Expressive Power, Safety and Cloud Implementation of Attribute and Relationship Based Access Control Models

Dissertation Defense: Tahmina Ahmed

Dissertation Committee:
Dr. Ravi Sandhu, Supervising Professor
Dr. Jianwei Niu
Dr. Gregory White
Dr. Weining Zhang
Dr. Ram Krishnan
Outline

- Introduction
- Comparison of ReBAC and ABAC
- Object-to-Object Relationship Based Access Control: Model and Multicloud demonstration
- Safety and Expressive Power Comparison of $\text{ABAC}_\alpha$ and its Enhancements
- Conclusion
Introduction

- Comparison of ReBAC and ABAC
- Object-to-Object Relationship Based Access Control: Model and Multicloud demonstration
- Safety and Expressive Power Comparison of $ABAC_\alpha$ and its Enhancements
- Conclusion
Access Control Evolution

Discretionary Access Control (DAC), 1970

Mandatory Access Control (MAC), 1970

Role Based Access Control (RBAC), 1995

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

Born 1990s

Relationship Based Access Control (ReBAC) ????

Born mid 2000s

Figure 1: Evolution of Access Control
ABAC: Using Attributes for controlling access

X.500 standard (1994): Manages object information through attributes

Using attributes for controlling usage of digital resources (Park and Sandhu 2004)
ReBAC: Using Relations for Controlling Access

IoT Application is a Graph

A sample social graph

A sample Provenance Graph (Park et al. 2012)
Problem Statement

Are they Comparable? Can Attributes Express Relationships?
Can ReBAC Configure ABAC? Vice versa?
Do they have equal expressive power? If not, which one is more expressive?

**ABAC vs. ReBAC**: There is a fundamental lack of understanding regarding the relationship between ABAC and ReBAC.

What are the novel ways other than OSN ReBAC can be seen, extended and applied?

**ReBAC Potential**: The potential of ReBAC has recently been recognized and there remain many directions in which ReBAC models can be developed.
Problem Statement (Cont..)

- Which one is a standard ABAC model: UCON? $ABAC_\alpha$? $ABAC_\beta$? NIST ABAC?
- What are the core characteristics of an ABAC model?
- What is the safety property and expressive power variance among the existing ABAC models?

**ABAC vs. ABAC:** There is a proliferation of ABAC models without a formal understanding of their safety properties and relative expressive power.
Summary of Contribution

- A Comparison of ReBAC and ABAC.
- A novel ReBAC model definition and its application in the cloud.
- Safety and Expressive Power analysis of $ABAC_\alpha$ and its extensions.
Outline

- Introduction
- Comparison of ReBAC and ABAC
- Object-to-Object Relationship Based Access Control: Model and Multicloud demonstration
- Safety and Expressive Power Comparison of $ABAC_\alpha$ and its Enhancements
- Conclusion
Attribute Types

1. Attribute Value Structure
   - Atomic-valued or Single-valued Attribute (e.g. gender)
   - Set-valued or Multi-valued Attribute (e.g. phoneNumber)
   - Structured Attribute (e.g. person-Info (name, age, phoneNumber))

2. Attribute Value Scope
   - Entity Attribute (e.g. friend)
   - Non-entity Attribute (e.g. age)

3. Boundedness of attribute range
   - Finite Domain Attribute (e.g. gender)
   - Infinite Domain Attribute (e.g. time)

4. Attribute association
   - Contextual or Environmental Attribute (e.g. currentTime)
   - Meta Attribute (e.g. role(user) = manager, task(manager) = supervise)

5. Attribute mutability
   - Mutable Attribute
   - Immutable Attribute
Expressing Multilevel Relationship With Attributes

<table>
<thead>
<tr>
<th>Attribute Composition</th>
<th>Composite Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Needs one attribute: friend</td>
<td>Needs two attributes</td>
</tr>
<tr>
<td>Policy Expression uses Attribute composition</td>
<td>1. friend</td>
</tr>
<tr>
<td>friend(Alice)={Bob}</td>
<td>2. friendOfFriend</td>
</tr>
<tr>
<td>friend(friend(Alice))={Carol}</td>
<td>Policy Expression uses direct attributes</td>
</tr>
<tr>
<td>friend(friend(Alice))={Carol}</td>
<td>friend(Alice) ={Bob}</td>
</tr>
</tbody>
</table>

friendOfFriend(Alice)={Carol}
Figure 2: ReBAC Classification
Figure 3: ABAC Framework
Expressing Relationship Graph with Attributes

- **Entity types = \{user, project, folder, document\}
- **Attributes:
  - User attributes = \{Participant-of, Supervises\}
  - Folder attributes = \{Resource-for, FolderMember-of\}
  - Project attributes = \{
  - Document attributes = \{DocMember-of\}

Figure 4: Relationship Graph
[Crampton et al. 2014] Expressible with ReBAC$_B$ and ABAC$_E$

Figure 5: Relationship Graph Expressible with ReBAC$_{BN}$ and ABAC$_E$

- **entityType = \{user\}
- **Attribute:
  - User’s entity attribute = \{friend\}
  - User’s Non Entity Attribute = \{Name, Age, Gender\}
Expressing Relationship Graph with Attributes (Continued...)

- entityType = \{user, project, tenant\}
- Attribute:
  - user’s atomic entity attribute = \{supervises\}
  - User’s structured entity Attribute = \{assignedBy\}
    e.g. assignedBy(Bob) = ("Project1", “supervises”, “Alice”)

- Entity types: \{user, tenant, role\}
- Attribute:
  - User’s atomic entity attribute: \{UO, UA\}
  - Users Structured Entity Attribute: \{dependentEdge\}
    dependentEdge(u) = (”r”, “UA”, \{(y,x,TT)\} )

Figure 6: Relationship Graph Expressible with ReBAC_{BE} and ABAC_{ES}

Figure 7: Relationship Graph [cheng et al 2016] Expressible with ReBAC_{BNES} and ABAC_{ES}
Comparison: On Dynamics

$ABAC_X \equiv ReBAC_Y$ Means

- Static and finite attribute domain
  $ABAC_X \equiv Static \ ReBAC_Y$
- $ABAC_X$ Attribute value changes with finite domain
  $\equiv Relationship \ Dynamic \ ReBAC_Y$
- $ABAC_X$ with entity changes and infinite domain entity attribute
  $\equiv node \ dynamic \ ReBAC_Y$

Figure 8: ReBAC Dynamics, ABAC Dynamics and Attribute Domain wise Comparison between ReBAC and ABAC
Comparison: Equivalent Structural Models for ReBAC and ABAC

Figure 9: Equivalence of ReBAC and ABAC Structural Classification
Comparison: Non-Equivalent Structural models for ReBAC and ABAC

Figure 10: Non-Equivalence of ReBAC and ABAC Structural Classification
Comparison

- Attribute Composition: Polynomial complexity for authorization policy and constant complexity on update
- Composite attribute: Constant complexity on authorization policy and polynomial complexity on update to maintain relationship changes.
- Performance Depends on:
  - Node Dynamics
  - Relationship Dynamics
  - Density of the Relationship Graph

Choice of Models:
- For static system or only non entity attribute change—— Composite attribute is the best approach
- System with huge node dynamics, relationship dynamics and high relationship density—— Attribute composition is the best option
- If the system is in the middle between two extremes ---- A hybrid approach where both composite attribute and attribute composition is used.
- Hybrid Approach:
  To achieve p level relationship composition it uses m level composite attribute and n level attribute composition where p = n X m.
Introduction

Comparison of ReBAC and ABAC

Object-to-Object Relationship Based Access Control: Model and Multicloud demonstration

Safety and Expressive Power Comparison of $ABAC_\alpha$ and its Enhancements

Conclusion
User to user relationships in a sample social graph [UURAC, Cheng et al. 2012]

User to user, user to resource and resource to resource relationships in a sample social graph [UURAC, Cheng et al. 2012]

Limitations:

Cannot configure relationship between objects independent of user.
Cannot express authorization policy solely considering object relationship.
An Object to Object Relationship Based Access Control

**Policy Level Example**

- $\text{ACL}(o_1) = \{u_1\}$
- $\text{ACL}(o_2) = \{\}$
- $\text{ACL}(o_3) = \{u_2\}$

- $\text{policyLevel}(a_1, o_1) = 2$
- $\text{policyLevel}(a_2, o_1) = 0$
- $\text{policyLevel}(a_1, o_2) = 1$
- $\text{policyLevel}(a_2, o_2) = 0$
- $\text{policyLevel}(a_1, o_3) = 3$
- $\text{policyLevel}(a_2, o_3) = 2$
- $\text{policyLevel}(a_1, o_4) = 2$
- $\text{policyLevel}(a_2, o_4) = 0$
**OOReBAC: Model Components and Definition**

- U is a set of users
- O is a set of objects
- \( R \subseteq \{z \mid z \subseteq O \land |z| = 2\} \)
- \( G=\langle O, R \rangle \) is an undirected relationship graph with vertices O and edges R
- A is a set of actions
- \( P^i(o_1) = \{ o_2 \mid \text{there exists a simple path of length } p \text{ in graph } G \text{ from } o_1 \text{ to } o_2 \} \)
- policyLevel: \( O \times A \rightarrow \mathbb{N} \)
- ACL: \( O \rightarrow 2^U \) which returns the Access control List of a particular object.
- There is a single policy configuration point. Authorization Policy. for each action \( a \in A \), \( \text{Authz}_a(u:U, o:O) \) is a boolean function which returns true or false and \( u \) and \( o \) are formal parameters.
- Authorization Policy Language: Each action “a” has a single authorization policy \( \text{Authz}_a(u:U, o:O) \) specified using the following language.
  \[ \phi := u \in \text{PATH}_i \]
  \[ \text{PATH}_i := \text{ACL}(P^0(o)) \cup \ldots \cup \text{ACL}(P^i(o)) \text{ where } i = \min(|O| - 1, \text{policyLevel}(a,o)) \]
  where for any set \( X \), \( \text{ACL}(X) = \bigcup_{x \in X} \text{ACL}(x) \)

---

**Figure 10: OOReBAC Model Components**
OOReBAC: An Example

Configuration:
- \( A = \{ \text{read, write} \} \)
- \( \text{Authz}_{\text{read}}(u:U,o:O) \equiv u \in P_{\text{policyLevel}(\text{read},o)} \)
- \( \text{Authz}_{\text{write}}(u:U,o:O) \equiv u \in P_{\text{policyLevel}(\text{write},o)} \)

Sequence of operations and its outcome:

- \( U = \{ u_1, u_2, u_3 \} \)
- \( O = \{ o_1, o_2, o_3, o_4 \} \)
- \( R = \{ (o_1, o_2), (o_2, o_3), (o_3, o_4) \} \)
- \( \text{ACL}(o_1) = \{ u_1 \} \)
- \( \text{ACL}(o_2) = \{ u_3 \} \)
- \( \text{ACL}(o_3) = \{ u_2 \} \)
- \( \text{ACL}(o_4) = \{ u_3 \} \)
- \( \text{policyLevel}(\text{read}, o_1) = 2 \)
- \( \text{policyLevel}(\text{write}, o_1) = 0 \)
- \( \text{policyLevel}(\text{read}, o_2) = 2 \)
- \( \text{policyLevel}(\text{write}, o_2) = 1 \)
- \( \text{policyLevel}(\text{read}, o_3) = 0 \)
- \( \text{policyLevel}(\text{write}, o_3) = 0 \)
- \( \text{policyLevel}(\text{read}, o_4) = 2 \)
- \( \text{policyLevel}(\text{write}, o_4) = 1 \)

- \( \text{read}(u_1, o_3), \text{write}(u_1, o_3) \) are denied
- \( \text{read}(u_2, o_1) \) is allowed, \( \text{write}(u_2, o_1) \) is denied
- \( \text{read}(u_1, o_4), \text{write}(u_1, o_4) \) are denied
An OOReBAC Instantiation

- \( U = \{ u_{pp}, u_{go}, u_{od}, u_{cp}, u_{cd}, u_{np} \} \)
- \( O = \{ mr_{pp}, mr_{go}, mr_{od}, mr_{cp}, mr_{cd}, mr_{np} \} \)
- \( R = \{ \{ mr_{pp}, mr_{go} \}, \{ mr_{go}, mr_{od} \}, \{ mr_{od}, mr_{cp} \}, \{ mr_{cp}, mr_{cd} \}, \{ mr_{cd}, mr_{np} \} \} \)

- \( ACL(mr_{pp}) = \{ u_{pp} \} \)
- \( ACL(mr_{go}) = \{ u_{go} \} \)
- \( ACL(mr_{od}) = \{ u_{od} \} \)
- \( ACL(mr_{cp}) = \{ u_{cp} \} \)
- \( ACL(mr_{cd}) = \{ u_{cd} \} \)
- \( ACL(mr_{np}) = \{ u_{np} \} \)

- Action \( = \{ \text{read, write} \} \)

- \( \text{policyLevel(read, mr}_{pp}) = \infty, \text{policyLevel(write, mr}_{pp}) = 0, \text{policyLevel(read, mr}_{go}) = \infty, \text{policyLevel(write, mr}_{go}) = 0, \text{policyLevel(read, mr}_{od}) = \infty, \text{policyLevel(write, mr}_{od}) = 0, \text{policyLevel(read, mr}_{cp}) = \infty, \text{policyLevel(write, mr}_{cp}) = 0, \text{policyLevel(read, mr}_{cd}) = \infty, \text{policyLevel(write, mr}_{cd}) = 0, \text{policyLevel(read, mr}_{np}) = \infty, \text{policyLevel(write, mr}_{np}) = 0 \)

- Authorization policy:
  - \( \text{Auth}_{\text{read}}(u, o) \equiv u \in P_{\text{policyLevel(read, o)}} \)
  - \( \text{Auth}_{\text{write}}(u, o) \equiv u \in P_{\text{policyLevel(write, o)}} \)

Sequence of Operations and Outcomes

1. read\((u_{pp}, mr_{pp})\) : authorized
2. read\((u_{cd}, mr_{pp})\) : authorized
3. write\((u_{pp}, mr_{cp})\) : authorized
4. write\((u_{cp}, mr_{pp})\) : denied
5. write\((u_{cd}, mr_{np})\) : denied

Figure 11: An Example of OOReBAC Application in Medical
Implementation: Openstack Object Storage (Swift)

Figure 12: OOReBAC Implementation
Outline

- Introduction
- Comparison of ReBAC and ABAC
- Object-to-Object Relationship Based Access Control: Model and Multicloud demonstration
- Safety and Expressive Power Comparison of $\textit{ABAC}_\alpha$ and its Enhancements
- Conclusion
Figure 13: \textit{ABAC}_\alpha \text{ Model [Jin et al. 2012]}
Precondition Constraints on target subject/object attributes in creating and non creating command.

Precondition Constraint on source subject attributes in non creating command.

Figure 14: $UCON_{\text{finite}}^{\text{preA}}$ Model
<table>
<thead>
<tr>
<th></th>
<th>$ABAC_\alpha$</th>
<th>$UCON^{finite}_{preA}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute Value Structure</td>
<td>Atomic and set valued</td>
<td>Atomic valued</td>
</tr>
<tr>
<td>Attribute Value Scope</td>
<td>finite entity + Non-entity</td>
<td>Non-entity</td>
</tr>
<tr>
<td>Boundedness of Attr. Range</td>
<td>finite</td>
<td>finite</td>
</tr>
<tr>
<td>Attribute Association</td>
<td>No context / meta attribute</td>
<td>No context/meta attribute</td>
</tr>
<tr>
<td>Attribute Mutability</td>
<td>Immutable</td>
<td>Mutable</td>
</tr>
<tr>
<td>Entities</td>
<td>User, subject, object</td>
<td>object</td>
</tr>
<tr>
<td>Operations</td>
<td>Configurable Condition + Mandatory update</td>
<td>Command specific precondition + tightly coupled optional update</td>
</tr>
<tr>
<td>Precondition</td>
<td>Configurable Boolean Expression</td>
<td>Command specific Boolean function</td>
</tr>
<tr>
<td>Update value</td>
<td>Direct value from range</td>
<td>Command specific computed value</td>
</tr>
</tbody>
</table>
Figure 15: Central Result

Decidable Safety
[Rajkumar et al 2016]

Undecidable
Simulation of one directional single tape Turing Machine

Decidable
State Matching Reduction

Undecidable
State Matching Reduction

(a) Expressive Power
(b) Safety
In addition to all the features of $ABAC^\alpha$ , $ABAC^\alpha_{AM}$ has the following properties:

1. Subject can create, delete or modify another subject and at the same time can modify its own attribute value.
2. Subject can modify itself.
3. Subject modification by user can modify user’s own attribute value.

In addition to all the features of $ABAC^\alpha_{AM}$ , $ABAC^\alpha_{MI}$ has the following properties:

- Infinite domain entity attribute.
Outline

➢ Introduction
➢ Comparison of ReBAC and ABAC
➢ Object-to-Object Relationship Based Access Control: Model and Multicloud demonstration
➢ Safety and Expressive Power Comparison of $ABAC_{\alpha}$ and its Enhancements
➢ Conclusion
The most general form ABAC and ReBAC are equivalent. The relationship between less general ABAC and ReBAC is subtle and variable depending on the precise flavor of these two access control approaches in any given model.

OOReBAC is the first attempt towards using object relationship independent of user in authorization policy specification. Its application is possible for multicloud resource sharing in Openstack object storage Swift.

Safety and Expressive power of an ABAC model depend onto the detail of that model.
Conclusion: Future Work

This work can be expanded in many directions:

• Formal definition of specific ReBAC and its structural equivalent ABAC model would bring more realistic result for theoretical equivalence.

• To better understand the relative advantages and disadvantages of ReBAC and ABAC we can consider metrics beyond theoretical equivalence such as performance, maintainability, robustness, and agility.

• OOReBAC model can be extended to accommodate multiple type asymmetric relationships to configure version control and object oriented system.

• Application of relationship based authorization policy in various fields such as IoT.
Conference Papers (Published):


Journal Papers (Work in Progress):

1. Tahmina Ahmed and Ravi Sandhu, “The $ABAC_{\alpha}$ Model: An Enhancement of $ABAC_{\alpha}$ Equivalent to $UCON_{preA}^{finite}$”

Questions/Comments