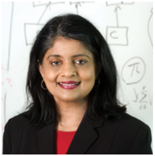
The Category-Based Approach to Access Control, Obligations and Privacy

Maribel Fernández
King's College London

SACMAT 2023

Joint work with

Sandra Alves   Clara Bertolissi   Jenjira Jaimunk   Bhavani Thuraisingham



Motivations

A variety of access control models...

...each with specific languages, techniques, properties.

- RBAC: Role-Based Access Control
- Mandatory (e.g., [Bell-Lapadula] military applications)
- Event-Based (e.g., DEBAC in banking applications)
- ABAC: Attribute-Based Access Control
- ...

⇒ An Access Control MetaModel [Barker09] based on the primitive notion of a category.

Category-Based MetaModel

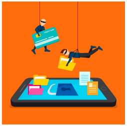
- Core set of principles of access control, can be specialised for specific applications
- Abstracts away complexities of specific access control models
- Formally defined: axiomatic approach
    - to compare policies rigorously
    - understand the consequences of changes
    - prove properties of policies and combinations of policies

In this talk:

- The category based metamodel
- Category-based
  - Access Control: CBAC

  - Obligations: CBAC-O

  - Privacy: CBDA



- Conclusions and future work

The Category-Based Metamodel: entities, relationships, axioms

Entities

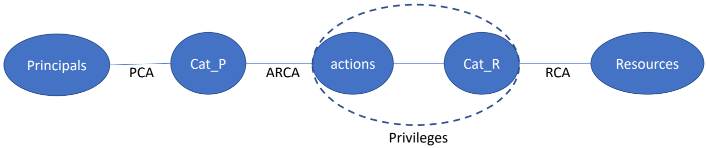
- countable set $\mathcal{C}$ of categories: $c_0, c_1, \ldots$
- countable set $\mathcal{P}$ of principals: $p_1, p_2, \ldots$
- countable set $\mathcal{A}$ of actions: $a_1, a_2, \ldots$
- countable set $\mathcal{R}$ of resources: $r_1, r_2, \ldots$
- countable set $\mathcal{S}$ of situational identifiers (locations, times)

Category: class, group, or domain, to which entities belong

Particular cases:
role, *security clearance*, *attribute-based*...

Relationships between entities

- Principal-category assignment $\mathcal{PCA}$:
  $(p, c) \in \mathcal{PCA}$ iff $p \in \mathcal{P}$ is assigned to $c \in C$
- Resource-category assignment $\mathcal{RCA}$:
  $(r, c) \in \mathcal{RCA}$ iff $r \in \mathcal{R}$ is assigned to $c \in C$
- Permission-category assignment $\mathcal{ARCA}$:
  $(a, c_r, c_p) \in \mathcal{ARCA}$ iff action $a \in A$ on resource category c_r may be performed by principals in the category c_p
- Authorisations $\mathcal{PAR}$:
  $(p, a, r) \in \mathcal{PAR}$ iff $p \in \mathcal{P}$ may perform action $a \in \mathcal{A}$ on resource $r \in \mathcal{R}$



Axioms

Core axiom:

$$(a1)\quad \forall p \in \mathcal{P},\ \forall a \in \mathcal{A},\ \forall r \in \mathcal{R},$$

$$((\exists c_p, c'_p, c_r, c'_r \in \mathcal{C},$$

$$(p, c_p) \in \mathcal{PCA} \wedge c_p \subseteq c'_p \wedge (r, c_r) \in \mathcal{RCA} \wedge c_r \subseteq c'_r \wedge$$
$$(a, c'_r, c'_p) \in \mathcal{ARCA})$$

$$\Leftrightarrow (p, a, r) \in \mathcal{PAR})$$

$\subseteq$ is a relationship between categories (equality, set inclusion, ...)

Additional relationships and axioms could be added

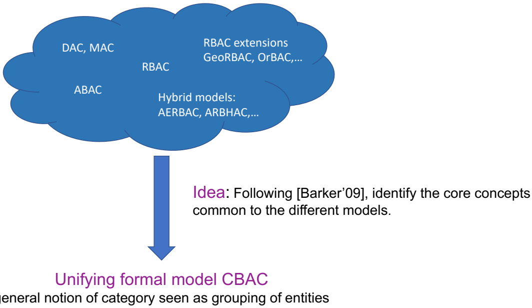
CBAC

Image designed by Freepik

Category-Based Access Control

A basic category based policy is a tuple $\langle \mathcal{E}, \mathcal{PCA}, \mathcal{ARCA}, \mathcal{PAR} \rangle$, where $\mathcal{E} = (\mathcal{P}, \mathcal{C}, \mathcal{A}, \mathcal{R}, \mathcal{S})$, and axiom $(a1)$ is satisfied.

Expressiveness:
A range of existing access control models can be represented as specialised instances of CBAC [Bertolissi and F 2010]:



Category-Based Access Control

Operational semantics: (a_1) is realised through a set of function definitions (rewrite rules) [Bertolissi and F, 2014]

Why rewriting:

- Expressive, multi-paradigm specification language
- Well-developed theory
- Languages and Tools for rapid prototyping/policy analysis: Maude, Tom, CiME

Operational Semantics

Rewrite-based specification of the axiom (a1):

$$(a2) \quad \mathrm{par}(P, A, R) \quad \rightarrow \quad \begin{array}{l} \textit{if} \\ \textit{inARCA}^*(A, \mathrm{contain}(\mathrm{rca}(R)), \mathrm{contain}(\mathrm{pca}(P))) \\ \textit{then}\ \mathrm{grant}\ \textit{else}\ \mathrm{deny} \end{array}$$

grant and deny are answers
pca, rca compute the list of categories of a principal/resource
contain computes the set of categories that contain any of the categories in its input
$\in$ is a membership operator
arca returns the list of all the permissions assigned to the categories in a set

Evaluation

An access request by a principal p to perform the action a on the resource r is evaluated by rewriting $par(p, a, r)$ to normal form.

Proposition:
$\text{par}(p, a, r) \to^* \text{grant}$ if and only if $(p, a, r) \in \mathcal{PAR}$

Example policy

Employees in a company are classified as managers, senior managers or senior executives.

To be categorised as a senior executive (*SeniorExec*), a principal must be a senior manager (*SeniorMng*) according to the information in site ν_1 and must be a member of the executive board.

Any senior executive is permitted to read the salary of an employee, provided the employee works in a profitable branch and is categorised as a Manager (*Manager*).

All managers' names are recorded locally, and the list of profitable branches is kept up to date at site ν_2 .

Example policy

Specific rules (to add to the generic rules computing par):

$$
\begin{array}{lcl}
\text{pca}(P) & \rightarrow & \textit{if}\ \text{SeniorMng} \in \text{pca}_{\upsilon 1}(P) \\
 & & \textit{then}\ (\textit{if}\ \ P \in \text{ExecBoard}\ \textit{then}\ [\text{SeniorExec}] \\
 & & \qquad\quad \textit{else}\ [\text{SeniorMng}]) \\
 & & \textit{else}\ [\text{Manager}] \\
\text{arca}(\text{SeniorExec}) & \rightarrow & \text{zip-read}(\text{managers}(\text{profbranch}_{\upsilon 2})
\end{array}
$$

where

zip-read, given a list $L = [l_1, \ldots, l_n]$, returns a list of pairs $[(\text{read}, l_1), \ldots, (\text{read}, l_n)]$

profbranch, defined at site υ_2, returns the list of profitable branches

manager returns the name of the manager of a branch B given as a parameter (managers does the same for a list of branches).

Properties of policies

Totality: Each request from a valid principal p to perform a valid action a on a resource r receives an answer.

Consistency: For any $p \in \mathcal{P}$, $a \in \mathcal{A}$, $r \in \mathcal{R}$, at most one result is possible for a request $\text{par}(p, a, r)$.

Soundness and Completeness: For any $p \in \mathcal{P}$, $a \in \mathcal{A}$, $r \in \mathcal{R}$, an access request by p to perform the action a on r is granted if and only if p belongs to a category that has the permission (a, r).

Remark:
Totality + consistency $\equiv$ confluence, termination, sufficient completeness

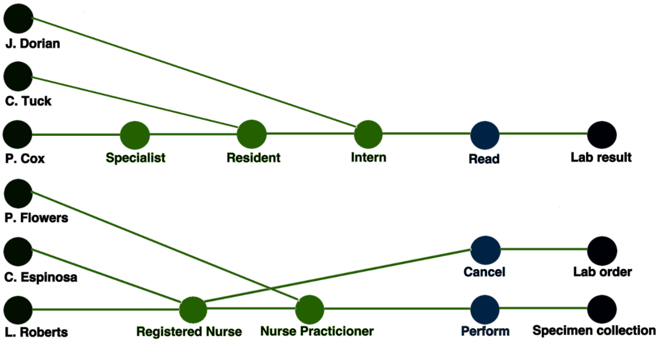
Sufficient conditions: orthogonality [Klop], rpo [Dershowitz], ...

Application: example policy is consistent, total, sound, complete

Policy Specification: Graph-Based Language

A *policy graph* is a tuple $\mathcal{G} = (\mathcal{V}, E, lv, le)$:

- $\mathcal{V}$ is a set of nodes;
- $E \subseteq \{\{v_1, v_2\} \mid v_1, v_2 \in \mathcal{V} \wedge v_1 \neq v_2\}$;
- lv is an injective labelling function $lv : \mathcal{V} \rightarrow \mathcal{C} \cup \mathcal{P} \cup \mathcal{A} \cup \mathcal{R}$;
- le is a labelling function for edges.



Well-typed graph

A well-typed graph contains only the following kinds of edges:

(a) $\{v_1, v_2\} \in E$ s. t. $\text{type}(v_1) = P \wedge \text{type}(v_2) = C_P$,
connects principals to categories — edge of type PC_P

(b) $\{v_1, v_2\} \in E$ s. t. $\text{type}(v_1) = C \wedge \text{type}(v_2) = C$,
connects categories — edge of type CC

(c) $\{v_1, v_2\} \in E$ s. t. $\text{type}(v_1) = C \wedge \text{type}(v_2) = A$,
connects categories to actions — edge of type CA

(d) $\{v_1, v_2\} \in E$ s. t. $\text{type}(v_1) = C \wedge \text{type}(v_2) = R$,
connects categories to resources — edge of type $C_R R$

Relations associated with the graph

$\mathcal{G}$ is a well-typed policy graph
Then:

- $\mathcal{PCA}_{\mathcal{G}} = \{(lv(v_1), lv(v_2)) \mid \text{type}(\{v_1, v_2\}) = PC_p\}$
- $\mathcal{RCA}_{\mathcal{G}} = \{(lv(v_1), lv(v_2)) \mid \text{type}(\{v_1, v_2\}) = RC_R\}$
- $\mathcal{ARCA}_{\mathcal{G}} = \{(lv(v_1), lv(v_2), lv(v_3)) \mid \text{type}(\{v_1, v_2\}) = AC_R, \text{type}(\{v_3, v_1\}) = C_P A\}$
- $\mathcal{PAR}_{\mathcal{G}}$ obtained via path computation (from P to R)

Administrative model: Admin-CBAC in the CBAC metamodel

Implementation

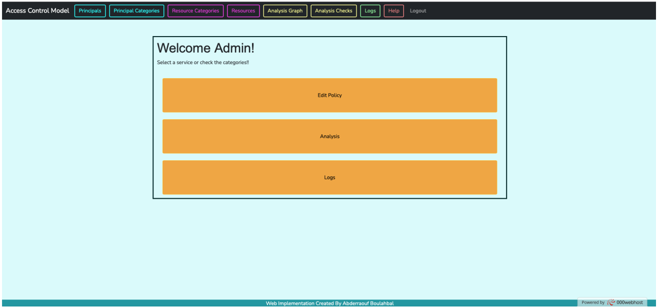


Figure: Interface of the prototype: landing page

Implementation

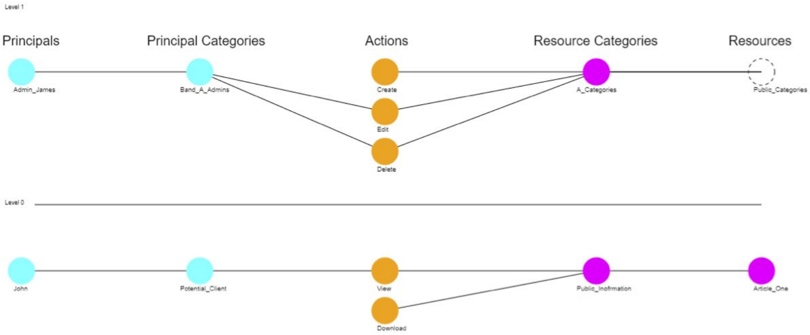


Figure: Test policy

Implementation

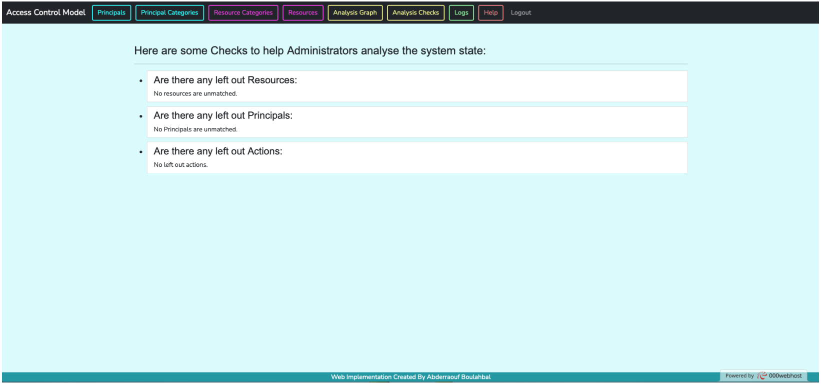


Figure: Interface of the prototype: analysis menu

Key findings

Expressive power:

- entities, relations: generic approach
- Axiomatisation: takes into account multi-site systems, combination of policies, administration
- Rewrite-based operational semantics: supports formal reasoning/policy analysis
- Graph-based policy representation: facilitates implementation/policy visualisation

Obligations

Image designed by Freepik

Obligations and Access Control

- Licence agreements
- EU GDPR - Data collection
- US Gramm-Leach-Bliley Act for financial institutions
- Medical policies: access to treatment/consent form

Features of Obligations

- Mandatory action
- Within an interval, defined by temporal constraints or events
- Atomic or compound actions
- May depend on conditions
- Interactions between obligations and permissions: fulfilling the obligations may depend on certain permissions.
- Accountability, if obligations go unfulfilled.

Events - Types, History, Interval

Event: an *action* that happened at a specific moment in time.

Event Type/Instance = Generic Event/Event

Examples:

$$\begin{aligned} alarmON &= \{\text{act} = fire\_alarmON, \text{ward} = neurology, \\ &\quad \text{happens} = 20220621.120000\}, \\ callFireDep &= \{\text{act} = call\_FireDEP, \text{ward} = neurology, \\ &\quad \text{happens} = 20220621.120500\}, \end{aligned}$$

Generic events include variables:

$$alarmON[W, T] = \{\text{act} = fire\_alarmON, \text{ward} = W, \text{happens} = T\}$$

Event history: ordered sequence h of events that happen in a run of the system

Obligations

A *generic obligation* is a tuple (a, r, ge_1, ge_2, s)
a action, r object, ge_1, ge_2 generic events (interval where the obligation must be fulfilled), $s = (op, sec)$ secondary obligations.

Example:
$(alert, firedept, alarmON[W, T], alarmOFF[W, T'], (\wedge, [o_{call}, o_{notify}]))$,

$o_{notify} = (notify, headTeam, alarmON[T, W], T + 3, id)$
$o_{call} = (call, firedept, alarmON[T, W], alarmOFF[W, T'], id)$

Generic vs Concrete obligation

Duties

A *duty* is a tuple (p, o) of a principal and a concrete obligation.

Duty status:

- *invalid*;
- *fulfilled*;
- *violated*;
- *pending*;

Obligations in the CBAC Metamodel

More entities:

- Countable sets $\mathcal{E}v$ and $\mathcal{GE}v$ of specific events and generic events, respectively: $e, e_1, \ldots$; $ge, ge_1, \ldots$
- Countable set $\mathcal{H}$ of event histories, $h, h_1, \ldots$.
- Countable sets $\mathcal{O}$ of obligations, $o, o_1, \ldots$, and $\mathcal{S}$ of subordinate pairs, $s, s_1, \ldots$.
  The elements of $\mathcal{S}$ are either *id* or pairs of the form (op, sec) where *op* is an operator and *sec* is a list of obligations

More relations

Obligation-category assignment:
$\mathcal{OCA} \subseteq \mathcal{O} \times \mathcal{C}$: $(o, c) \in \mathcal{OCA}$ iff o is assigned to principals in c.

Obligation-principal assignment:
$\mathcal{OPA} \subseteq \mathcal{O} \times \mathcal{P}$: $(o, p) \in \mathcal{OPA}$ iff $p \in \mathcal{P}$ has the obligation o.

Duty assignment:
$\mathcal{DA} \subseteq \mathcal{O}^{\varnothing} \times \mathcal{P} = \mathcal{D}$, such that $(o, p) \in \mathcal{DA}$ iff the principal $p \in \mathcal{P}$ must fulfil the concrete obligation $o = (a, r, e_1, e_2, s)$.

Examples: $o[P, D] = (visa, passport(P), \perp, registration(P,D), id)$
$\mathcal{OCA}$: $(o[P, D], international\text{-}student)$
$\mathcal{PCA}$: $(JohnSmith, international\text{-}student)$
$\mathcal{OPA}$: $(o[JohnSmith, D], JohnSmith)$
$\mathcal{DA}$: $(o[JohnSmith, 20.09.2022], JohnSmith)$

Obligation axioms

$$(o1) \quad \forall o \in \mathcal{O}, \forall p \in \mathcal{P}\big(\big(\exists c, c' \in \mathcal{C},$$
$$(p, c) \in \mathcal{PCA} \wedge \; c \subseteq_o c' \; \wedge (o, c') \in \mathcal{OCA}\big) \Leftrightarrow (o, p) \in \mathcal{OPA}\big)$$

$$(o_2) \quad \forall p \in \mathcal{P}, \forall a \in \mathcal{A}, \forall \in \mathcal{R}, \forall ge_1, ge_2 \in \mathcal{GE}, \forall e_1, e_2 \in \mathcal{E}, \forall s, s_c \in \mathcal{S},$$
$$\big(\big(\exists((a, r, ge_1, ge_2, s), p) \in \mathcal{OPA},$$
$$(e_1, ge_1), (e_2, ge_2) \in \mathcal{EI}, (s_c, s) \in \mathcal{SI}\big)$$
$$\Leftrightarrow ((a, r, e_1, e_2, s_c), p) \in \mathcal{DA}\big)$$

The relations $\mathcal{FULFILLED}$, $\mathcal{PENDING}$ and $\mathcal{VIOLATED}$ are also axiomatised.

Analysis of policies

- models of access control and obligation
- authorisations and obligations co-exist: interdependencies
- dynamic categories: e.g. depending on events in h

Checking interactions: every user has the required permissions in order to fulfill the duties

Strong and Weak Compatibility: Sufficient conditions to ensure that only duties that are consistent with authorisations are issued.

Privacy

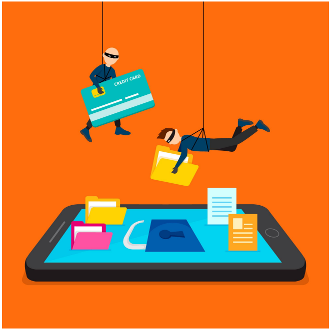


Image designed by Freepik

Web Services, Mobile Apps, Internet of Things . . .

large quantities of data are transmitted to external services

Benefits:
+++ collected data can be used to provide better services to users

Risks:
−−− security and privacy

Goal: Users control which/when data is collected and shared
Supported by regulations: GDPR in EU, FTC in US, etc.
Techniques??
encryption/differential privacy. . . useful but not sufficient

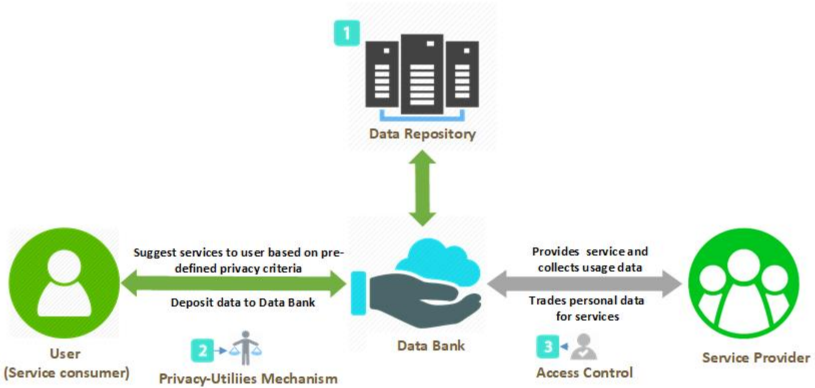
Controlling Data-Collection and Data-Sharing

Two key notions:
- control the way data is collected/transmitted
- control access to data

Requirements:
a cloud-IoT architecture with
integrated data-collection, storage and data management model +
policy languages, enforcement mechanisms, reasoning techniques

Challenges:
variety of IoT devices and services, variety of users, policy specification and enforcement

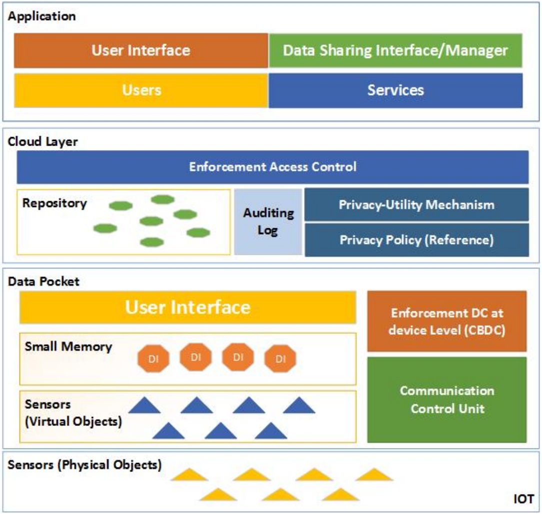
DataBank: A Privacy-Preserving Cloud-IoT Architecture



Main features:

- data repositories: cloud + local Data Pocket
- data collection control before uploading to the cloud
- access control to restrict access to data by third parties
- Implemented by Privasee

DataBank



Application
User Interface
Data Sharing Interface/Manager
Users
Services
Cloud Layer
Enforcement Access Control
Repository
Auditing Log
Privacy-Utility Mechanism
Privacy Policy (Reference)
Data Pocket
User Interface
Enforcement DC at device Level (CBDC)
Small Memory
DI
DI
DI
DI
Communication Control Unit
Sensors (Virtual Objects)
Sensors (Physical Objects)
IOT

Category-Based Data Access Model (CBDA)

Entities:

- $\mathcal{D}$: data sources
- $\mathcal{DI}$: data items e.g. location, time, speed
- $\mathcal{S}$: external services that process data
- $\mathcal{C}$: categories partitioned into
  - $\mathcal{C}_{\mathcal{D}_{\mathcal{U}}}$: unprocessed data
  - $\mathcal{C}_{\mathcal{D}_{\mathcal{S}}}$: stored data for sharing
  - $\mathcal{C}_{\mathcal{S}}$: services
- $\mathcal{A}$: actions, partitioned into
  - $\mathcal{A}_{\mathcal{D}}$: data collection actions e.g., upload, average, encrypt
  - $\mathcal{A}_{\mathcal{S}}$: service actions on stored data, e.g., view, transfer

Categories can be dynamic: defined via attributes

CBDA Model

Relationships:

- *Device-Data Assignment:* $\mathcal{DUA} \subseteq \mathcal{D} \times \mathcal{D_U}$
- *Data Item-Category Assignment:* $\mathcal{DICA} \subseteq \mathcal{DI} \times \mathcal{C}$, partitioned into $\mathcal{DICA}_U$ and $\mathcal{DICA}_S$
- *Action-Category Assignment:* $\mathcal{ACA} \subseteq \mathcal{A} \times \mathcal{C} \times \mathcal{C}$, partitioned $\mathcal{ACA}_D$ (data collection actions) and $\mathcal{ACA}_S$ (service actions):
- *Service-Category Assignment:* $\mathcal{SCA} \subseteq \mathcal{S} \times \mathcal{C}$
- *Authorised Data Collection:* $\mathcal{AD} \subseteq \mathcal{A} \times \mathcal{D_U} \times \mathcal{D_S}$ $(da, ud, sd) \in \mathcal{AD}$ iff the data collection action $da \in \mathcal{A_D}$ is authorised on $ud \in \mathcal{D_U}$ to produce $sd \in \mathcal{D_S}$.
- *Authorised Data Access:* $\mathcal{ADS} \subseteq \mathcal{A} \times \mathcal{D_S} \times \mathcal{S}$, such that $(sa, sd, s) \in \mathcal{ADS}$ iff service action $sa \in \mathcal{A_S}$ is authorised on the stored data item $sd \in \mathcal{D_S}$ for the service $s \in \mathcal{S}$.

CBDA Model

Axioms for authorisations (simplified: no category hierarchies)

Data Collection: unprocessed data → stored data

$$(da1)\quad \forall ud \in \mathcal{D_U},\ \forall sd \in \mathcal{D_S},\ \forall da \in \mathcal{A_D},\ (\exists udc, dsc \in \mathcal{C}, (ud, udc) \in \mathcal{DICA}_U \wedge (da, udc, dsc) \in \mathcal{ACA}_D \wedge (da, ud, sd) \in \mathcal{OP} \wedge (sd, dsc) \in \mathcal{DICA}_S) \Leftrightarrow (da, ud, sd) \in \mathcal{AD}$$

Data Access: stored data → services

$$(da5)\quad \forall sd \in \mathcal{D_S},\ \forall sa \in \mathcal{A_S},\ \forall s \in \mathcal{S},\ (\exists dsc, sc \in \mathcal{C}, (sd, dsc) \in \mathcal{DICA}_S \wedge (s, sc) \in \mathcal{SCA} \wedge (sa, dsc, sc) \in \mathcal{ACA}_S) \Leftrightarrow (sa, sd, s) \in \mathcal{ADS}$$

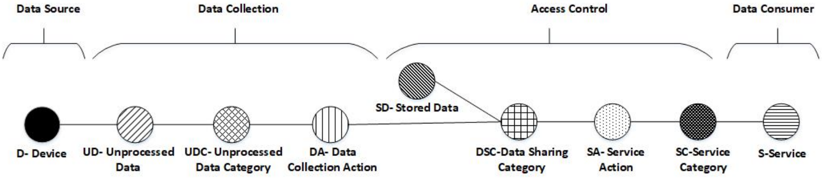
Graph-Based CBDA Policies

CBDA *policy graphs*:

- nodes represent policy *entities*,
- edges represent *relations* defining how entities are categorised and authorised/prohibited actions for each category of entities.

Graph elements are labelled

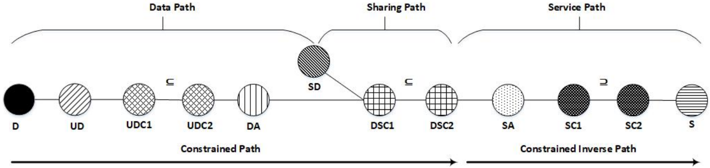
Types of nodes in a CBDA graph:



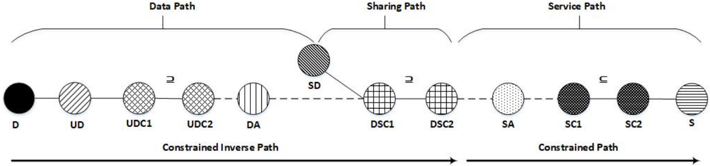
Graph-Based CBDA Policies

Well-typed graphs represent policies.
Paths of specific types define the authorised and prohibited actions for each kind of data item and service.

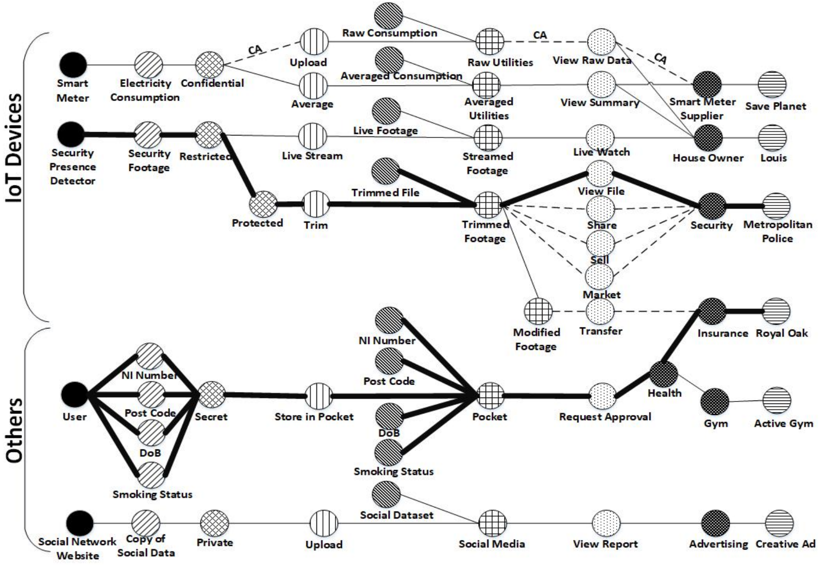
Authorisation Path:



Prohibition Path:



Example CBDA Policy Graph



CBDA Policy Analysis/Queries and Enforcement

Policy queries answered by graph traversal.
Example Policy Content Query:
Are there (permitted or banned) actions assigned to each category?
Answer:
All the categories have some associated action if and only if each node v of type C is in an authorisation or prohibition path.

Example Policy Effect Queries: Absence of conflict (mutually exclusive actions $a1$, $a2$ on di are not permitted).
Answer:
Set of authorisation paths starting in di does not contain paths via a1 and paths via a2.

Enforcement of privacy preferences:
service obligation policy matches CBDA policy

Conclusions - Future work

- Categorisation: powerful abstraction mechanism
- Future work:
  - Policy languages / Usability
  - Policy enforcement / obligations: privacy
  - Negotiation: Risk-Benefit Analysis - optimal policy
  - Policy Mining
  - Policy composition . . .

Conclusions - Future work

- Categorisation: powerful abstraction mechanism
- Future work:
  - Policy languages / Usability
  - Policy enforcement / obligations: privacy
  - Negotiation: Risk-Benefit Analysis - optimal policy
  - Policy Mining
  - Policy composition ...

This talk is based on:

- Bertolissi and F. *A metamodel of access control for distributed environments.* Information and Computation 2014
- Alves and F. *A graph-based framework for the analysis of access control policies.* Theoretical Computer Science 2017
- Alves and F. *An Expressive Model for the Specification and Analysis of Obligations.* Preprint 2023
- F., Jaimunk, Thuraisingham. *A Privacy-Preserving Architecture and Data-Sharing Model for Cloud-IoT Applications.* IEEE TDSC 2022