A Multi-Tenant RBAC Model for Collaborative Cloud Services

Bo Tang, Qi Li and Ravi Sandhu

Presented by Bo Tang

at

The 11th International Conference on

Privacy, Security and Trust (PST)

July 12, 2013

Tarragona, Spain

OUTLINE

- Introduction and Background
- A Family of Multi-Tenant RBAC (MT-RBAC) Models
  - MT-RBAC$_{0,1,2}$
  - Administrative MT-RBAC (AMT-RBAC) model
  - Constraints
- Prototype Implementation and Evaluation
- Related Work
- Conclusion and Future Work

OUTLINE

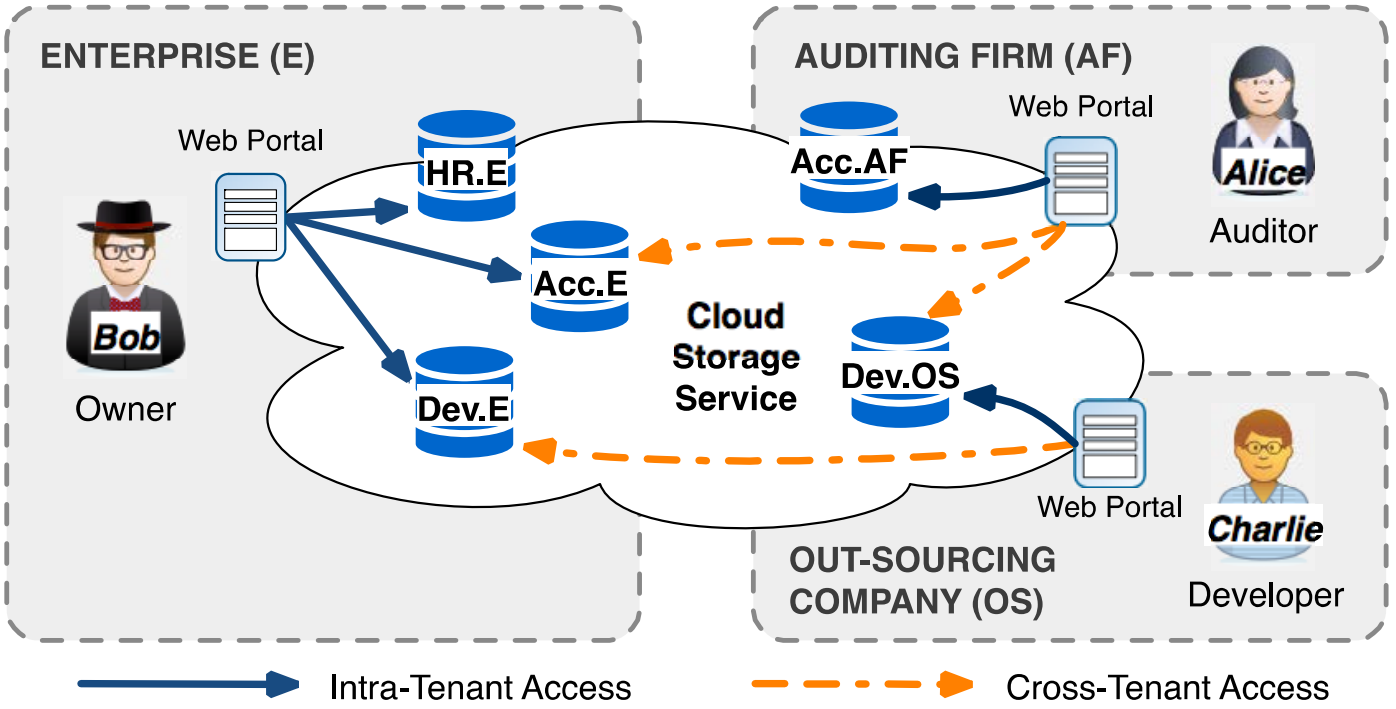
- Introduction
- A Family of Multi-Tenant RBAC (MT-RBAC) Models
  - $MT\text{-}RBAC_{0,1,2}$
  - Administrative MT-RBAC (AMT-RBAC) model
  - Constraints
- Prototype Implementation and Evaluation
- Related Work
- Conclusion and Future Work

Cloud Computing

- Shared infrastructure
  - [$$$] -----> [$|$|$]
- Multi-Tenancy
  - Virtually dedicated resources
- Drawbacks:
  - Data Locked-in
    - Collaborations can only be achieved through desktop.
    - E.g.: create/edit Word documents in Dropbox.
  - How to collaborate in the cloud?

Source: http://blog.box.com/2011/06/box-and-google-docs-accelerating-the-cloud-workforce/

Motivation



- C1. *Charlie* as a *developer* in OS has to access the source code stored in Dev.E to perform his out-sourcing job;
- C2. *Alice* as an *auditor* in AF requires read-only access to financial reports stored in Acc.E; and
- C3. *Alice* needs read-only accesses to Dev.E and Dev.OS in order to audit the out-sourcing project.

Industry Solutions

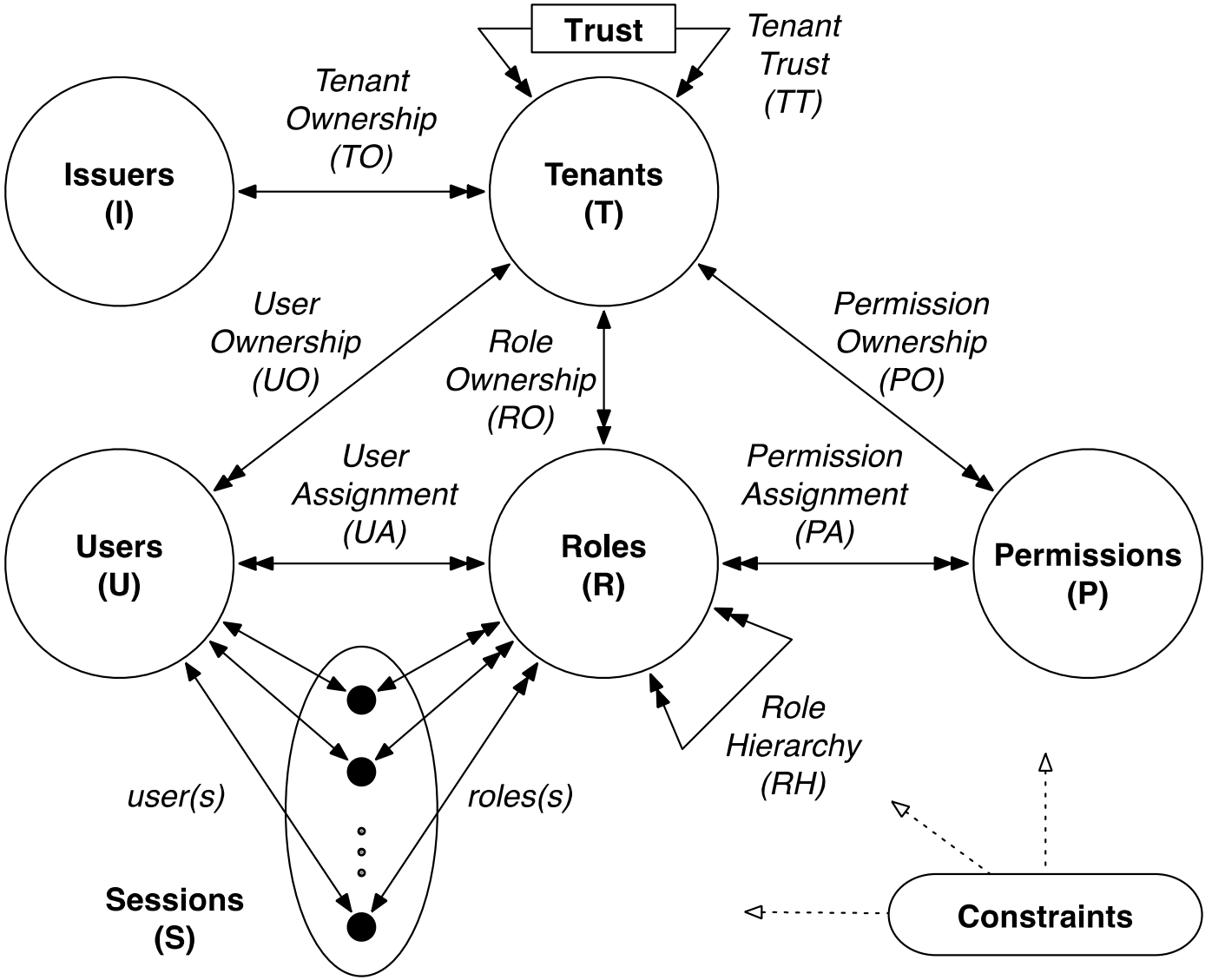
- Microsoft and IBM: Fine-grained data sharing in SaaS using DB schema
  - Only feasible in DB
- NASA: RBAC + OpenStack (Nebula)
  - Lacks ability to support multi-org collaborations
- Salesforce (Force.com): Single Sign-On + SAML
  - Focus on authentication and simple authorization
  - Heavy management of certificates

Source: http://msdn.microsoft.com/en-us/library/aa479086.aspx
http://nebula.nasa.gov/blog/2010/06/03/nebulas-implementation-role-based-access-control-rbac/
http://wiki.developerforce.com/page/Single_Sign-On_with_SAML_on_Force.com

OUTLINE

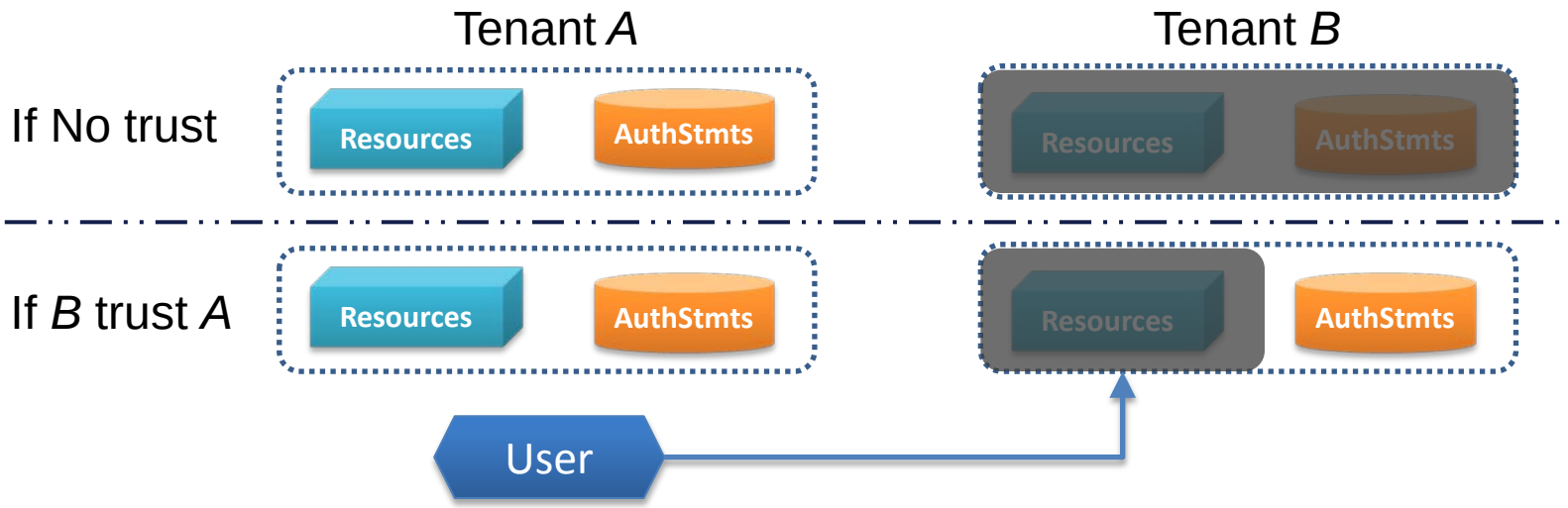
- Introduction
- A Family of Multi-Tenant RBAC (MT-RBAC) Models
  - MT-RBAC$_{0,1,2}$
  - Administrative MT-RBAC (AMT-RBAC) model
  - Constraints
- Prototype Implementation and Evaluation
- Related Work
- Conclusion and Future Work

MT-RBAC


Trust
Tenant Trust (TT)
Tenant Ownership (TO)
Issuers (I)
Tenants (T)
User Ownership (UO)
Role Ownership (RO)
Permission Ownership (PO)
User Assignment (UA)
Permission Assignment (PA)
Users (U)
Roles (R)
Permissions (P)
Role Hierarchy (RH)
user(s)
roles(s)
Sessions (S)
Constraints

Trust Model

- If B (resource owner) trusts A then A can assign
  - B's permissions to A's roles; and
  - B's roles as junior roles to A's roles.
- CanUse(r_B) = {*A*, *B*, …}



MT-RBAC$_{0,1,2}$

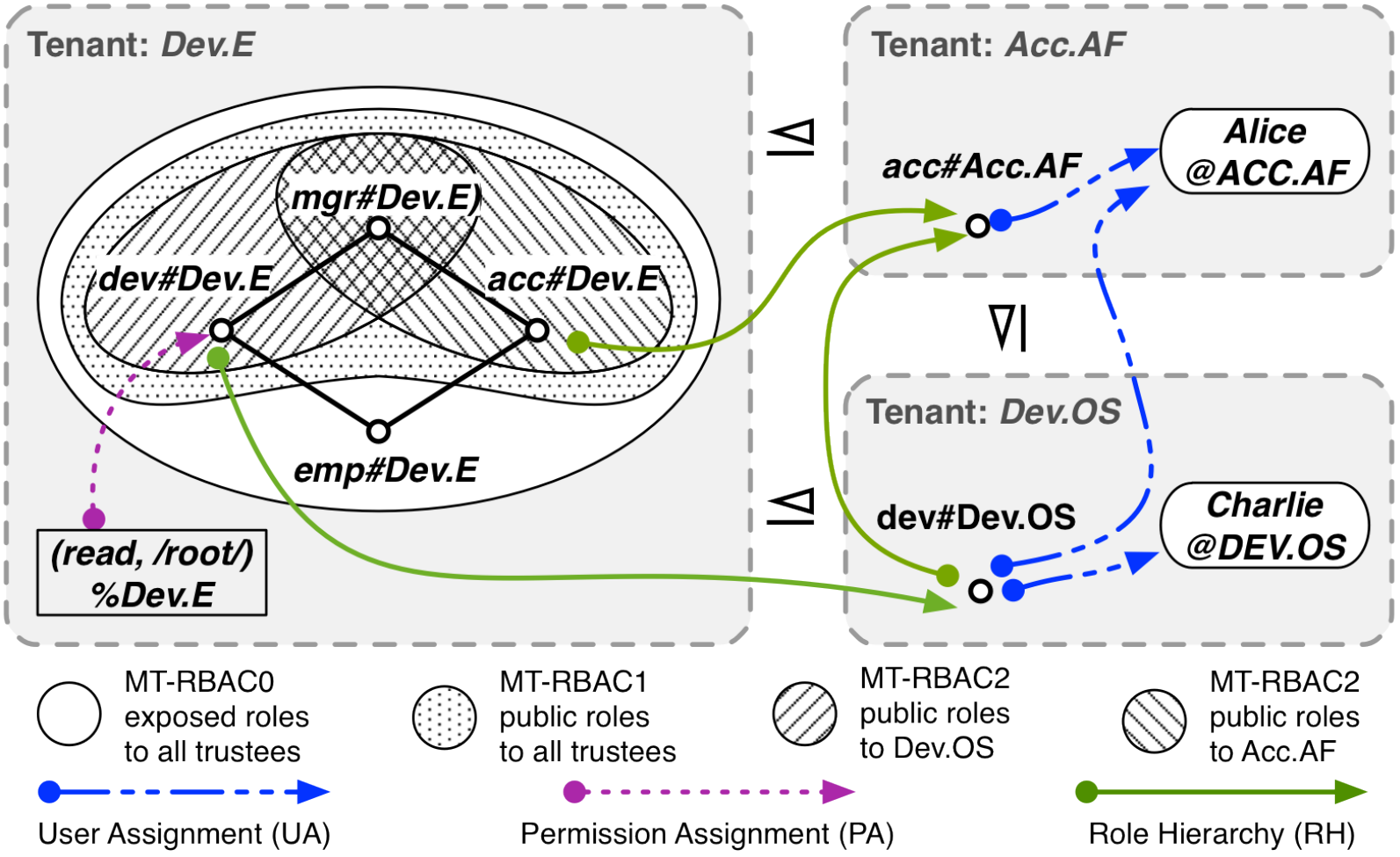

Tenant: Dev.E
mgr#Dev.E)
dev#Dev.E
acc#Dev.E
emp#Dev.E
(read, /root/)
%Dev.E
Tenant: Acc.AF
acc#Acc.AF
Alice
@ACC.AF
Tenant: Dev.OS
dev#Dev.OS
Charlie
@DEV.OS
MT-RBAC0
exposed roles
to all trustees
MT-RBAC1
public roles
to all trustees
MT-RBAC2
public roles
to Dev.OS
MT-RBAC2
public roles
to Acc.AF
User Assignment (UA)
Permission Assignment (PA)
Role Hierarchy (RH)

Table 3.2: Administration functions available to issuer i in AMT-RBAC

Function	Precondition	Update
$assignUser$ (t, r, u)	$(t,i) \in TO \wedge (u,t) \in UO \wedge t \in canUse(r)$	$UA' = UA \cup \{(u,r)\}$
$revokeUser$ (t, r, u)	$(t,i) \in TO \wedge (u,t) \in UO \wedge t \in canUse(r) \wedge (u,r) \in UA$	$UA' = UA \setminus \{(u,r)\}$
$assignPerm$ (t, r, p)	$(t,i) \in TO \wedge (r,t) \in RO \wedge (p,t) \in PO$	$PA' = PA \cup \{(p,r)\}$
$revokePerm$ (t, r, p)	$(t,i) \in TO \wedge (r,t) \in RO \wedge (p,t) \in PO \wedge (p,r) \in PA$	$PA' = PA \setminus \{(p,r)\}$
$assignRH$ (t, r_{asc}, r_{desc})	$(t,i) \in TO \wedge (r_{asc},t) \in RO \wedge t \in canUse(r_{desc}) \wedge \neg(r_{asc} \gg r_{desc})$ † $\wedge \neg(r_{desc} \geq r_{asc})$ ‡	$\geq' = \geq \cup \{r, q : R \mid r \geq r_{asc} \wedge r_{desc} \geq q \wedge roleOwner(r) \in canUse(q) \bullet (r,q)\}$
$revokeRH$ (t, r_{asc}, r_{desc})	$(t,i) \in TO \wedge (r_{asc},t) \in RO \wedge t \in canUse(r_{desc}) \wedge r_{asc} \gg r_{desc}$	$\geq' = (\gg \setminus \{(r_{asc}, r_{desc})\})^*$ §

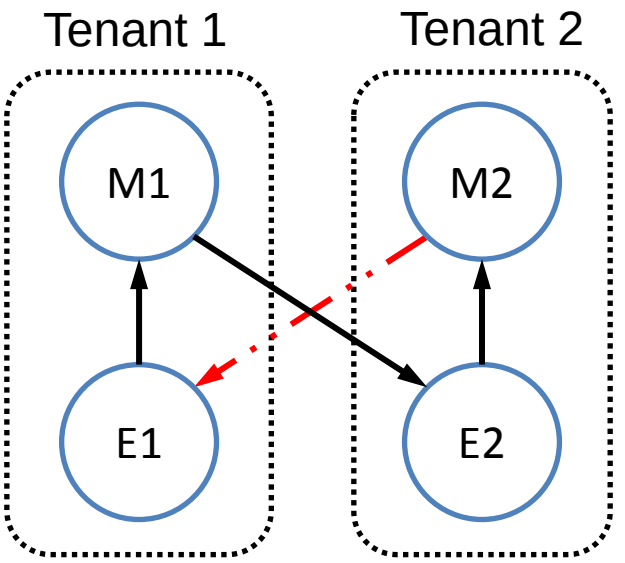
AMT-RBAC (Contd.)

$assignTrust(t, t_1)$	$t_1 \in T$	$\trianglelefteq' = \trianglelefteq \cup \{(t, t_1)\}$
$revokeTrust(t, t_1)$	$t_1 \in T \wedge t \neq t_1 \wedge t \trianglelefteq t_1$	$\trianglelefteq' = \trianglelefteq \setminus \{(t, t_1)\}$ ¶
$addTenant(t)$	$i \in I \wedge t \notin T$	$T' = T \cup \{t\}$
$deleteTenant(t)$	$(t, i) \in TO \wedge t \in T$	$[\forall t_1 : T \Rightarrow revokeTrust(t, t_1)]$ $[\forall t_2 : T \Rightarrow revokeTrust(t_2, t)]$ $UA' = UA \setminus \{(u, r) \mid (u, t) \in UO \wedge (r, t) \in RO\}$ $PA' = PA \setminus \{(p, r) \mid (p, t) \in PO \wedge (r, t) \in RO\}$ $RH' = RH \setminus \{(r, r') \mid (r, t) \in RO \wedge (r', t) \in RO\}$ $U' = U \setminus \{u \mid (u, t) \in UO\}$ $UO' = UO \setminus \{(u, t) \mid u \notin U\}$ $R' = R \setminus \{r \mid (r, t) \in RO\}$ $RO' = RO \setminus \{(r, t) \mid r \notin R\}$ $P' = P \setminus \{p \mid (p, t) \in PO\}$ $PO' = PO \setminus \{(p, t) \mid p \notin P\}$ $T' = T \setminus \{t\}$ $TO' = TO \setminus \{(t, i)\}$

† The notation "$\gg$" represents an immediate inheritance relation.
‡ This condition avoids the creation of role cycles.
§ The notation "$*$" represents recursive updates for the entire RH assignments. Implied RH relations are preserved after revocation.
¶ The revocation of a trust relation automatically triggers updates in the $canUse()$ function of all t's roles and then corresponding UA and RH accordingly.

Constraints

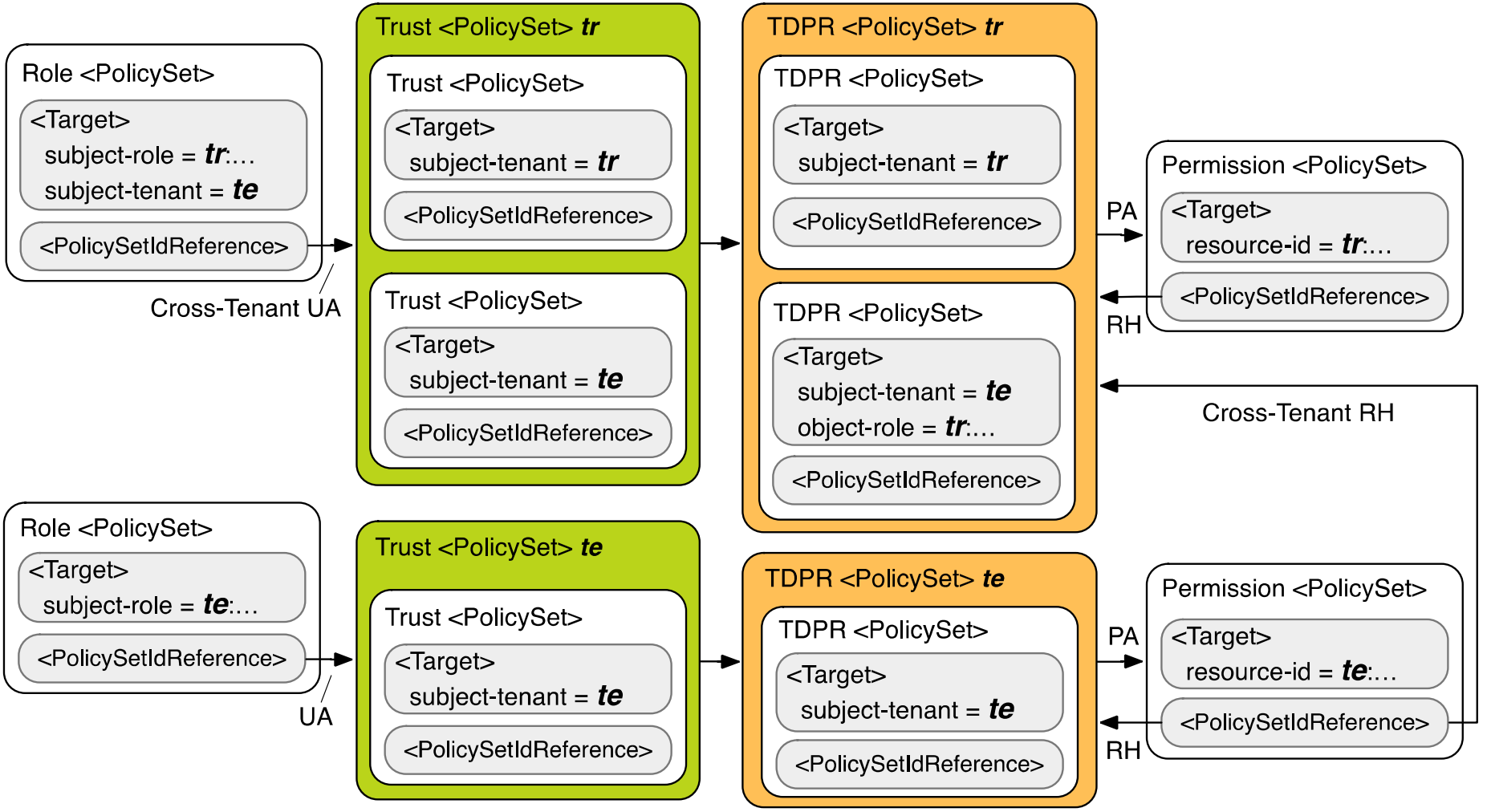
- Cyclic Role Hierarchy: lead to implicit role upgrades in the role hierarchy
- SoD: conflict of duties
  - Tenant-level
    - E.g.: SOX compliance companies may not hire the same company for both consulting and auditing.
  - Role-level
    - across tenants
- Chinese Wall: conflict of interests among tenants
  - E.g.: do not share infrastructure with competitors.



OUTLINE

- Introduction
- A Family of Multi-Tenant RBAC (MT-RBAC) Models
  - MT-RBAC$_{0,1,2}$
  - Administrative MT-RBAC (AMT-RBAC) model
  - Constraints
- Prototype Implementation and Evaluation
- Related Work
- Conclusion and Future Work

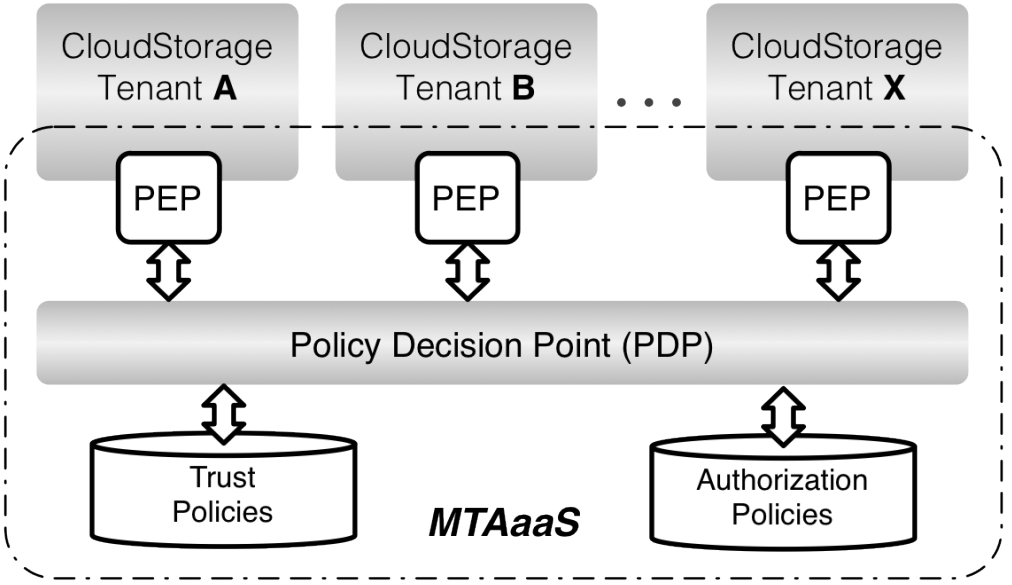
Policy Specification of MT-RBAC$_2$



user = "Charlie"; permission = "(read, /root)%Dev.E"
tr = "Dev.E"; te = "Dev.OS"

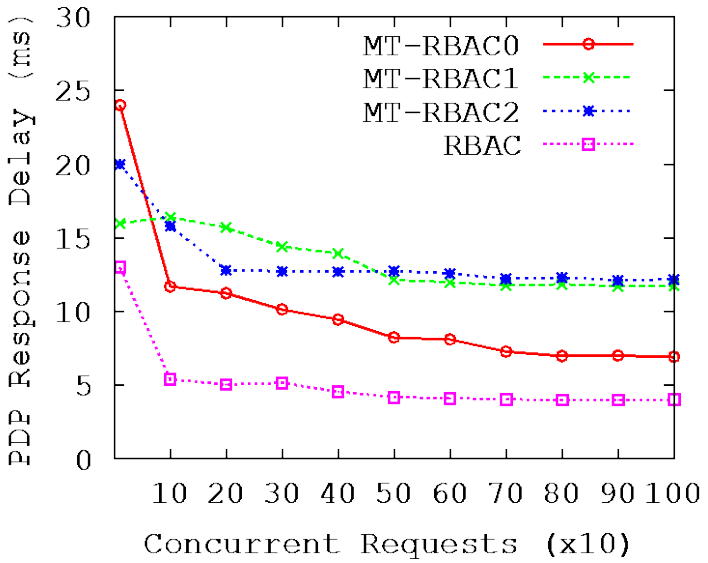
MTAaaS Platform Prototype

- Experiment Settings
  - CloudStorage: an open source web based cloud storage and sharing system.
  - Joyent, FlexCloud
- Authorization Service
  - Centralized PDP
  - Distributed PEP

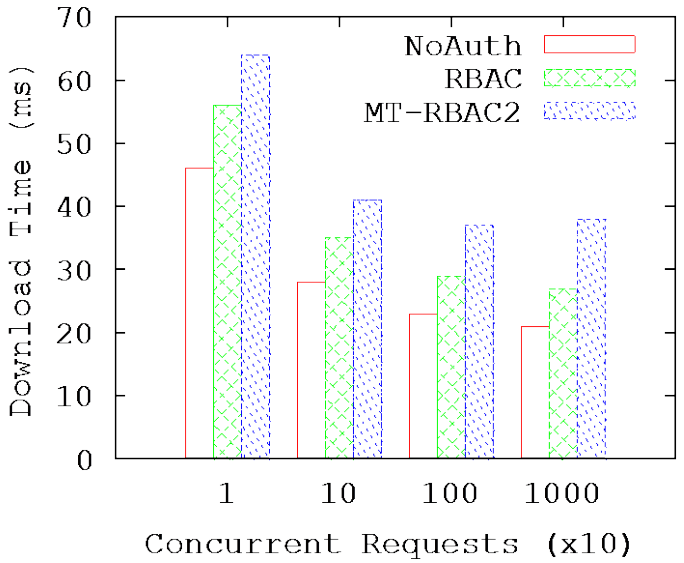


Evaluation: Performance

- MT-RBAC vs RBAC
  - More policy references incur more decision time
- MT-RBAC$_2$ introduces 16 ms overhead on average.



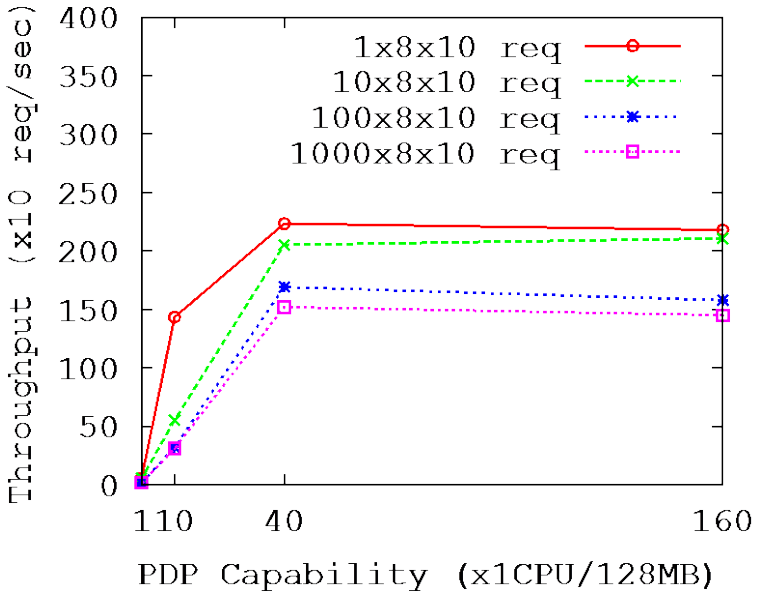
PDP Performance



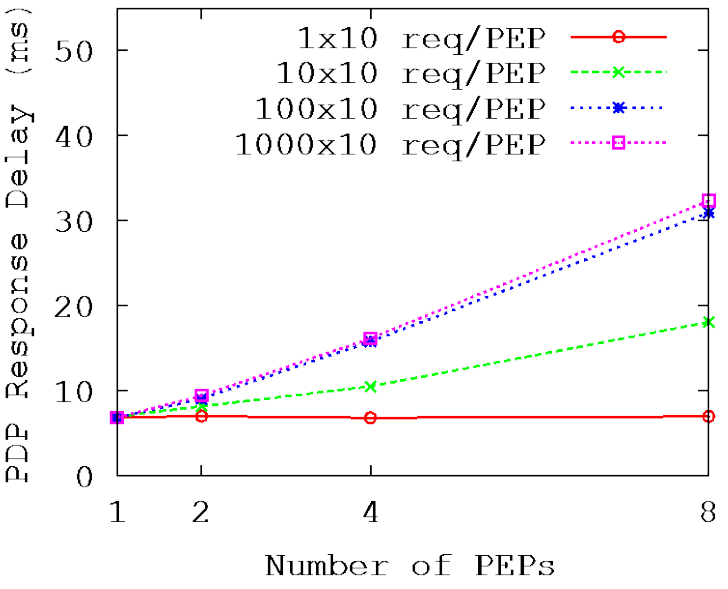
Client-End Performance

Evaluation: Scalability

- Scalable by changing either
  - PDP capability; or
  - Number of PEPs.



PDP Scalability



PEP Scalability

OUTLINE

- Introduction
- A Family of Multi-Tenant RBAC (MT-RBAC) Models
  - MT-RBAC$_{0,1,2}$
  - Administrative MT-RBAC (AMT-RBAC) model
  - Constraints
- Prototype Implementation and Evaluation
- Related Work
- Conclusion and Future Work

Characteristics of Cloud

- Agility
  - Collaboration and collaborators are temporary
- Centralized Facility
  - No need to use cryptographic certificates
- Homogeneity
  - Same access control model in each tenant
- Out-Sourcing Trust
  - Collaboration spirit

Literature in Multi-org/dom

- RBAC
  - CBAC, GB-RBAC, ROBAC (e.g.: player transfer in NBA)
  - Require central authority managing collaborations
- Delegation Models
  - dRBAC and PBDM (e.g.: allowing subleasing)
  - Lacks agility (which the cloud requires)
- Grids
  - CAS, VOMS, PERMIS
  - Absence of centralized facility and homogeneous architecture (which the cloud has)

Literature (Contd.)

- Role-based Trust
  - RT, Traust, RMTN AND RAMARS_TM
  - Calero et al: towards a multi-tenant authorization system for cloud services
    - Implementation level PoC
    - Coarse-grained trust model
  - MTAS
  - Suits the cloud (out-sourcing trust)

Role-Based Trust Model Comp.

Table 3.3: Trust Model Comparison. A and B represent two entities, issuers and tenants respectively in RT, MTAS and MT-RBAC. A represents the resource owner and B the requester.

	RT	**MTAS**	**MT-RBAC**
trust relation required	A trust B	B trust A	A trust B
trust assigner	A	B	A
authorization assigner	A	A	B
User Assignment (UA)	$U \rightarrow A.R$	$U \rightarrow A.R$	$B.U \rightarrow B.R \cup A.R$
Permission Assignment (PA)	$A.P \rightarrow A.R$	$A.P \rightarrow A.R \cup B.R$	$B.P \rightarrow B.R$
Role Hierarchy (RH)	$A.R \leq B.R$	$A.R \leq B.R$	$A.R \leq B.R$
require common vocabulary	Yes	No	No
require centralized facility	No	Yes	Yes

OUTLINE

- Introduction
- A Family of Multi-Tenant RBAC (MT-RBAC) Models
  - MT-RBAC$_{0,1,2}$
  - Administrative MT-RBAC (AMT-RBAC) model
  - Constraints
- Prototype Implementation and Evaluation
- Related Work
- Conclusion and Future Work

Conclusion

- Collaboration needs among cloud services
- MT-RBAC model family
  - Formalization
  - Administration
  - Constraints
- MTAaaS architecture viable in the cloud
- Overhead ≈ 16ms and scalable in the cloud
- Comparison of role-based trust models

Future Work

- Cross-tenant trust models in cloud computing
- Other multi-tenant access control models
  - MT-ABAC
  - MT-RT
  - MT-PBAC and more.
- Implementation MT-RBAC in OpenStack API.


Q & A

Thank You!

Multi-Tenant Authorization as a Service (MTAaaS)