I·C·S
The Institute for Cyber Security

C·SPECC
Center for Security and Privacy Enhanced Cloud Computing

Safety and Consistency of Mutable Attributes using Quotas: A Formal Analysis

Mehrnoosh Shakarami
Ravi Sandhu

Institute for Cyber Security (ICS)
Center for Security and Privacy Enhanced Cloud Computing (C-SPECC)
Department of Computer Science
University of Texas at San Antonio

The First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS'19)
Los Angeles, CA, USA – Dec 12-14, 2019

World-Leading Research with Real-World Impact!

UTSA Computer Science

1 **Introduction & Motivation**

What is Attribute Based Access Control?

Why I should care about consistency problem?

2 **Background and Preliminaries**

Previous Research

Preliminaries: Mutability and Quota-based Approach

3 **Underlying Assumptions and Proposed Solution**

System Assumptions and Practical Use Cases

Consistency Considerations, Level details and properties

4 **Discussion, Conclusion and Future Work**

Limitations and Practical Issues

What has been done? What to do next?

Introduction

- Access control regulates access to protected resources in the system with respect to the policy.

SUBJECT

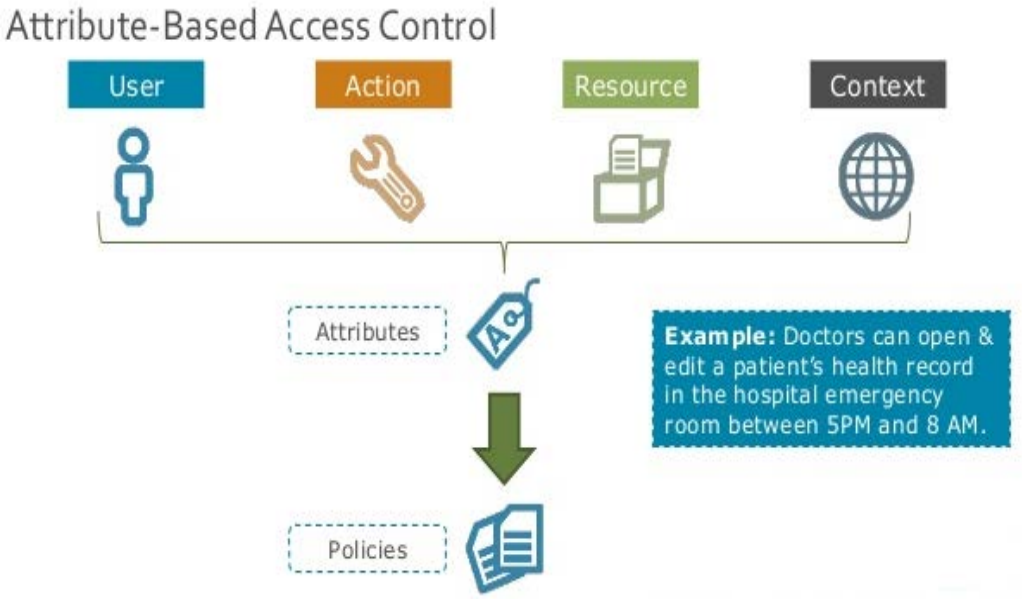
Generally an individual, process, or device causing information to flow among objects or change to the system state.

OBJECT

System-related protected entity (e.g., devices, files, records, tables, processes, programs, domains) containing or receiving information.

Policy

A set of rules which regulates access of subjects to protected objects in the system.



Consistency Problem

- Consistency Problem: When multiple attributes are involved, consistency problem results in granting access when it should be denied (safety violation) or denying access when it should be granted (availability violation), due to following reasons:
  - Asynchronous nature of distributed systems
  - Cached values of attributes
  - Network and system failures
  - Incremental assembly of subject attributes
  - Differing validity periods for subject attribute values

1
Introduction & Motivation
What is Attribute Based Access Control?
Why I should care about consistency problem?

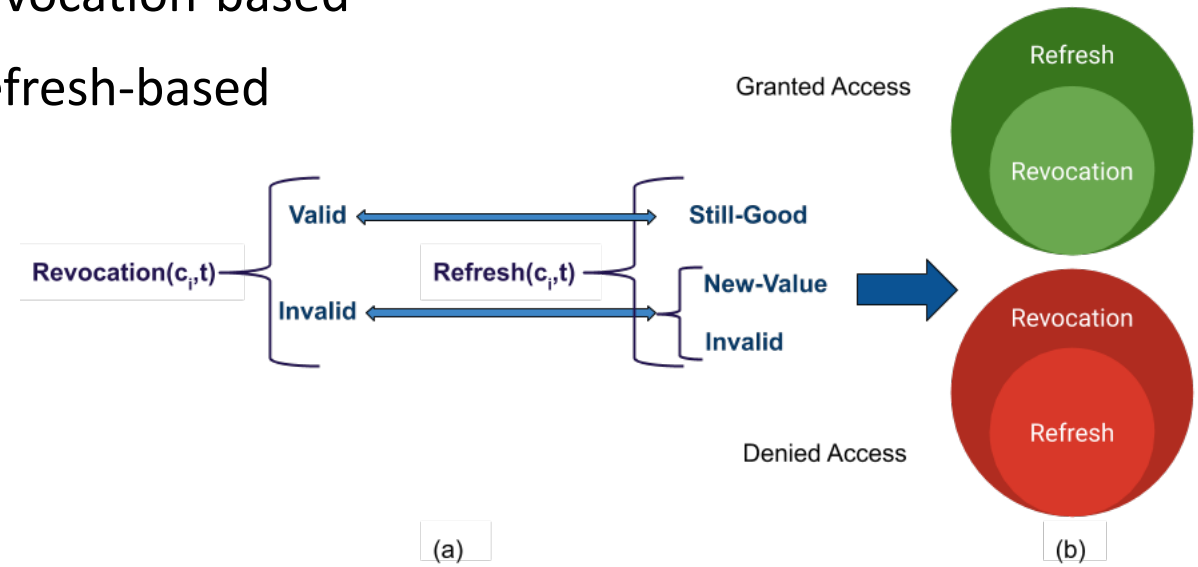
2
Background and Preliminaries
Previous Research
Preliminaries: Mutability and Quota-based Approach

3
Underlying Assumptions and Proposed Solution
System Assumptions and Practical Use Cases
Consistency Considerations, Level details and properties

4
Discussion, Conclusion and Future Work
Limitations and Practical Issues
What has been done? What to do next?

Previous Research

- Safety and consistency in trust negotiation [Lee-Winslett, CCS'06]
- Safety and consistency in ABAC
  - Revocation-based
  - Refresh-based



- Consistency of Mutable attribute: this work

Mutablility and Consistency

- First introduced in $UCON_{ABC}$: a family of access control models to extend traditional access control.
- Mutability: attribute changes as a side effect of access
  - Account balance changes after each payment
- Mutability adds further complication to safety and consistency management, as modification of attribute values should be done in a trusted way to avoid outdated values

Quota-based Approach

- Quota:
  - For reusable resources
  - For consumable resources
- Each mutable attribute has a global limit known to AA which could be managed centrally or be distributed to local servers
- Quota apportion could be done through:
  - Service-based
  - User-based

1
Introduction & Motivation
What is Attribute Based Access Control?
Why I should care about consistency problem?

2
Background and Preliminaries
Previous Research
Preliminaries: Mutability and Quota-based Approach

3
Underlying Assumptions and Proposed Solution
System Assumptions and Practical Use Cases
Consistency Considerations, Level details and properties

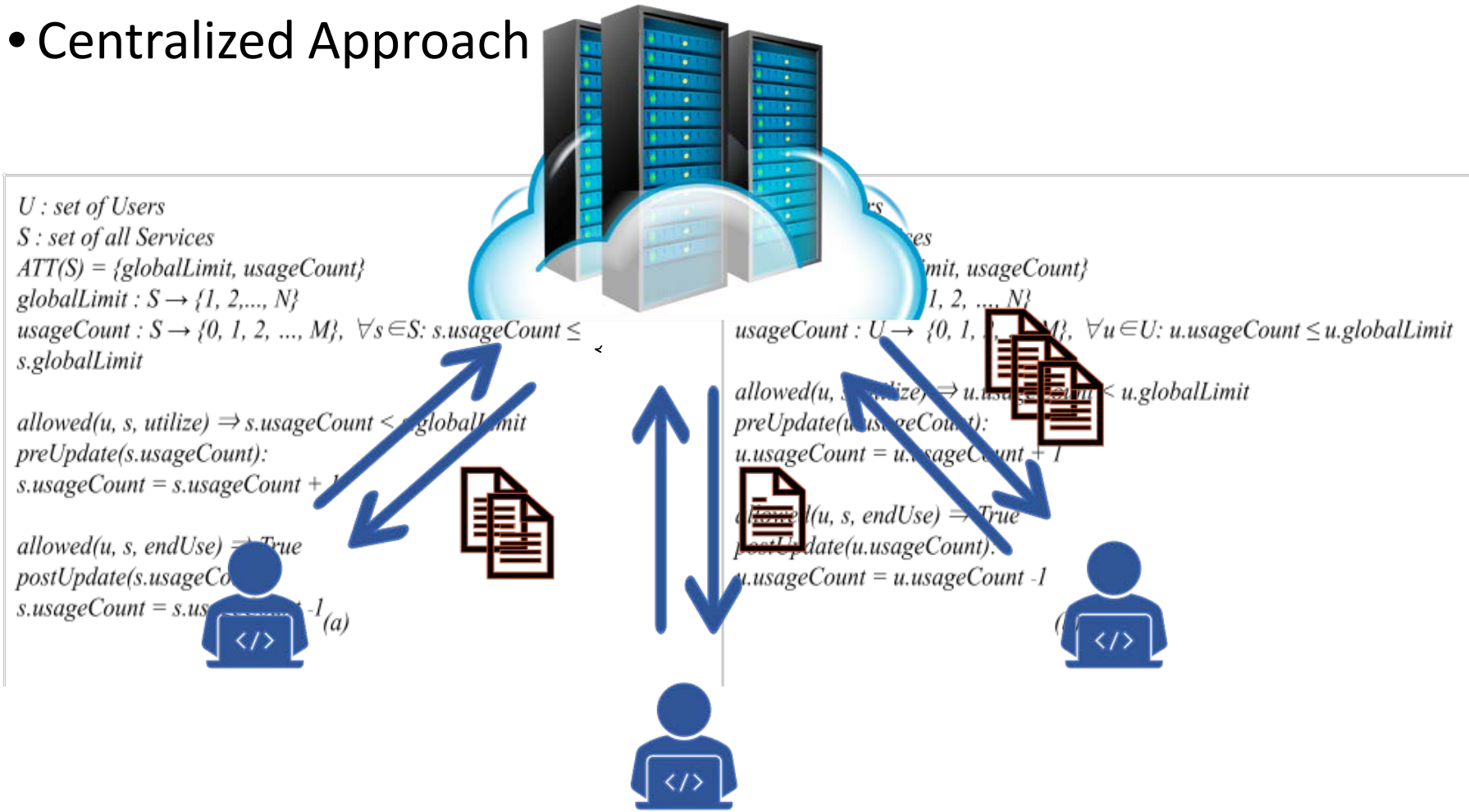
4
Discussion, Conclusion and Future Work
Limitations and Practical Issues
What has been done? What to do next?

Underlaying Assumptions

- An ABAC model is in place
- Attribute credential provision is through multiple authorities
- Administrative changes are always done through AA
- Quota-based approach has been applied to manage concurrency while using multiple authorities
- Decision point is the entity which determines the set of relevant attributes

Practical Use Case Scenarios(1)

- Centralized Approach



Practical Use Case Scenarios(2)

- Distributed approach:

U : set of Users
S : set of Services
SIO : set of Service Instance Objects
ATT(S) = {globalLimit, SIOSet}
ATT(SIO) ={Quota, usageCount, SIOUsers}
globalLimit : S → {1, 2, ..., N}
SIOSet : S → 2^{SIO}
Quota : SIO → {0, 1, 2, ...,M}
usageCount : SIO → {0, 1, 2, ..., C}
SIOUsers: SIO → 2^{U}

allowed(s, sio, create(q)) ⇒ ((q ≤ s.globalLimit) ∧ ((s.globalLimit - ∑ $_{\forall sio \in s.SIOSet}$ sio.Quota) > q)
preUpdate(sio.Quota): sio.Quota = q
preUpdate(sio.usageCount): sio.usageCount = 0
preUpdate(s.SIOSet): s.SIOSet = s.SIOSet ∪ {sio}
preOBL ⊂ OBS × OBO × OB
OBS = {u}
OBO = {sio}
OB = {enduse}
getPreOBL: S × SIO × {delete} → {True, False}
preFulfilled: OBS × OBO × OB → {True, False}
getPreOBL(s, sio, delete) = { ∀u ∈ sio.SIOUsers: allowed(u, sio, enduse)}
allowed(s, sio, delete) ⇒ preFulfilled(getPreOBL(s, sio, delete))
postUpdate(s.SIOSet): s.SIOSet = s.SIOSet \ {sio}

allowed(u, sio, utilize) ⇒ sio.usageCount < sio.Quota
preUpdate(sio.usageCount): sio.usageCount = sio.usageCount + 1
preUpdate(sio.SIOUsers): sio.SIOUsers = sio.SIOUsers ∪ {u}
allowed(u, sio, enduse) ⇒ True
postUpdate(sio.usageCount): sio.usageCount = sio.usageCount - 1
postUpdate(sio.SIOUsers): sio.SIOUsers = sio.SIOUsers \ {u}

(a)

...stances
...t, UISet}
...geCount}
..., N}
..., M};M < N
usageCount : UI → {0, 1, 2, ..., C};C < M

allowed(u, ui, create(q)) ⇒ (u.globalLimit - ∑ $_{\forall ui \in u.UISet}$ ui.Quota) > q
preUpdate(ui.Quota): ui.Quota = q
preUpdate(ui.usageCount): ui.usageCount = 0
preUpdate(u.UISet): u.UISet = u.UISet ∪ {ui}
preOBL ⊂ OBS × OBO × OB
OBS = {ui}
OBO = {s}
OB = {enduse}
getPreOBL: U × UI × {delete} → {True, False}
preFulfilled: OBS × OBO × OB → {True, False}
getPreOBL(u, ui, delete) = {allowed(ui, s, enduse)}
allowed(u, ui, delete) ⇒ preFulfilled(getPreOBL(u, ui, delete))
postUpdate(u.UISet): u.UISet = u.UISet \ {ui}

allowed(ui, s, utilize) ⇒ ui.usageCount < ui.Quota
preUpdate(ui.usageCount): ui.usageCount = ui.usageCount + 1
allowed(ui, s, enduse) ⇒ True
postUpdate(ui.usageCount): ui.usageCount = ui.usageCount - 1

(b)

Distributing Servers

Distributed vs. Centralized Approach

- Property 1:

  Centralized quota management provides correct access control decision.

- Property 2:

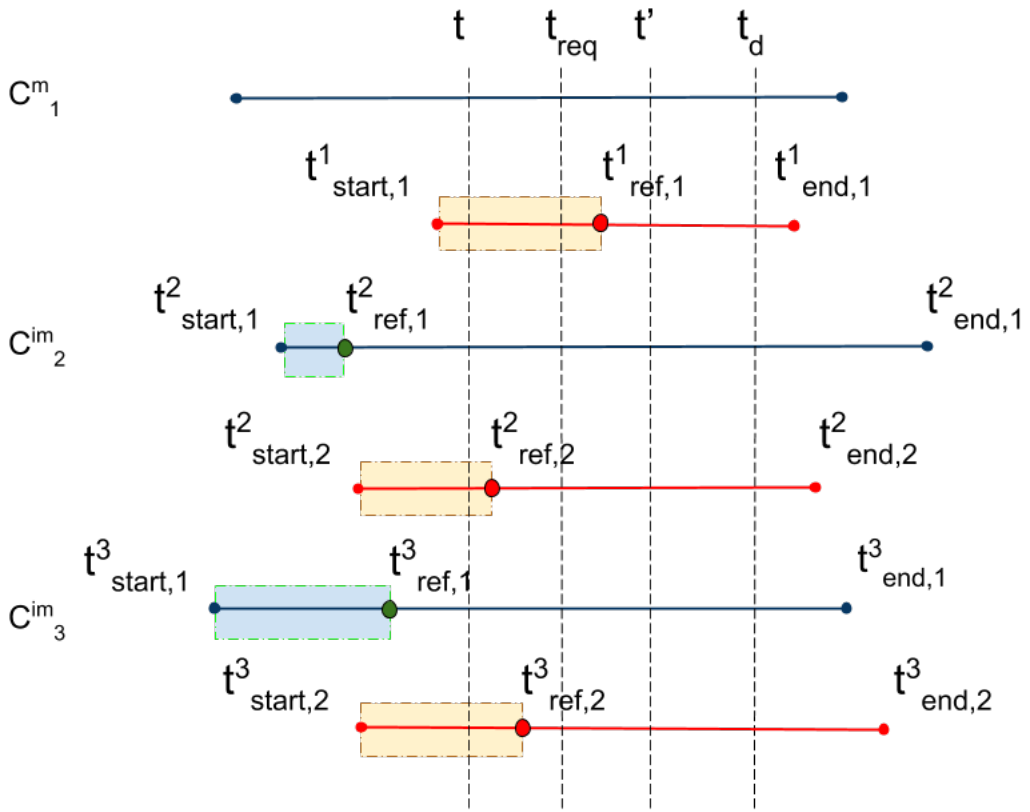
  Distributed quota management approach provides less availability and less utilization, comparing to the centralized approach.

- Property 3:

  Distributed quota management access provision is correct.

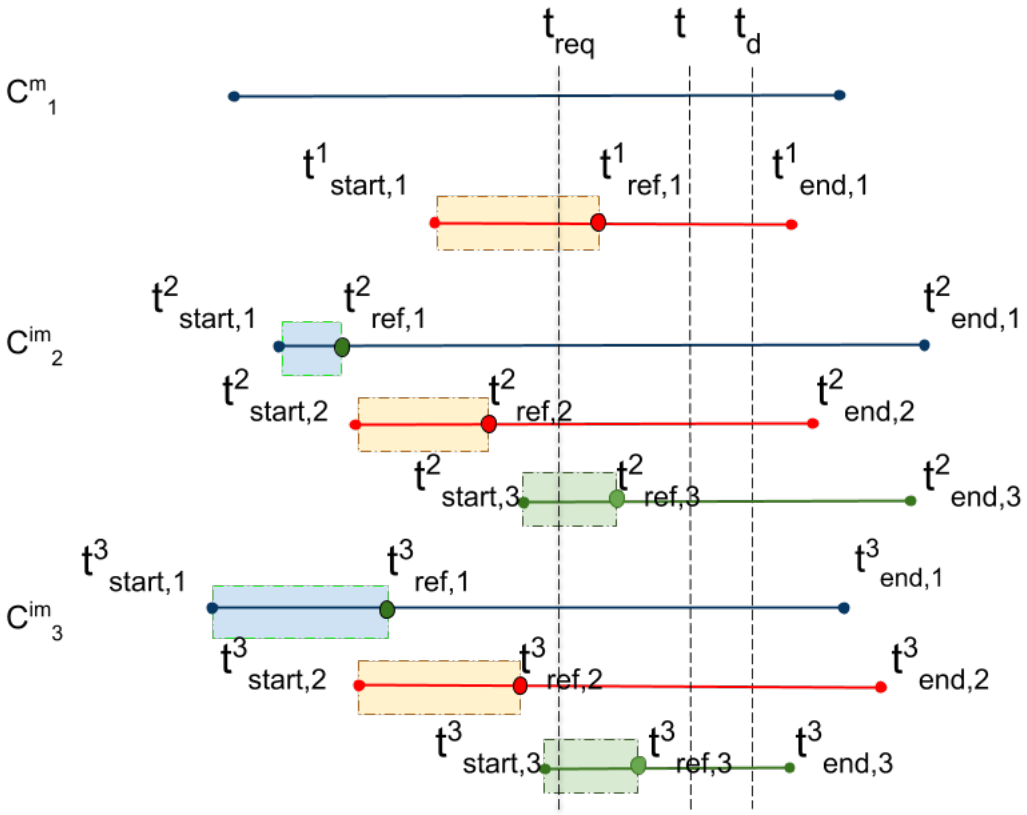
Proposed Consistency Levels

- Lifetime Overlap



Proposed Consistency Levels

- Freshness Overlap



1
Introduction & Motivation
What is Attribute Based Access Control?
Why I should care about consistency problem?

2
Background and Preliminaries
Previous Research
Preliminaries: Mutability and Quota-based Approach

3
Underlying Assumptions and Proposed Solution
System Assumptions and Practical Use Cases
Consistency Considerations, Level details and properties

4
Discussion, Conclusion and Future Work
Limitations and Practical Issues
What has been done? What to do next?

Consistency Considerations for Mutable Attributes

- We observe revocation as inappropriate for mutable attributes, so we used refresh.
- As for immutable attribute, consistency problem arise only if there are multiple attributes in relevant attribute sets

Conclusion and Future Work

- The safety and availability of mutable in multi-authority distributed ABAC systems has been formally characterized.
- The revocation scenario claimed to be inappropriate for mutable attributes.
- We proposed two consistency levels which are totally ordered in strictness.

Some future research directions:

- Other access control information is subject to staleness, e.g. policy and object attributes.
- Models could be developed for ongoing authorization.

Thank You