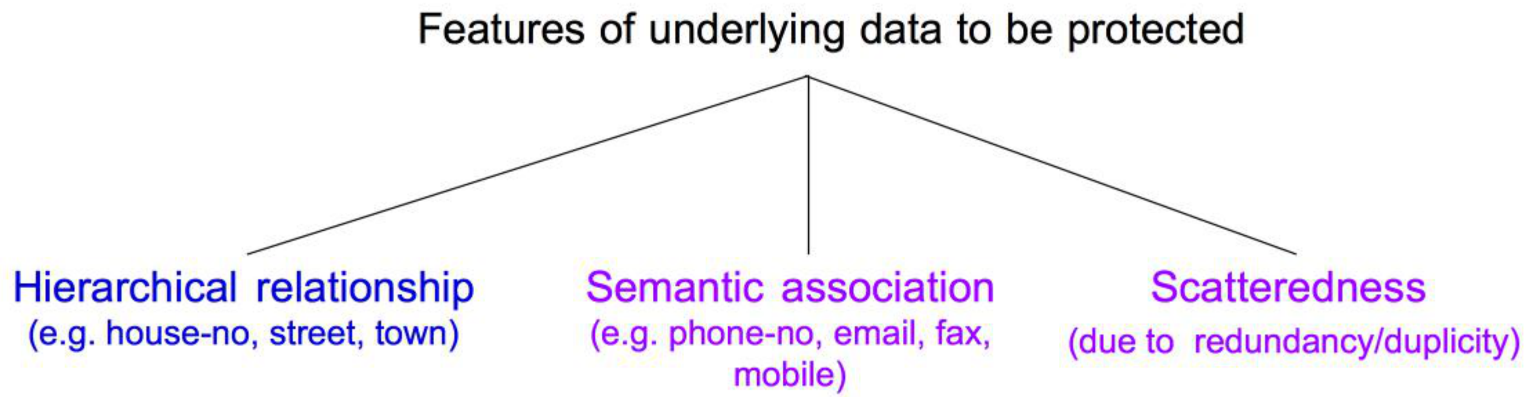
Enforcement

Protection model for JSON documents

Motivation (why?)

Why JSON?

Why not reuse XML protection models?



Features of underlying data to be protected

Hierarchical relationship
(e.g. house-no, street, town)

Semantic association
(e.g. phone-no, email, fax, mobile)

Scatteredness
(due to redundancy/duplicity)

- Considered in XML protection models

- Considered in our proposed model but not in XML protection models.

Existing XML models vs proposed model

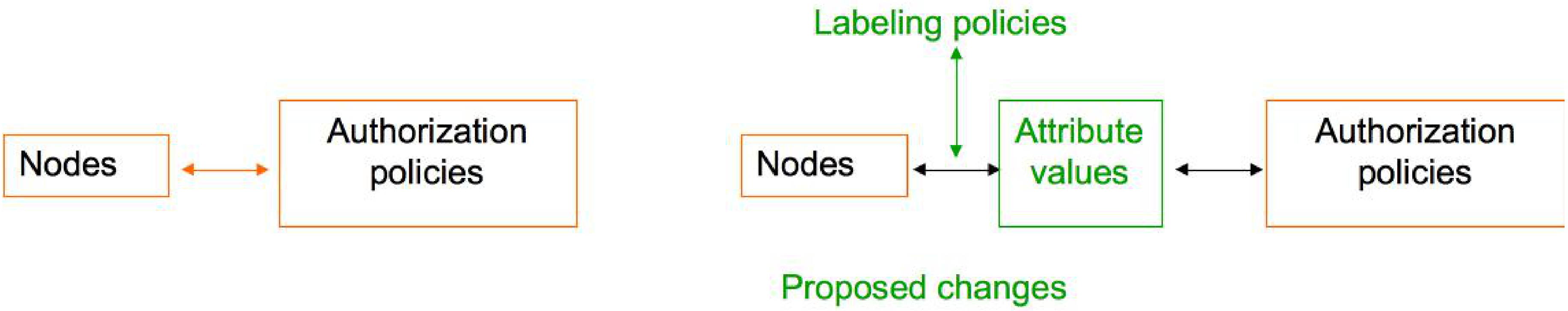


Figure 14: XML vs proposed JSON protection model

Background - JSON

JSON data forms a rooted tree hierarchical structure (like XML)

```
{
"emp-rec":{
"name": "...",
"con-info":{
        "email": "...",
        "work-phone": "..."
        },
"emp-info":{
        "mobile": "...",
        "EID":  "...",
        "salary": "..."
}
"sen-info": {
        "SSN": "...",
        "salary": "..."
        }
}
}
```

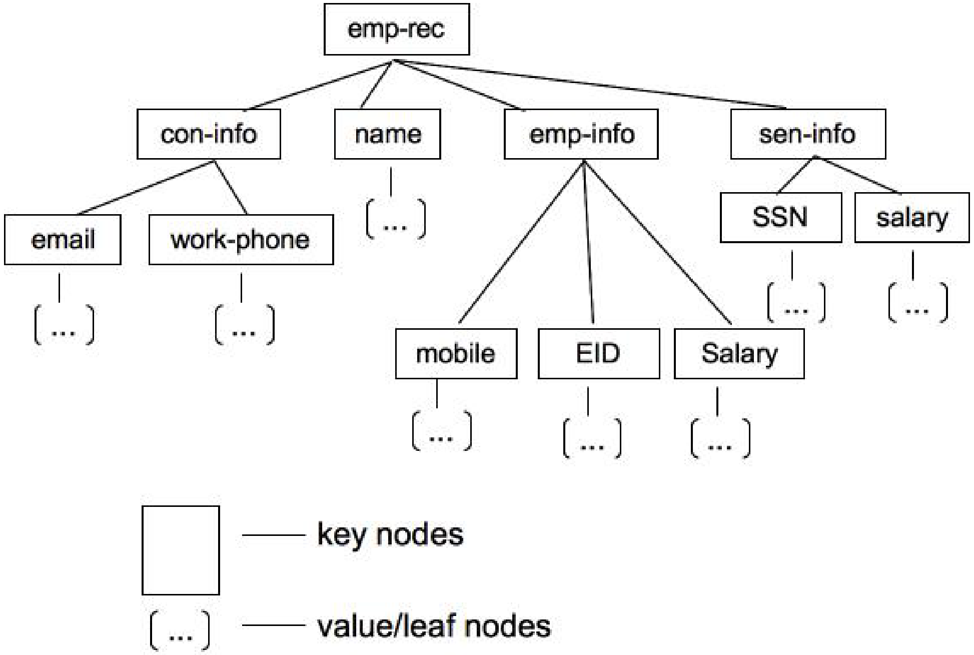
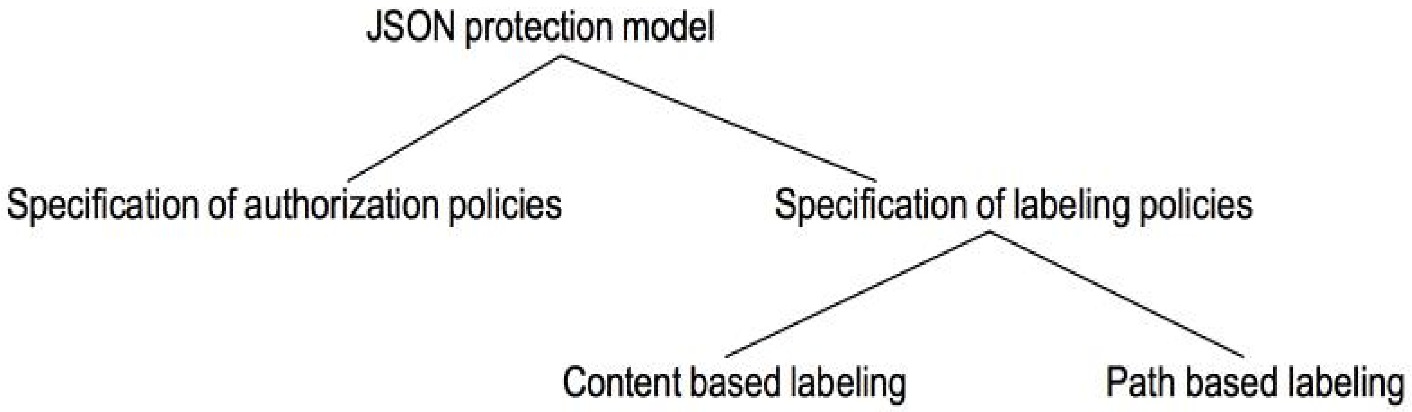
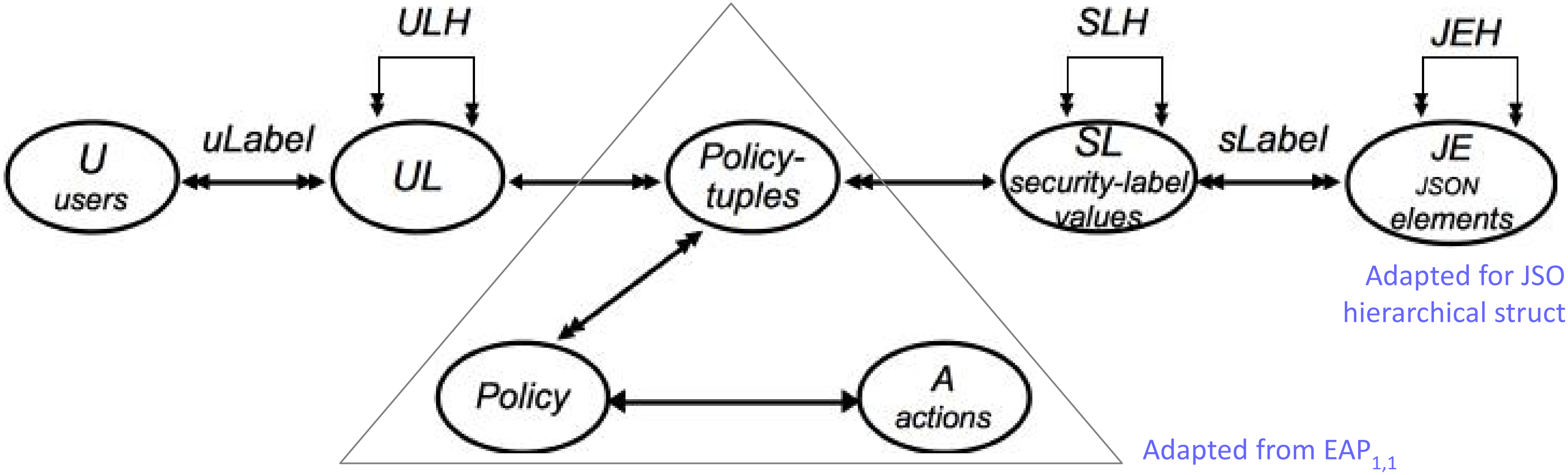


Figure 15: JSON data and JSON tree

JSON protection model (scope)


JSON protection model
Specification of authorization policies
Specification of labeling policies
Content based labeling
Path based labeling

JSON protection model (continuing)



Adapted for JSON hierarchical structure

Adapted from $EAP_{1,1}$

Figure 16: The Attribute-based Operational Model (AtOM)

World-Leading Research with Real-World Impact!

JSON protection model - examples

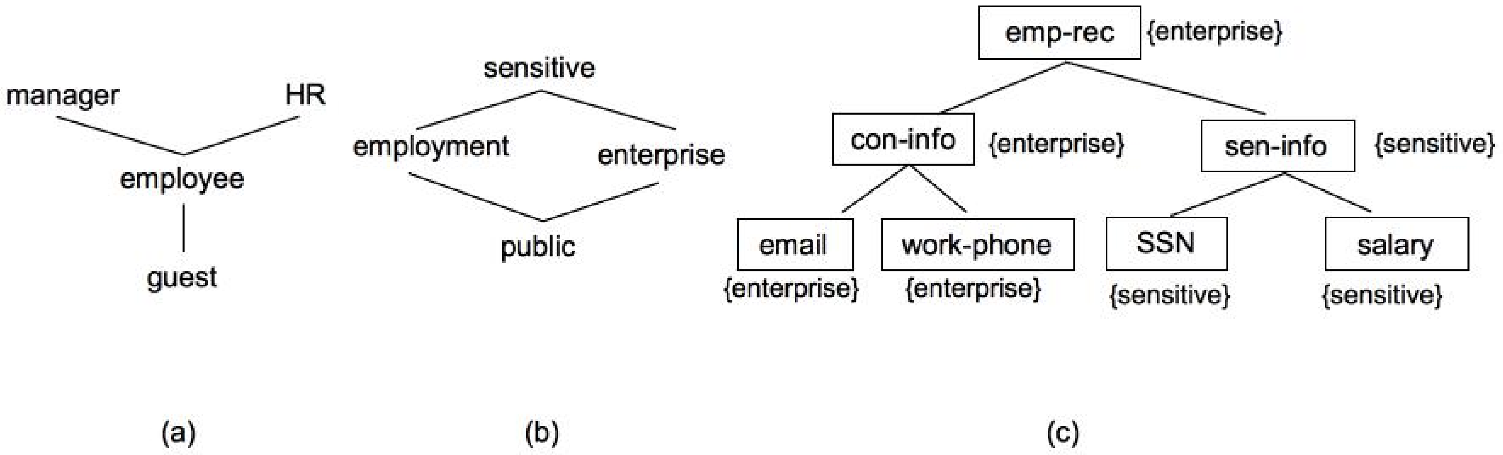
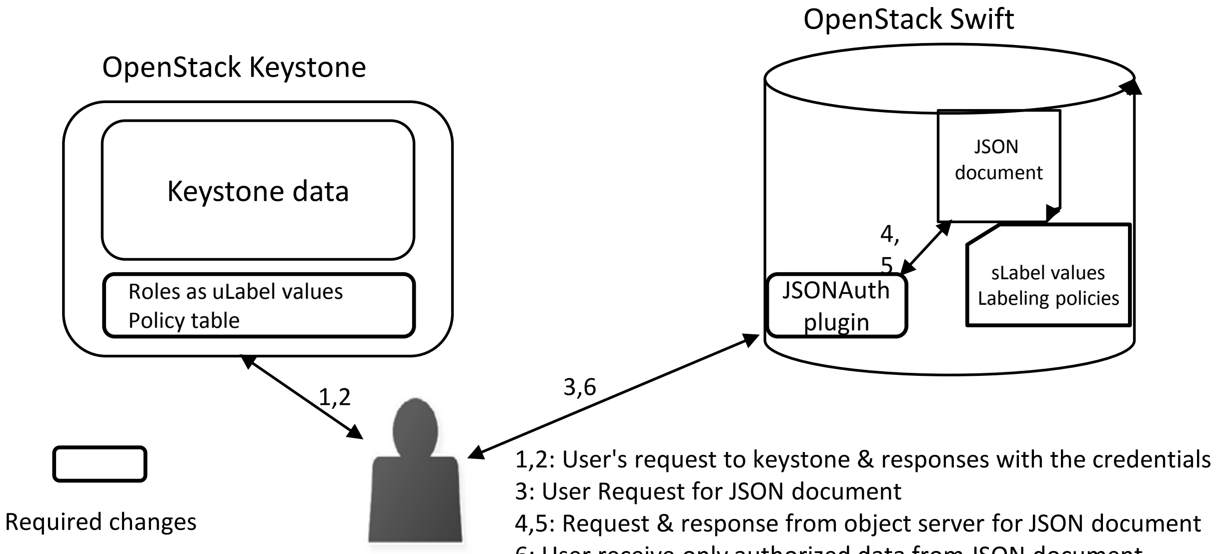


Figure 17: Examples of (a) User-label values, (b) security-label values and (c) annotated JSON tree

Example of a protection policy:

$\text{Policy}_{\text{read}} \equiv$ {(manager, sensitive), (employee, enterprise) }

Prototype implementation - (content-level access control for OpenStack Swift)



1,2: User's request to keystone & responses with the credentials
3: User Request for JSON document
4,5: Request & response from object server for JSON document
6: User receive only authorized data from JSON document

Figure 22: Implementation in OpenStack Swift Cloud

Implementation - evaluation

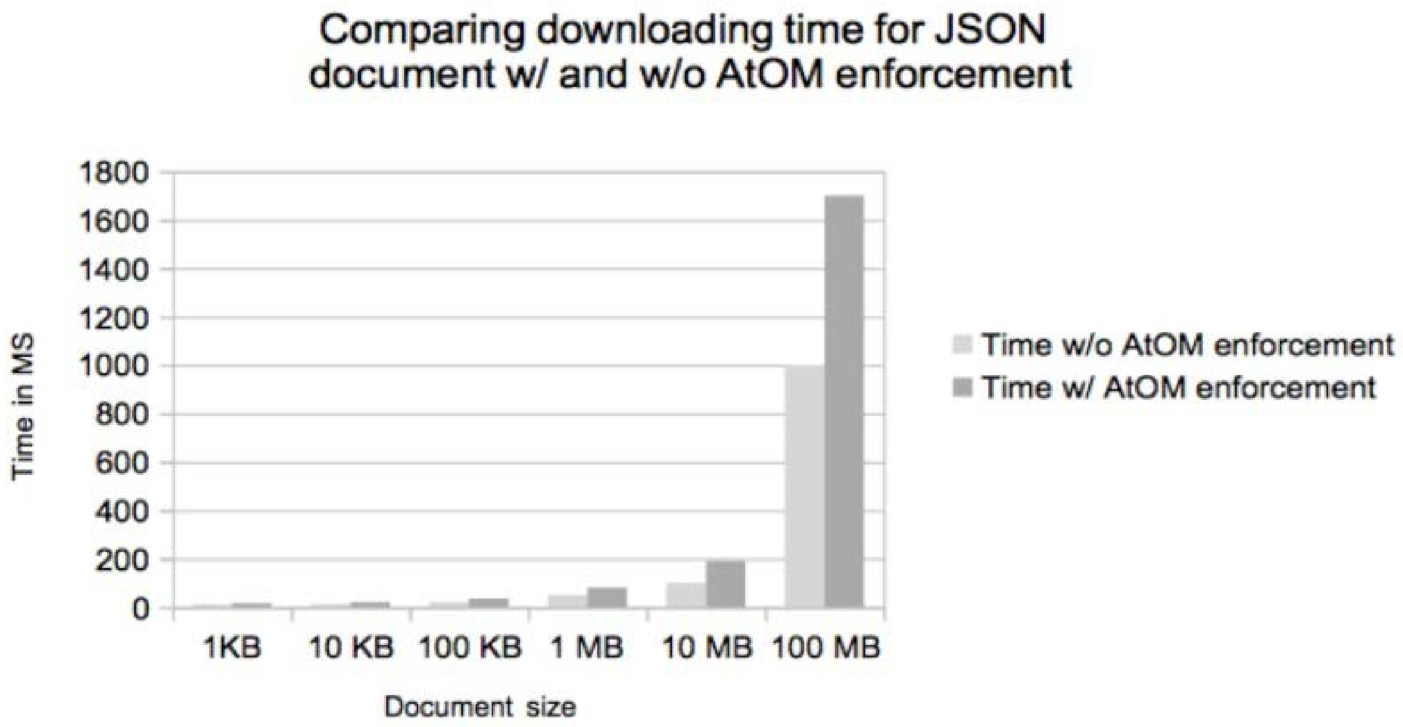
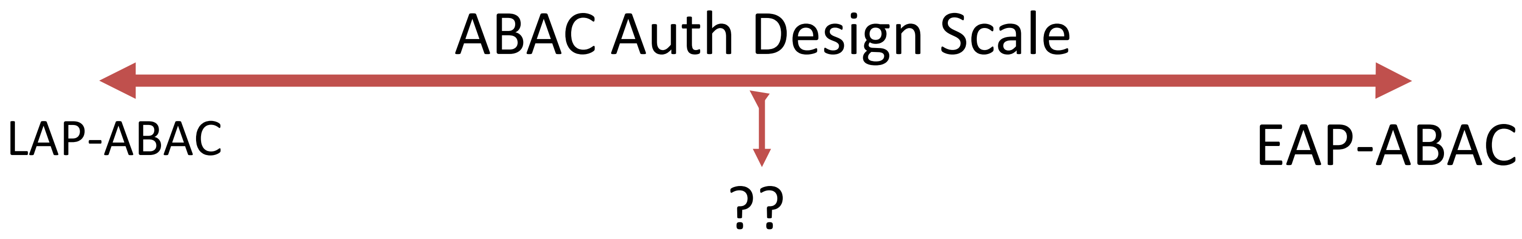


Fig 23: Performance evaluation

Future work and Conclusion

Future work

Optimal representation of authorization policy:



ABAC Auth Design Scale

LAP-ABAC

??

EAP-ABAC

Future work

Administration of

- enumerated authorization-policy
    - enumerated vs logical-formula authorization-policy

Conclusion

- Enumerated authorization-policy models
- Enumerated vs logical-formula authorization-policy models
- Enforcement

Publications

Included in the dissertation:

1. **Biswas, Prosunjit**, Ravi Sandhu, and Ram Krishnan. "Label-based access control: an ABAC model with enumerated authorization policy." Proceedings of the 2016 ACM International Workshop on Attribute Based Access Control. ACM, 2016. [No-of-pages: 12, Status: Full Paper]

2. **Biswas, Prosunjit**, Ravi Sandhu, and Ram Krishnan. "A comparison of logical-formula and enumerated authorization policy ABAC models." IFIP Annual Conference on Data and Applications Security and Privacy. Springer International Publishing, 2016. [No-of-pages: 8, Status: Short Paper]

3. **Biswas, Prosunjit**, Ravi Sandhu, and Ram Krishnan. "An Attribute-Based Protection Model for JSON Documents." International Conference on Network and System Security. Springer International Publishing, 2016. [No-of-pages: 15, Status: Full Paper]

4. **Biswas, Prosunjit**, Farhan Patwa, and Ravi Sandhu. "Content level access control for openstack swift storage." Proceedings of the 5th ACM Conference on Data and Application Security and Privacy. ACM, 2015. [No-of-pages: 4, Status: Poster]

Beyond dissertation:

5. **Biswas, Prosunjit**, Ravi Sandhu, and Ram Krishnan. "Uni-ARBAC: A Unified Administrative Model for Role-Based Access Control." International Conference on Information Security. Springer International Publishing, 2016. [No-of-pages: 14, Status: Full Paper]

6. **Biswas, Prosunjit**, Ravi Sandhu, and Ram Krishnan. "Attribute Transformation for Attribute-Based Access Control." Proceedings of the 2017 ACM International Workshop on Attribute Based Access Control. ACM, 2017. [No-of-pages: 8, Status: Full Paper]