Graph-Based Data-Collection Policies for the Internet of Things

Maribel Fernández,
Jenjira Jaimunk,
Bhavani Thuraisingham

King's College London, University of Texas at Dallas

ICSS, December 2018

Background

Problem:

The data collected can be used to improve services; however, there are serious privacy risks.

+ Addressed by means of privacy policies.

- Difficult to understand the scope and consequences of such policies.

Contributions

Main contribution:

Demonstrating how to specify and visualise the CBDC policies using a graph-based policy language

Suggest how to analyse policy properties by provided queries and show how these queries can be answered

Preliminaries: Category-Based metamodel for Data Collection CBDC

Recall:

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

Entities:

- A countable set $\mathcal{D}ev$ of IoT devices $d_1, d_2, \ldots$: to represent data sources and channels; e.g. individual sensors, aggregators, clocks, etc.
- A countable set $\mathcal{D}i$ of data items $di_1, di_2, \ldots$: to represent data emanating from sensors and also contextual information (such as location, time, identifier, etc.)
- A set $\mathcal{A}$ of actions: e.g., send, receive, block, encrypt, decrypt, etc.
- A countable set $\mathcal{C}$ of categories c_0, $c_1, \ldots$
- A countable set $\mathcal{S}$ of services: to represent actual services or users that own/process data.

Preliminaries: Category-Based metamodel for Data Collection CBDC

Axioms:

$(dc1)$ $\forall d \in \mathcal{D}ev,\ \forall a \in \mathcal{A},\ \forall di \in \mathcal{DI}_d,$
$(\exists c, c' \in \mathcal{C}, (di, c) \in \mathcal{DICA}_d \wedge c \subseteq c' \wedge (a, c') \in \mathcal{ACA})$
$\Leftrightarrow (a, di) \in \mathcal{ADI}_d$

$(dc2)$ $\forall d \in \mathcal{D}ev,\ \forall a \in \mathcal{A},\ \forall di \in \mathcal{DI}_d,$
$(\exists c, c' \in \mathcal{C}, (di, c) \in \mathcal{DICA}_d \wedge c' \subseteq c \wedge (a, c') \in \mathcal{BACA})$
$\Leftrightarrow (a, di) \in \mathcal{BADI}_d$

$(dc3)$ $\forall d \in \mathcal{D}ev,\ \forall a \in \mathcal{A}, \forall di \in \mathcal{DI}_d,$
$((a, di) \notin \mathcal{ADI}_d \ \wedge \ (a, di) \notin \mathcal{BADI}_d)$
$\Leftrightarrow (a, di) \in UNDET$

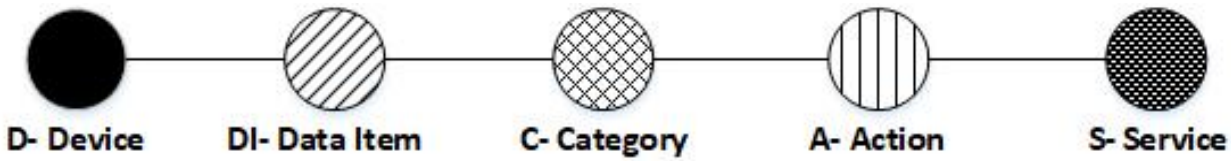
$(dc4)$ $\forall d \in \mathcal{D}ev,\ \mathcal{ADI}_d \cap \mathcal{BADI}_d = \emptyset$

Policy Graph

A *policy graph*, or *graph* for short, is a tuple $\mathcal{G} = (V, E, lv, le)$, where V is a set of nodes, E is a set of undirected edges, $lv : V \to \mathcal{REC}$ is a labelling function for nodes, such that, for every $v \in V$, $lv(v).\text{ent} \in \mathcal{D} \cup \mathcal{D}i \cup \mathcal{C} \cup \mathcal{A} \cup \mathcal{S}$, and $le : E \to \mathcal{REC}$ is a labelling function for edges, such that, for every $e \in E$ between nodes v_1 and v_2, $le(e).\text{adj} = \{v_1, v_2\}$, where $v_1, v_2 \in V$ and $v_1 \neq v_2$. In addition, we assume that the record labels of nodes contain a field $\text{type} = T$, where $T \in \{D, Di, C, A, S\}$, such that $lv(v).\text{type} = D$ if $lv(v).\text{ent} = d \in \mathcal{D}$ (that is, D is the type of the nodes representing devices), and similarly Di represents data items, C categories, S services and A actions. The type of an edge is determined by the type of its adjacent nodes, that is, if $le(e).\text{adj} = \{v_1, v_2\}$, then $\text{type}(e) = (lv(v_1).\text{type}, lv(v_2).\text{type})$.

Policy Graph [con.]

Figure below shows a path in a CBDC policy with five entities of types D (device), Di (data items), C (category), A (action) and S (service).



Relations in a policy graph with prohibitions

- $DDIA_{\mathcal{G}} = \{(lv(v_1).\text{ent}, lv(v_2).\text{ent}) \mid \text{type}(v_1, v_2) = DDi\}$.
- $DDICA_{\mathcal{G}} = \{(lv(v_1).\text{ent}, lv(v_2).\text{ent}, lv(v_3).\text{ent}) \mid \text{type}(v_1, v_2, v_3) = DDi, DiC\}$.
- $ASCA_{\mathcal{G}} = \{(lv(v_1).\text{ent}, lv(v_2).\text{ent}, lv(v_3).\text{ent}) \mid \text{type}(v_3, v_1, v_2) = CA^A, AS\}$.
- $BASCA_{\mathcal{G}} = \{(lv(v_1).\text{ent}, lv(v_2).\text{ent}, lv(v_3).\text{ent}) \mid \text{type}(v_3, v_1, v_2) = CA^B, AS\}$.

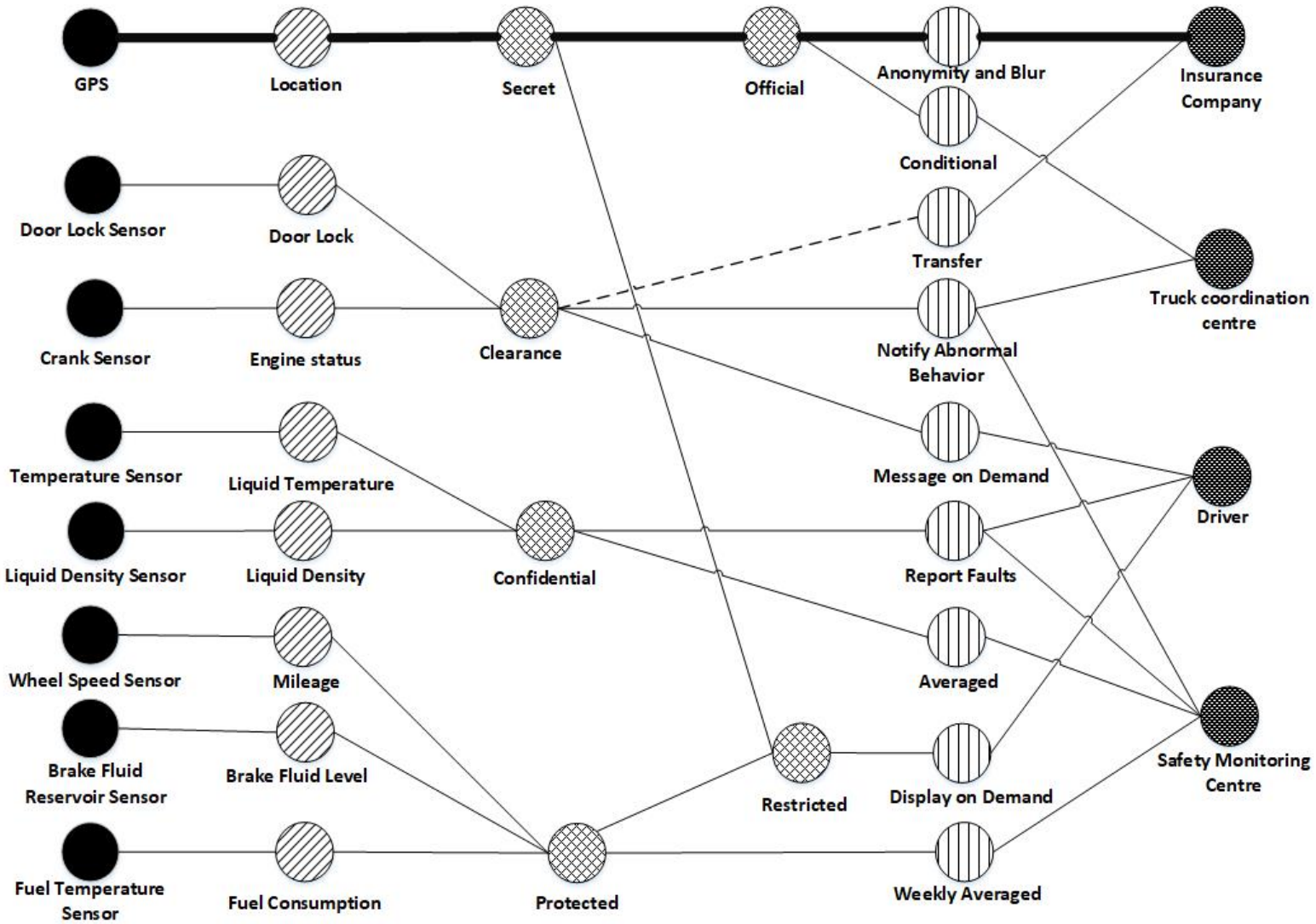
Relations in a policy graph with prohibitions[con.]

- $ASDID_{\mathcal{G}} = \{(lv(v_1).\text{ent}, lv(v_2).\text{ent}, lv(v_4).\text{ent}, lv(v_5).\text{ent}) \mid \exists v_{31}, \ldots, v_{3n}\ s.t.\ \text{type}(v_5, v_4, v_{31}, \ldots, v_{3n}, v_1, v_2) = DDi, DiC, (\overrightarrow{CC})^*, CA^A, AS\}$.
- $BASDID_{\mathcal{G}} = \{(lv(v_1).\text{ent}, lv(v_2).\text{ent}, lv(v_4).\text{ent}, lv(v_5).\text{ent}) \mid \exists v_{31}, \ldots, v_{3n}\ s.t.\ \text{type}(v_5, v_4, v_{31}, \ldots, v_{3n}, v_1, v_2) = DDi, DiC, (\overleftarrow{CC})^*, CA^B, AS\}$.
- $UNDET_{\mathcal{G}} = \{(lv(v_1).\text{ent}, lv(v_2).\text{ent}, lv(v_3).\text{ent}, lv(v_4).\text{ent}) \mid lv(v_1).\text{type} = D, lv(v_2).\text{type} = DI, lv(v_3).\text{type} = A, lv(v_4).\text{type} = S\} - (ASDID_{\mathcal{G}} \cup BASDID_{\mathcal{G}})$.

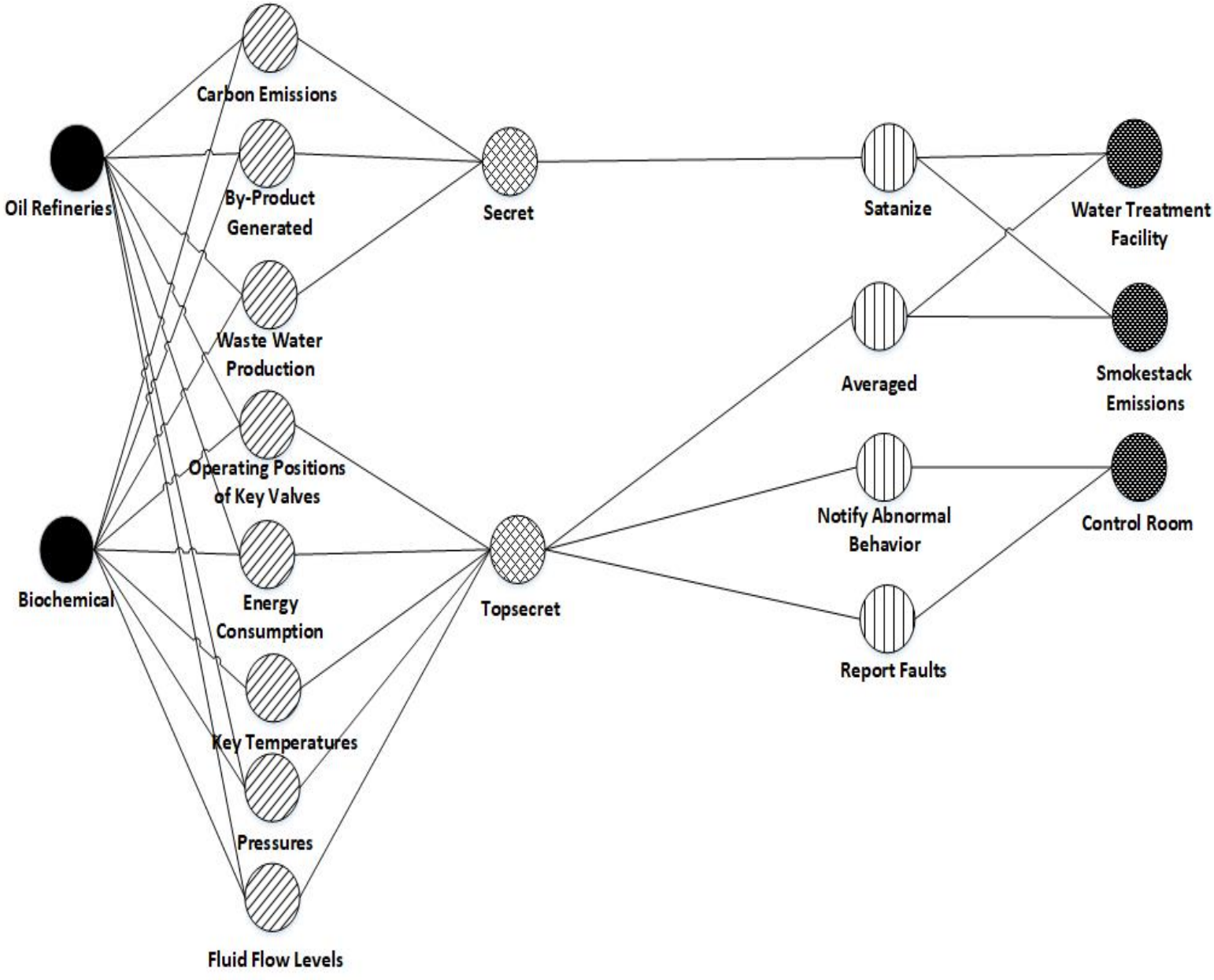
Policy

$CBDC_{\mathcal{G}}$ is an abbreviation for the tuple
$\langle \mathcal{E}, DDIA_{\mathcal{G}}, DDICA_{\mathcal{G}}, ASCA_{\mathcal{G}}, ASDID_{\mathcal{G}}, BASCA_{\mathcal{G}}, BASDID_{\mathcal{G}},$
$UNDET_{\mathcal{G}} \rangle$

Example: Policy and path for a tanker tracking system in a chemical plant



Example: Chemical Plant Policy



CBDC Policy Analysis and Queries

Policy content queries:

The objective of the policy content queries is to analyse policies by obtaining policy information at the same time as the query is established.

Here are some examples:

Are there (permitted or prohibited) actions assigned to each category?

CBDC Policy Analysis and Queries [con.]

This query can be answered by graph-theoretic methods when using graph policies, as follows:

All the categories have some associated (permitted or prohibited) actions if and only if for each node v of type C there is a path of type $(\overrightarrow{CC})^*, CA^A, AS$ or a path of type $(\overleftarrow{CC})^*, CA^B, AS$ starting in v.

CBDC Policy Analysis and Queries

Policy effect queries: Totality

A CBDC policy is total if, for every device *d* generating data items *di*, the policy specifies all the actions and services that are permitted on *di* and all the actions and services that are not permitted on *di*.

CBDC Policy Analysis and Queries

Policy effect queries: Totality [con.]

Assuming there is a well-formed policy graph $\mathcal{G}$ representing the CBDC policy, totality can be verified by computing the relations $ASDID_{\mathcal{G}}$ and $BASDID_{\mathcal{G}}$ and checking that $ASDID_{\mathcal{G}} \cup BASDID_{\mathcal{G}}$ contains all the relevant tuples from $\mathcal{A} \times \mathcal{S} \times \mathcal{D}i \times \mathcal{D}$.

A CBDC policy defined by a policy graph $\mathcal{G}$ is total if and only if for all $(di, d) \in \mathcal{D}i \times \mathcal{D}$, if $(a, s) \in \mathcal{A} \times \mathcal{S}$ applies to (di, d) then $(a, s, di, d) \in ASDID_{\mathcal{G}} \cup BASDID_{\mathcal{G}}$.

CBDC Policy Analysis and Queries

Policy effect queries: Consistency

A CBDC policy is consistent if, for any given data item, all relevant actions are either permitted or prohibited, but not both.

Consistency is enforced by axiom ($dc4$), that is, the policy graph satisfies $ASDID_{\mathcal{G}} \cap BASDID_{\mathcal{G}} = \emptyset$, which means that data collection policies defined by well formed graphs are consistent by construction.

CBDC Policy Analysis and Queries

Policy effect queries: Absence of Conflicts

The absence of rules that permit two mutually exclusive actions on a data item. If an action a_1 performed by a service s_1 is in conflict with an action a_2 performed by s_2 then the policy should ensure that if a_1 is authorised then a_2 is forbidden and viceversa.

CBDC Policy Analysis and Queries

Policy effect queries: Absence of Conflicts

Let $\mathcal{G}$ be a well-formed policy graph. Assume an action a_1 performed by a service s_1 is in conflict with an action a_2 by s_2. The policy graph $\mathcal{G}$ ensures absence of conflict between a_1, s_1 and a_2, s_2 if for each kind of data item di generated by a device d, in the paths of type $DDI, DIC, (\overrightarrow{CC})^*, CA, AS$ linking the nodes representing d and the nodes representing s_1 and s_2, the set of values for the attribute ent in labels of nodes of type A in each path do not contain a_1 at the same time as a_2.

Future work

- Policy combinations
- IoT-cloud architecture for performance analysis on larger scale applications.