# COMPOSITION OF ACCESS CONTROL SCHEMES VIA COEXISTENCE: THREE CASE STUDIES

11/17/2026

**Indrani Ray**, Mahesh Tripunitara, Kami Vaniea

{**iray**, tripunit, kvaniea}@uwaterloo.ca

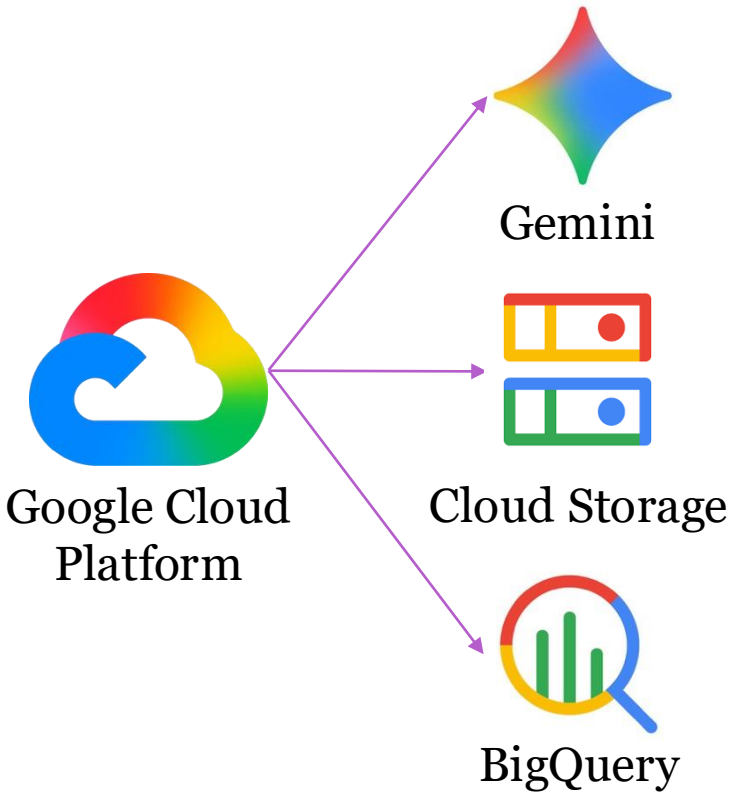
UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

# Problem: Multiple Access Controls in a Single System

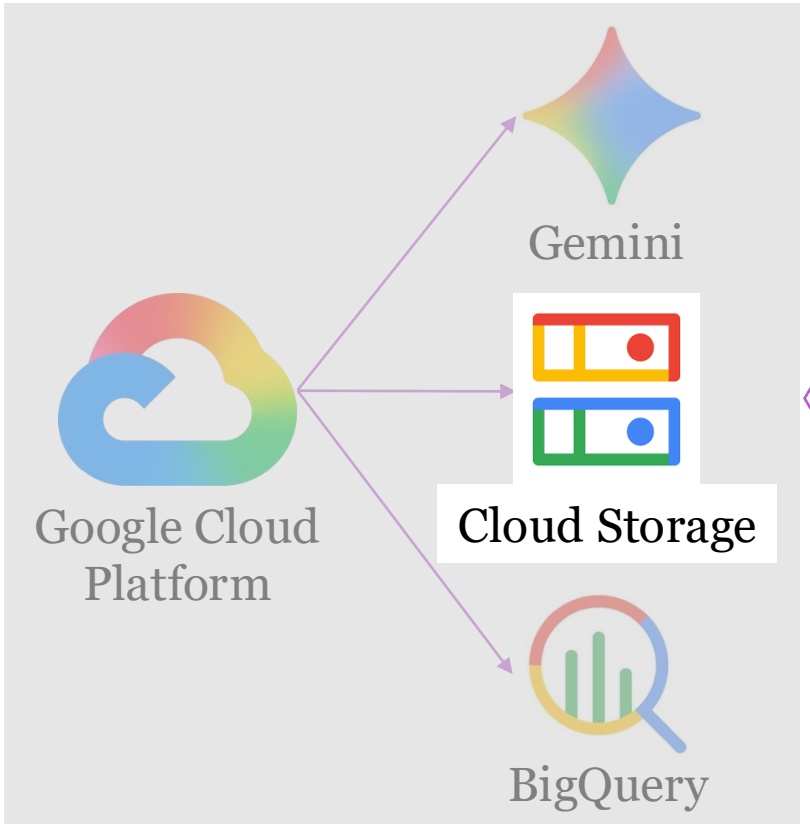


Google Cloud Platform

# Problem: Multiple Access Controls in a Single System

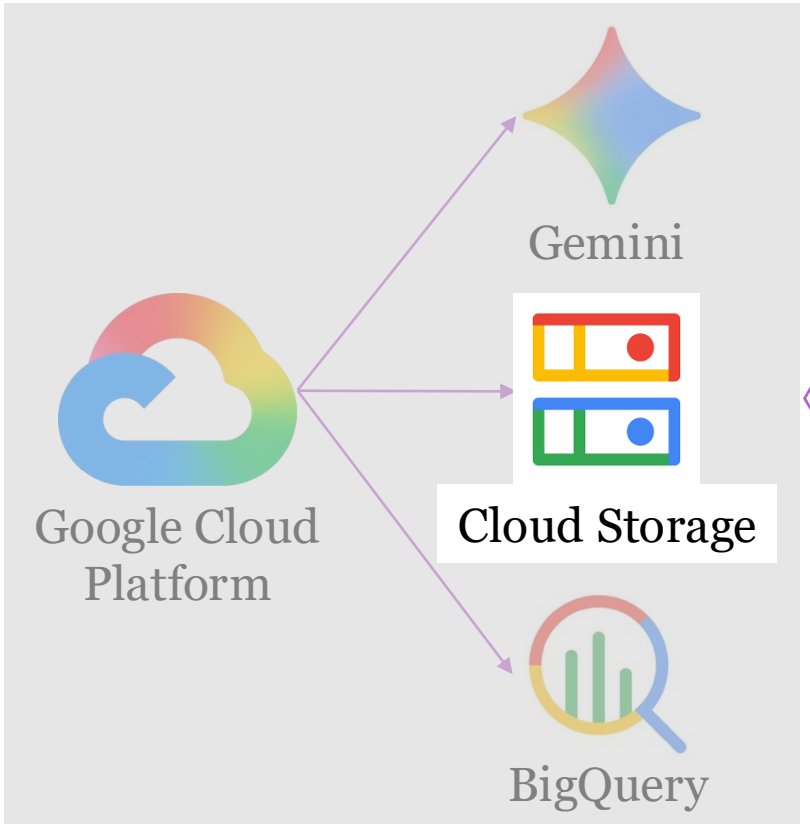


# Problem: Multiple Access Controls in a Single System






# Problem: Multiple Access Controls in a Single System




IAM

Resource

bucket_name_taken

Add principals

New principals *

abchild88@gmail.com

Assign roles

Role *

Storage Admin

Grants full control of buckets and objects.

IAM condition (optional)

IAM conditions disabled

ACL

**Object name:** nalgene_cap.jpg

| Entity 1 * | Name 1 | Access 1 * |
|---|---|---|
| User | abchild88@gmail.c | Reader |
| Entity 2 * | Name 2 | Access 2 * |
| User | rayindrani100@gm | Owner |

# How Should They Compose?

- Prior work: (1) policy composition logic or (2) encode access control models
  - Are not based on real-world implementations
- **We consider how actual implementations co-exist to derive schemes (authorization state-transition systems)**

| Google Cloud Storage | Amazon Simple Storage Service | Oracle Database |
|---|---|---|
| Identity and Access Management | Identity-based | Role-based |
| Access Control List | Resource-based | Discretionary-based |

# How Should They Compose?

- Prior work: (1) policy composition logic or (2) encode access control models
  - Are not based on real-world implementations
- **We consider how actual implementations co-exist to derive schemes (authorization state-transition systems)**

| Google Cloud Storage | Amazon Simple Storage Service | Oracle Database |
|---|---|---|
| Identity and Access Management | Identity-based | Role-based |
| Access Control List | Resource-based | Discretionary-based |

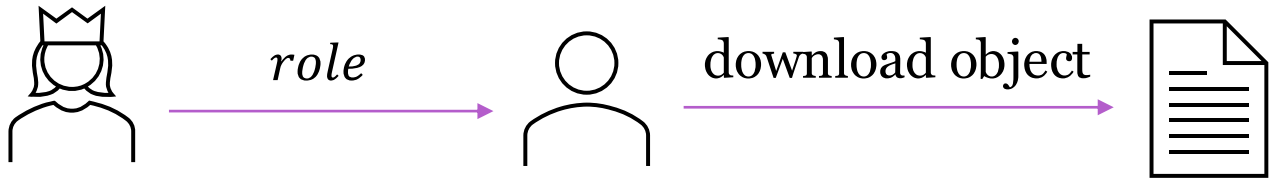
# Before we Begin:

- *Project*: "fundamental organizing entity…essential for creating, enabling, and using Google Cloud services."
- Principals: Google Cloud users
- Resources: *buckets* (containers) and *objects* (uploaded data).
  - A bucket contains zero or more objects; we represent this as function $containedIn: O \rightarrow B$.
- Two access controls:
  - Identity and Access Management (IAM)
  - Access Control List (ACL)

# Outline

- Example IAM data collection, state, and transition (slides 8-10)
- Example ACL data collection, state, and transition (slides 11-13)
- Composition of IAM scheme and ACL scheme (slides 14-16)

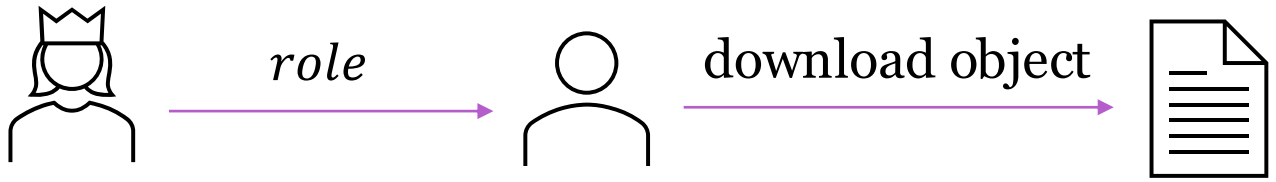
# IAM Data Collection Example




*role*

```
storage.buckets.create
storage.objects.list
firebase.projects.get
resourcemanager.projects.get
storage.objects.get
storage.buckets.getIAMpolicy
…
```

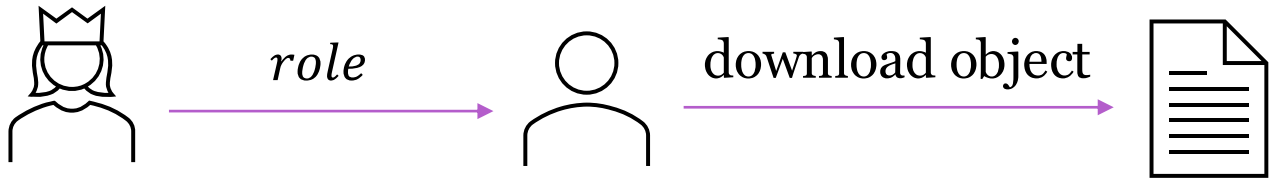
# IAM Data Collection Example




*role*

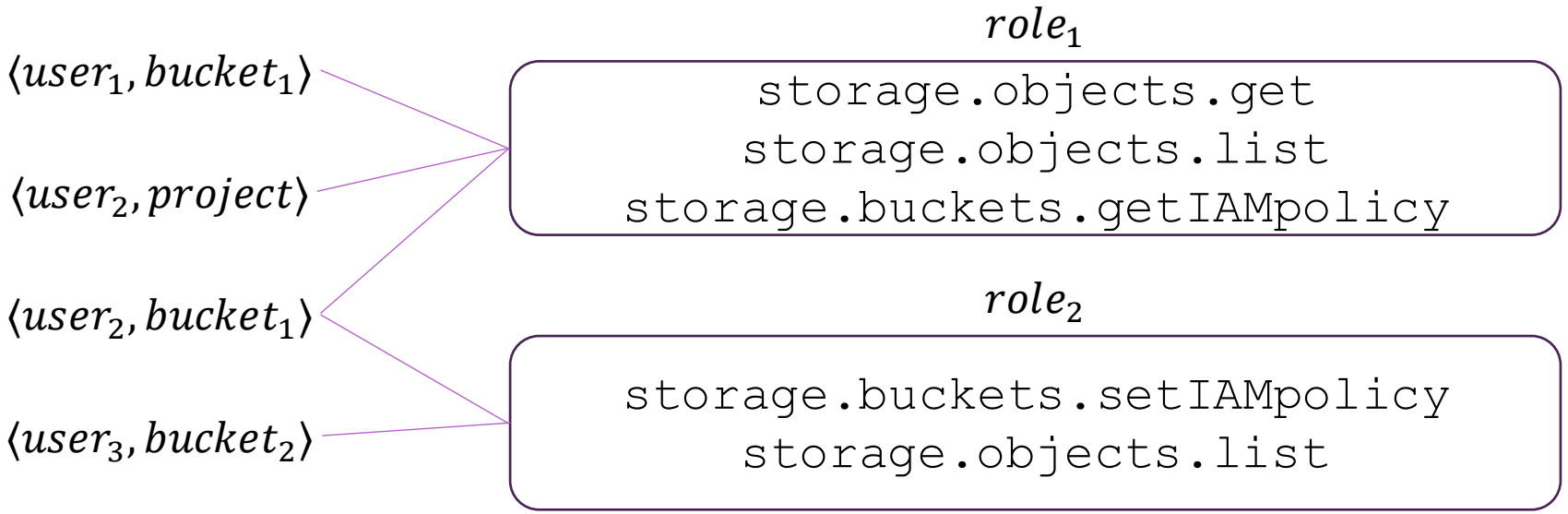
~~storage.buckets.create~~
storage.objects.list
~~firebase.projects.get~~
~~resourcemanager.projects.get~~
storage.objects.get
~~storage.buckets.getIAMpolicy~~
…

# IAM Data Collection Example



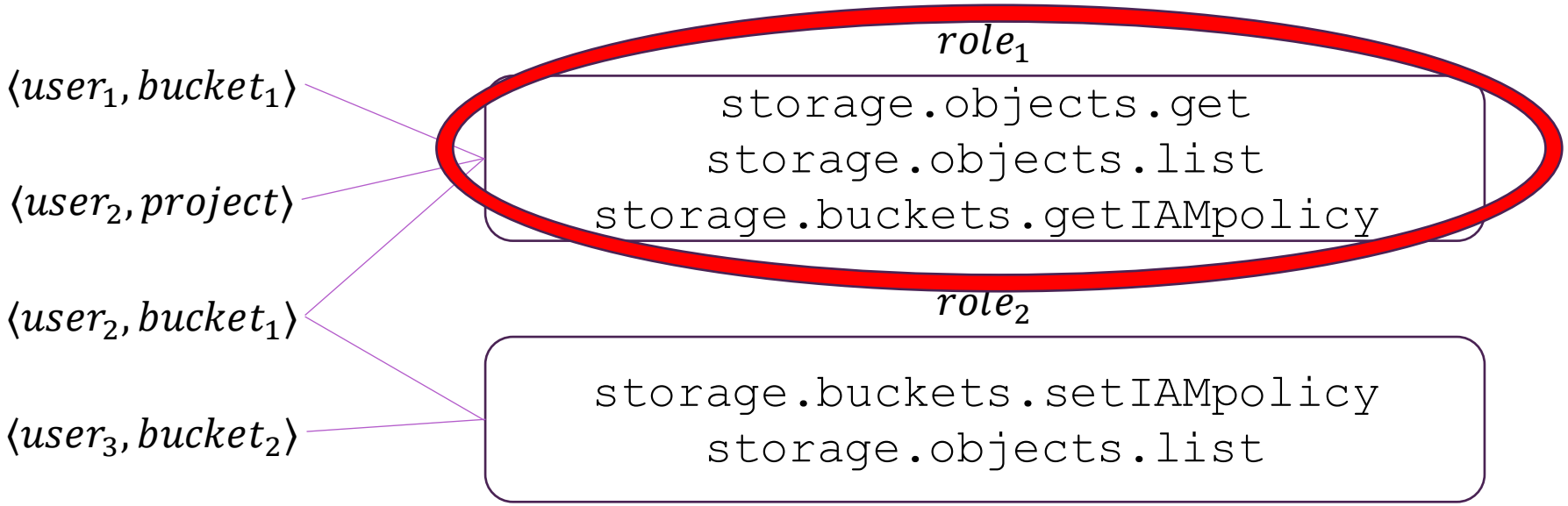

*role*
storage.objects.list
storage.objects.get

# IAM Authorization State Example



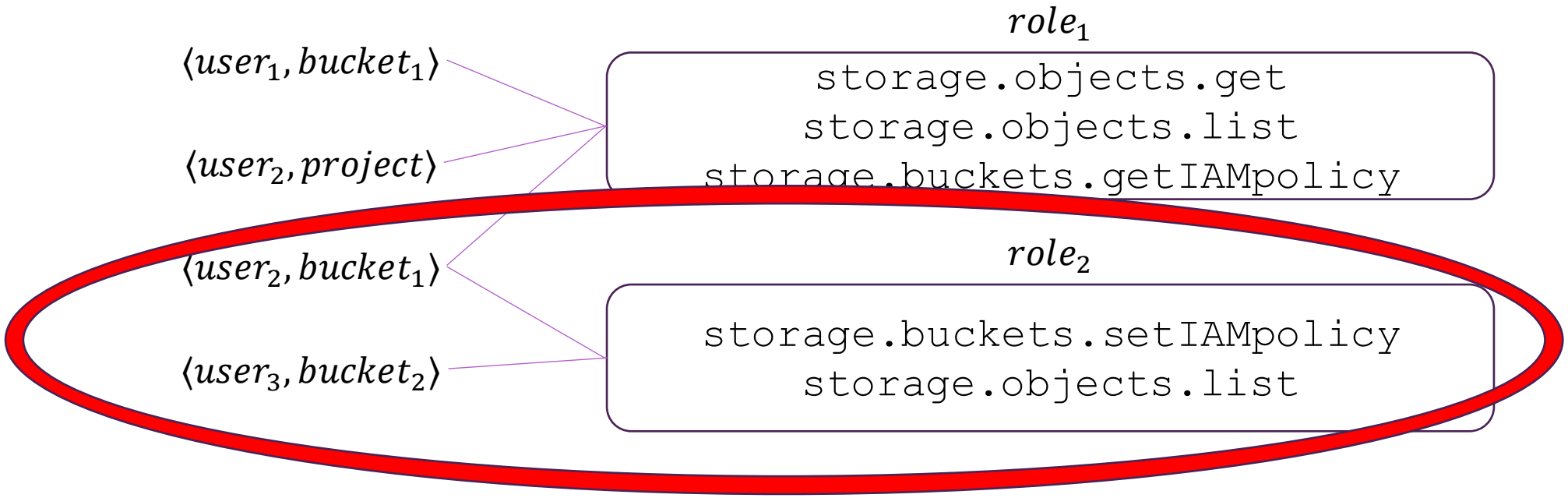

- Users ($U_1$), Buckets ($B_1$), Objects ($O_1$), Permissions ($P_1$), Roles ($L$), Project ($\pi$)
- Functions: $ul: L \rightarrow 2^{\{U_1 \times (B_1 \cup \{\pi\})\}}$, and $lp: L \rightarrow 2^{P_1}$

# IAM Authorization State Example



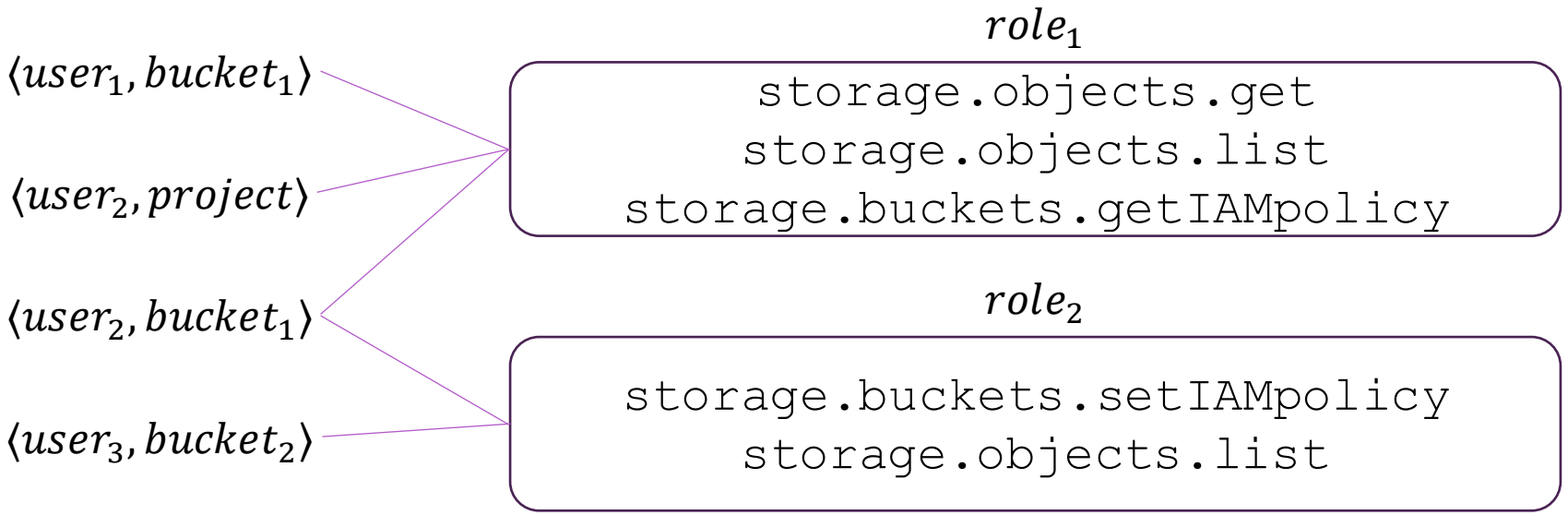

- Users ($U_1$), Buckets ($B_1$), Objects ($O_1$), Permissions ($P_1$), Roles ($L$), Project ($\pi$)
- Functions: $ul: L \rightarrow 2^{\{U_1 \times (B_1 \cup \{\pi\})\}}$, and $lp: L \rightarrow 2^{P_1}$

# IAM Authorization State Example




- Users ($U_1$), Buckets ($B_1$), Objects ($O_1$), Permissions ($P_1$), Roles ($L$), Project ($\pi$)
- Functions: $ul: L \rightarrow 2^{\{U_1 \times (B_1 \cup \{\pi\})\}}$, and $lp: L \rightarrow 2^{P_1}$

# IAM Authorization State Example



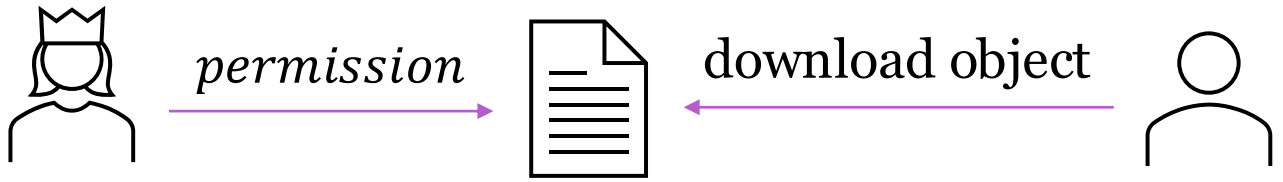

Users ($U_1$), Buckets ($B_1$), Objects ($O_1$), Permissions ($P_1$), Roles ($L$), Project ($\pi$)

Functions: $ul: L \rightarrow 2^{\{U_1 \times (B_1 \cup \{\pi\})\}}$, and $lp: L \rightarrow 2^{P_1}$

# IAM State Transition Example

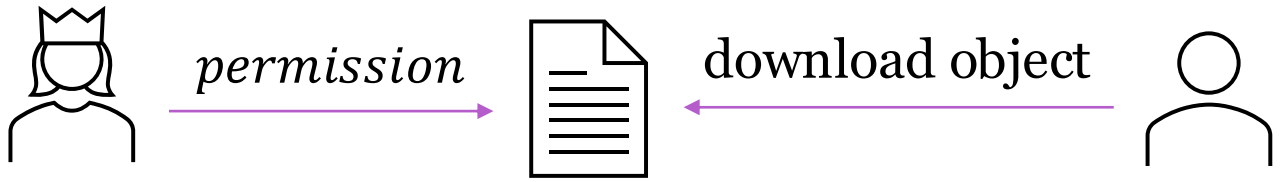
- `remove role from user on bucket`
- For user $i$ (initiator) to remove role $l$ from user $u$ on bucket $b$:
  - There is a subset of roles: $\exists L_1 \subseteq L$
  - Each role in the subset is assigned to the initiator on the bucket or project: for $l_i \in L_1$, $\langle i, \pi \rangle$ or $\langle i, b \rangle \in ul(l_i)$
  - The roles collectively contain two specific permissions: `storage.buckets.getIAMpolicy`, `storage.buckets.setIAMpolicy` $\in \bigcup_i lp(l_i)$

# ACL Data Collection Example



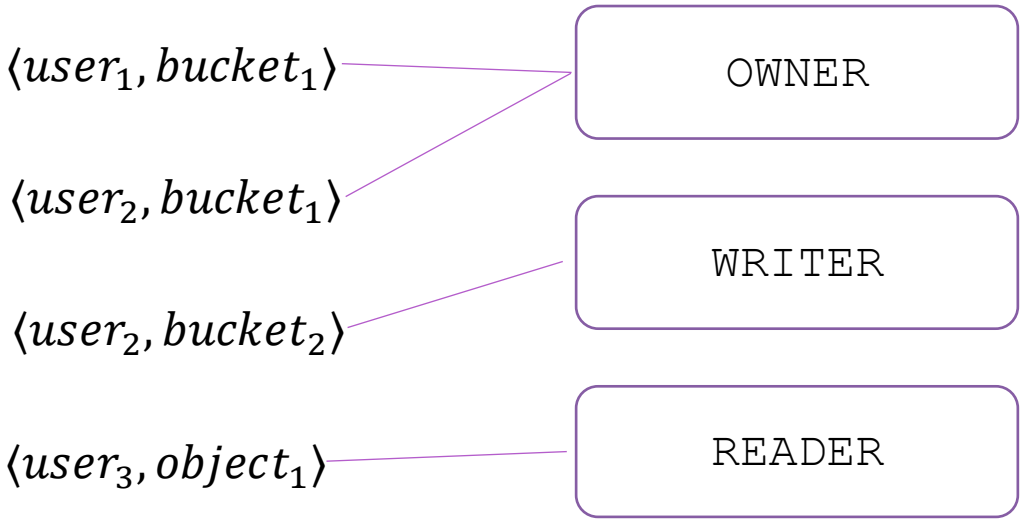

| *permission* |
| --- |
| OWNER |

# ACL Data Collection Example



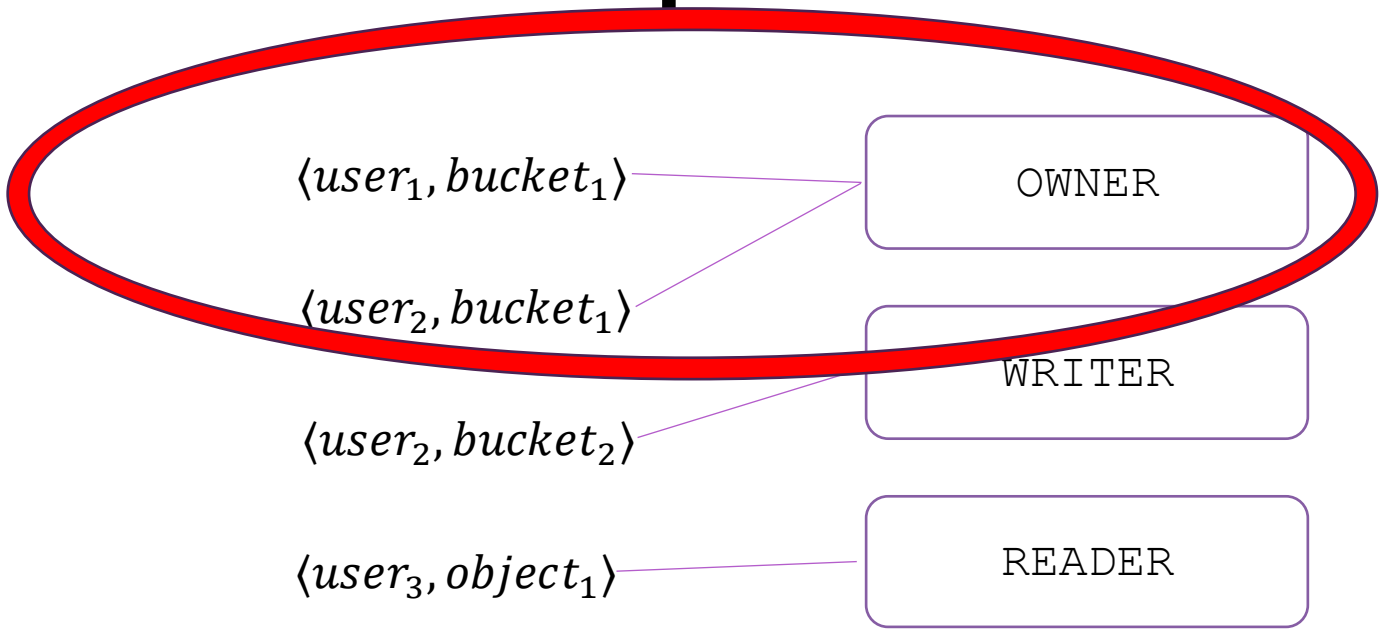

| *permission* |
| --- |
| READER |

# ACL Authorization State Example



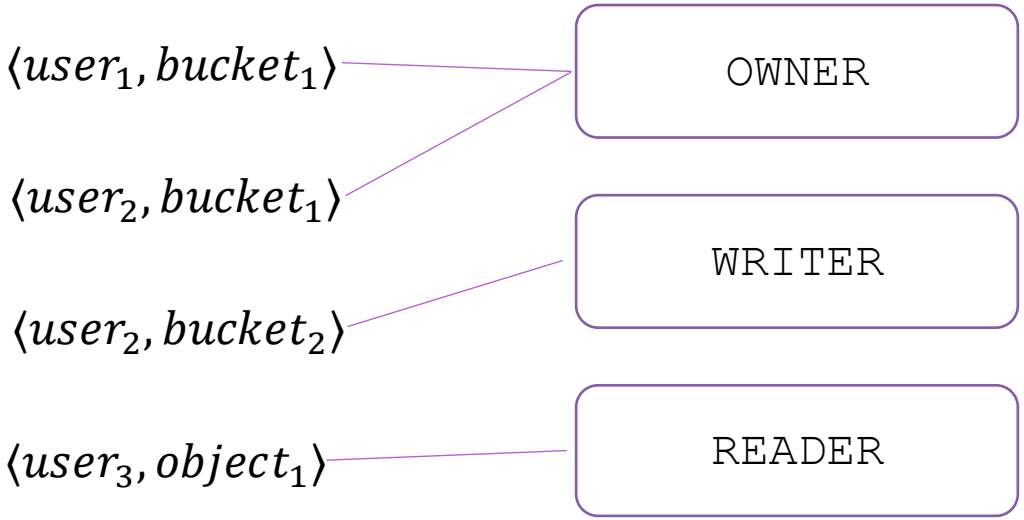

- Users ($U_2$), Buckets ($B_2$), Objects ($O_2$), Permissions ($P_2$)
- Functions $up$: $P_2 \rightarrow 2^{\{U_2 \times (B_2 \cup O_2)\}}$

# ACL Authorization State Example




- Users ($U_2$), Buckets ($B_2$), Objects ($O_2$), Permissions ($P_2$)
- Functions $up: P_2 \rightarrow 2^{\{U_2 \times (B_2 \cup O_2)\}}$

# ACL Authorization State Example



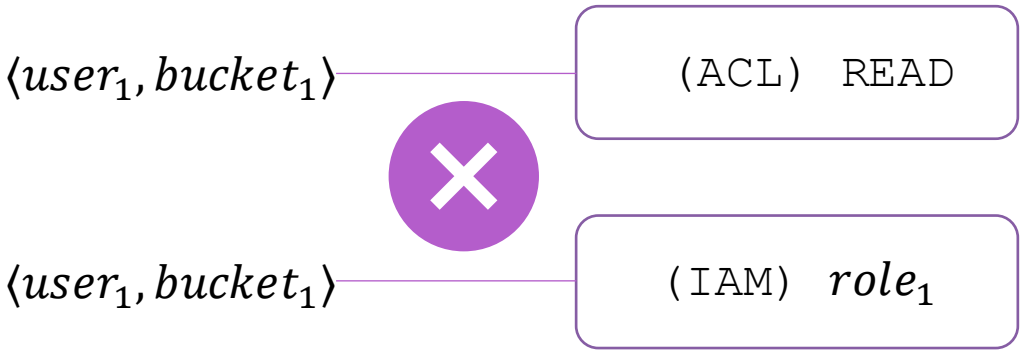

- Users ($U_2$), Buckets ($B_2$), Objects ($O_2$), Permissions ($P_2$)
- Functions $up$: $P_2 \rightarrow 2^{\{U_2 \times (B_2 \cup O_2)\}}$

# ACL State Transition Example

- `change permission on bucket`
- For user $i$ (initiator) to change permission $p_1$ to $p_2$ for user $u$ on bucket $b$:
  - The initiator must be the owner of the bucket: $\langle i, b \rangle \in up(\texttt{OWNER})$

# Composition of IAM and ACL Schemes

- Not all states compose.

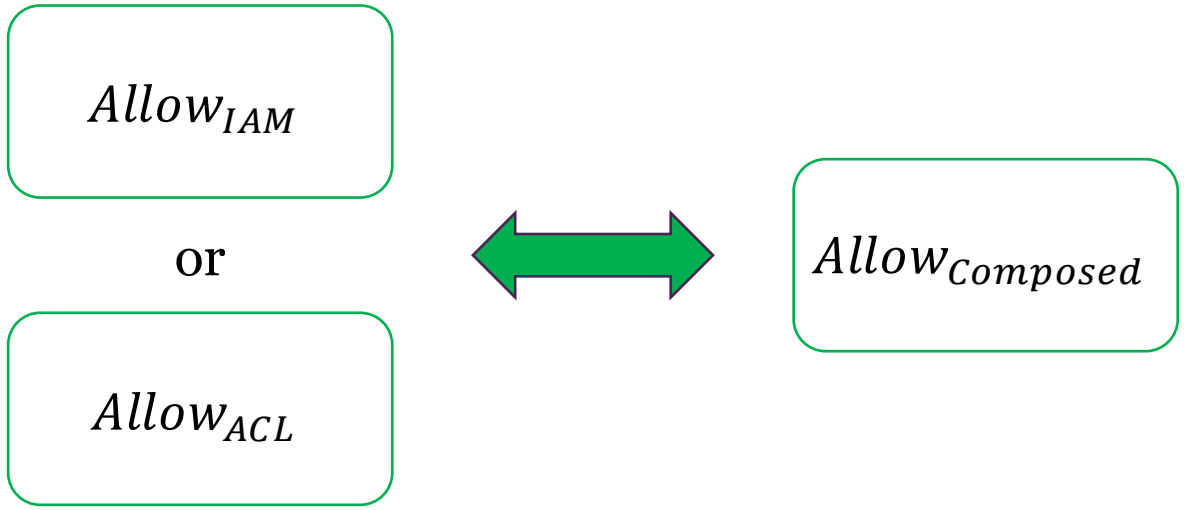


# Composition of IAM and ACL Schemes

- Not all states compose.
- Composition: unions of principals and resources + component constructs and functions. Transitions are prior ones.

# Composition of IAM and ACL Schemes

- Not all states compose.
- Composition: unions of principals and resources + component constructs and functions. Transitions are prior ones.
- Access allowed i.f.f. allowed via one of the component schemes



# Observations

| Google Cloud Storage | Amazon Simple Storage Service | Oracle Database |
|---|---|---|
| • Not all states compose<br>• No denials<br>• State reflection<br>• Cross-scheme authorization<br>• ACL "dependent" on IAM | • All states compose<br>• Deny-override<br>• Resource-based "dependent" on identity-based | • All states compose<br>• No denials<br>• Cross-scheme authorization<br>• Role-based "dependent" on discretionary-based |

Detailed in Paper

Name: Indrani Ray

Email: iray@uwaterloo.ca

LinkedIn: indrani-ray

# Summary

- Analyzed storage platforms– each with different access controls protecting the same resources.
- Formalized **access control schemes** and their compositions
  - Built from real systems where different access controls co-exist
  - Capture how authorizations change as opposed to, e.g., static policy compositions




YOU+WATERLOO

*Our greatest impact happens together.*