Capacity: an Abstract Model of Control over Personal Data

Daniel LE MÉTAYER and Pablo RAUZY

planete.inrialpes.fr/people/lemetayer
daniel.le-metayer@inria.fr

pablo.rauzy.name
pablo.rauzy@univ-paris8.fr

2019-03-18 @ CNRS, Paris
Journée du GT Méthodes Formelles pour la Sécurité 2019

OA version of the paper: hal-01638190

- Control over personnal data
- Modeling control
- Characterizing control
- Evaluating concrete systems

- The notion of *privacy by control* is predominant in the privacy literature.
- However, it lacks a formal definition.
- This makes it hard to check for compliance, to compare design options, etc.

→ We want a formal framework to specify the notion of *control over personal data*.

Control

- Formally capturing the notion of control is notoriously difficult.
- Control is about a potential rather than one particular realization.
- Existing control literature (e.g., *access control* and *usage control*) does not really encapsulates the intuition underlying the notion of control over personal data.

Three dimensions of control

- In their 2015 paper*, Lazaro and Le Métayer identified three dimensions of control over personal data.
- These three dimensions corresponds to the capacities for an individual:
  - to *perform* actions on their personal data,
  - to *prevent* others from performing actions on their personal data, and
  - to *be informed* of actions performed by others on their personal data.

→ Based on this work, we built *Capacity*.

* `http://script-ed.org/?p=1927`

- *Capacity*'s goal is to model control over personal data in a very general way.
- Thus, guiding principles of its design are abstraction and minimality.
- Basically, *agents* can perform *operations* on *resources* in given *contexts*.
- Control is modeled by *requirements* expressing constraints on those operations.

→ Running example for this: rudimentary photo sharing service.

Running example: *Album*

- This talk uses a simple photo sharing service, named *Album*, as an example.
- *Album* is a centralized service where:
  - users can upload, delete, and access photos in their album ;
  - users can connect to each other to become friends ;
  - users can see their friends photos ;
  - users can tag theirs and their friends photos with their name or the names of friends ;
  - users are notified when they are tagged in a photo by someone else.

Objects

- There are four types of atomic objects in *Capacity*:
  - Agents:
    - agents model users and services,
    - the set of agents is $\mathcal{A}$,
    - examples: `Album` (the service) and its users (`Daniel`, `Pablo`, ...) ;
  - Resources:
    - resources model data, and typically personal data,
    - the set of resources is $\mathcal{R}$,
    - examples: usernames (`Pablo`), users' album ($album_{\texttt{Pablo}}$), and photos ().
  - Operations:
    - operations model what can be performed on resources,
    - the set of operations is $\mathcal{O}$,
    - examples: $\mathrm{connect}$, upload, tag, access, delete ;
  - Contexts:
    - contexts model any external factors relevant to an operation,
    - the set of contexts is $\mathcal{C}$,
    - examples: location, time, relationship between agents, purpose, exposure.

Objects

- There are four types of atomic objects in *Capacity*:
  - Agents:
    - agents model users and services,
    - the set of agents is $\mathcal{A}$,
    - examples: `Album` (the service) and its users (`Daniel`, `Pablo`, ...) ;
  - Resources:
    - resources model data, and typically personal data,
    - the set of resources is $\mathcal{R}$,
    - examples: usernames (`Pablo`), users' album ($album_{\texttt{Pablo}}$), and photos ().
  - Operations:
    - operations model what can be performed on resources,
    - the set of operations is $\mathcal{O}$,
    - examples: $\mathrm{connect}$, upload, tag, access, delete ;
  - Contexts:
    - contexts model any external factors relevant to an operation,
    - the set of contexts is $\mathcal{C}$,
    - examples: location, time, relationship between agents, purpose, exposure.

Objects

- There are four types of atomic objects in *Capacity*:
  - Agents:
    - agents model users and services,
    - the set of agents is $\mathcal{A}$,
    - examples: `Album` (the service) and its users (`Daniel`, `Pablo`, ...) ;
  - Resources:
    - resources model data, and typically personal data,
    - the set of resources is $\mathcal{R}$,
    - examples: usernames (`Pablo`), users' album ($album_{\texttt{Pablo}}$), and photos ().
  - Operations:
    - operations model what can be performed on resources,
    - the set of operations is $\mathcal{O}$,
    - examples: $\mathrm{connect}$, upload, tag, access, delete ;
  - Contexts:
    - contexts model any external factors relevant to an operation,
    - the set of contexts is $\mathcal{C}$,
    - examples: location, time, relationship between agents, purpose, exposure.

Objects

- There are four types of atomic objects in *Capacity*:
  - Agents:
    - agents model users and services,
    - the set of agents is $\mathcal{A}$,
    - examples: `Album` (the service) and its users (`Daniel`, `Pablo`, ...) ;
  - Resources:
    - resources model data, and typically personal data,
    - the set of resources is $\mathcal{R}$,
    - examples: usernames (`Pablo`), users' album ($album_{\texttt{Pablo}}$), and photos ().
  - Operations:
    - operations model what can be performed on resources,
    - the set of operations is $\mathcal{O}$,
    - examples: $\mathrm{connect}$, upload, tag, access, delete ;
  - Contexts:
    - contexts model any external factors relevant to an operation,
    - the set of contexts is $\mathcal{C}$,
    - examples: location, time, relationship between agents, purpose, exposure.

Objects

- There are four types of atomic objects in *Capacity*:
  - Agents:
    - agents model users and services,
    - the set of agents is $\mathcal{A}$,
    - examples: `Album` (the service) and its users (`Daniel`, `Pablo`, ...) ;
  - Resources:
    - resources model data, and typically personal data,
    - the set of resources is $\mathcal{R}$,
    - examples: usernames (`Pablo`), users' album ($album_{\texttt{Pablo}}$), and photos ().
  - Operations:
    - operations model what can be performed on resources,
    - the set of operations is $\mathcal{O}$,
    - examples: $\mathrm{connect}$, upload, tag, access, delete ;
  - Contexts:
    - contexts model any external factors relevant to an operation,
    - the set of contexts is $\mathcal{C}$,
    - examples: location, time, relationship between agents, purpose, exposure.

Actions

- *Actions* model the application of an operation to a list of parameters in a context.
  - Action $\mathrm{op}_c(x_1, \ldots, x_n)$ is the application of operation op to $x_1, \ldots, x_n$ in context c.
  - Parameters x_i can be resources or agents.
- Examples:
  - $\mathrm{connect}_c(\texttt{Daniel})$,
  - $\mathrm{upload}_c($$, \ album_{\texttt{Pablo}})$,
  - $\mathrm{tag}_c($$, \texttt{Daniel})$.
- The set of actions is Δ.

Relations

- We define three *relations* on atomic objects:
  - $Pers(r, a)$ expresses that resource r is a personal data of agent a,
  - $In(r, \alpha)$ expresses that resource r is involved in action α,
  - $Trust(a, b)$ expresses that agent a trusts agent b.
- Examples:
  - $Pers($, $\texttt{Pablo})$,
  - $In($, $\text{tag}_c($, $\texttt{Pablo}))$,
  - $Trust(\texttt{Pablo}, \texttt{Daniel})$.

Requirements

- A *requirement* R is a relation $Can^R \subseteq \mathcal{A} \times \Delta \times \mathcal{P}(\mathcal{A}) \times \mathcal{P}(\mathcal{A})$.
- Intuitively, $Can^R(a, \alpha, E, W)$ means that:
  - agent a can perform action α
  - only if this action is enabled by all agents in E
  - while all agents in W have to to be informed of it.
- Examples:
  - $Can^R(\texttt{Pablo}, \mathrm{upload}_c($$, album_{\texttt{Pablo}}), \{\texttt{Album}\}, \{\texttt{Album}\})$,
  - $Can^R(\texttt{Daniel}, \mathrm{upload}_c($$, album_{\texttt{Pablo}}), \{\bot\}, \{\bot\})$,
  - $Can^R(\texttt{Pablo}, \mathrm{tag}_c($$, \texttt{Daniel}), \{\texttt{Daniel}, \texttt{Album}\}, \{\texttt{Daniel}, \texttt{Album}\})$.
- This single relation can express the three capacities of control of personal data:
  - when $x = a$ it expresses the capacity of x to *perform* action α,
  - when $x \in E$ it expresses the capacity of x to *prevent* action α,
  - when $x \in W$ it expresses the capacity of x to *be informed* of action α.

- A *requirement* R is a relation $Can^R \subseteq \mathcal{A} \times \Delta \times \mathcal{P}(\mathcal{A}) \times \mathcal{P}(\mathcal{A})$.
- Intuitively, $Can^R(a, \alpha, E, W)$ means that:
  - agent a can perform action α
  - only if this action is enabled by all agents in E
  - while all agents in W have to to be informed of it.
- Examples:
  - $Can^R(\texttt{Pablo}, \mathrm{upload}_c($  $, album_{\texttt{Pablo}}), \{\texttt{Album}\}, \{\texttt{Album}\})$,
  - $Can^R(\texttt{Daniel}, \mathrm{upload}_c($  $, album_{\texttt{Pablo}}), \{\bot\}, \{\bot\})$,
  - $Can^R(\texttt{Pablo}, \mathrm{tag}_c($  $, \texttt{Daniel}), \{\texttt{Daniel}, \texttt{Album}\}, \{\texttt{Daniel}, \texttt{Album}\})$.
- This single relation can express the three capacities of control of personal data:
  - when $x = a$ it expresses the capacity of x to *perform* action α,
  - when $x \in E$ it expresses the capacity of x to *prevent* action α,
  - when $x \in W$ it expresses the capacity of x to *be informed* of action α.

- Requirements semantics is given by characterizing execution traces.
- Traces are characterized using four *abstract properties*:
  - $\theta \vdash \mathit{Requests}(a, \alpha)$:
    - in trace θ, agent a attempts to perform action α,
    - example: $\theta \vdash \mathit{Requests}(\texttt{Pablo}, \mathrm{tag}_c($  $, \texttt{Daniel}))$;
  - $\theta \vdash \mathit{Enables}(a, b, \alpha)$:
    - in trace θ, agent a enables the performance of action α by agent b,
    - example: $\theta \vdash \mathit{Enables}(\texttt{Album}, \texttt{Pablo}, \mathrm{tag}_c($  $, \texttt{Daniel}))$,
      $\theta \vdash \mathit{Enables}(\texttt{Daniel}, \texttt{Pablo}, \mathrm{tag}_c($  $, \texttt{Daniel}))$;
  - $\theta \vdash \mathit{Does}(a, b, \alpha)$:
    - in trace θ, agent a performs action α on behalf of agent b,
    - example: $\theta \vdash \mathit{Does}(\texttt{Album}, \texttt{Pablo}, \mathrm{tag}_c($  $, \texttt{Daniel}))$;
  - $\theta \vdash \mathit{Notifies}(a, b, c, \alpha)$:
    - in trace θ, agent a notifies to agent b the performance of action α on behalf of agent c,
    - example: $\theta \vdash \mathit{Notifies}(\texttt{Album}, \texttt{Daniel}, \texttt{Pablo}, \mathrm{tag}_c($  $, \texttt{Daniel}))$.

Abstract trace properties

- Requirements semantics is given by characterizing execution traces.
- Traces are characterized using four *abstract properties*:
  - $\theta \vdash \mathit{Requests}(a, \alpha)$:
    - in trace θ, agent a attempts to perform action α,
    - example: $\theta \vdash \mathit{Requests}(\texttt{Pablo}, \mathrm{tag}_c(\,\cdot\,, \texttt{Daniel}))$;
  - $\theta \vdash \mathit{Enables}(a, b, \alpha)$:
    - in trace θ, agent a enables the performance of action α by agent b,
    - example: $\theta \vdash \mathit{Enables}(\texttt{Album}, \texttt{Pablo}, \mathrm{tag}_c(\,\cdot\,, \texttt{Daniel}))$,
      $\theta \vdash \mathit{Enables}(\texttt{Daniel}, \texttt{Pablo}, \mathrm{tag}_c(\,\cdot\,, \texttt{Daniel}))$;
  - $\theta \vdash \mathit{Does}(a, b, \alpha)$:
    - in trace θ, agent a performs action α on behalf of agent b,
    - example: $\theta \vdash \mathit{Does}(\texttt{Album}, \texttt{Pablo}, \mathrm{tag}_c(\,\cdot\,, \texttt{Daniel}))$;
  - $\theta \vdash \mathit{Notifies}(a, b, c, \alpha)$:
    - in trace θ, agent a notifies to agent b the performance of action α on behalf of agent c,
    - example: $\theta \vdash \mathit{Notifies}(\texttt{Album}, \texttt{Daniel}, \texttt{Pablo}, \mathrm{tag}_c(\,\cdot\,, \texttt{Daniel}))$.

- Requirements semantics is given by characterizing execution traces.
- Traces are characterized using four *abstract properties*:
  - $\theta \vdash \mathit{Requests}(a, \alpha)$:
    - in trace θ, agent a attempts to perform action α,
    - example: $\theta \vdash \mathit{Requests}(\texttt{Pablo}, \mathrm{tag}_c($$, \texttt{Daniel}))$;
  - $\theta \vdash \mathit{Enables}(a, b, \alpha)$:
    - in trace θ, agent a enables the performance of action α by agent b,
    - example: $\theta \vdash \mathit{Enables}(\texttt{Album}, \texttt{Pablo}, \mathrm{tag}_c($$, \texttt{Daniel}))$,
      $\theta \vdash \mathit{Enables}(\texttt{Daniel}, \texttt{Pablo}, \mathrm{tag}_c($$, \texttt{Daniel}))$;
  - $\theta \vdash \mathit{Does}(a, b, \alpha)$:
    - in trace θ, agent a performs action α on behalf of agent b,
    - example: $\theta \vdash \mathit{Does}(\texttt{Album}, \texttt{Pablo}, \mathrm{tag}_c($$, \texttt{Daniel}))$;
  - $\theta \vdash \mathit{Notifies}(a, b, c, \alpha)$:
    - in trace θ, agent a notifies to agent b the performance of action α on behalf of agent c,
    - example: $\theta \vdash \mathit{Notifies}(\texttt{Album}, \texttt{Daniel}, \texttt{Pablo}, \mathrm{tag}_c($$, \texttt{Daniel}))$.

Abstract trace properties

- Requirements semantics is given by characterizing execution traces.
- Traces are characterized using four *abstract properties*:
  - $\theta \vdash \mathit{Requests}(a, \alpha)$:
    - in trace θ, agent a attempts to perform action α,
    - example: $\theta \vdash \mathit{Requests}(\texttt{Pablo}, \mathrm{tag}_c(\;, \texttt{Daniel}))$;
  - $\theta \vdash \mathit{Enables}(a, b, \alpha)$:
    - in trace θ, agent a enables the performance of action α by agent b,
    - example: $\theta \vdash \mathit{Enables}(\texttt{Album}, \texttt{Pablo}, \mathrm{tag}_c(\;, \texttt{Daniel}))$,
      $\theta \vdash \mathit{Enables}(\texttt{Daniel}, \texttt{Pablo}, \mathrm{tag}_c(\;, \texttt{Daniel}))$;
  - $\theta \vdash \mathit{Does}(a, b, \alpha)$:
    - in trace θ, agent a performs action α on behalf of agent b,
    - example: $\theta \vdash \mathit{Does}(\texttt{Album}, \texttt{Pablo}, \mathrm{tag}_c(\;, \texttt{Daniel}))$;
  - $\theta \vdash \mathit{Notifies}(a, b, c, \alpha)$:
    - in trace θ, agent a notifies to agent b the performance of action α on behalf of agent c,
    - example: $\theta \vdash \mathit{Notifies}(\texttt{Album}, \texttt{Daniel}, \texttt{Pablo}, \mathrm{tag}_c(\;, \texttt{Daniel}))$.

- Requirements semantics is given by characterizing execution traces.
- Traces are characterized using four *abstract properties*:
  - $\theta \vdash$ *Requests*(a, α):
    - in trace θ, agent a attempts to perform action α,
    - example: $\theta \vdash$ *Requests*$(\texttt{Pablo}, \mathrm{tag}_c(\ , \texttt{Daniel}))$;
  - $\theta \vdash$ *Enables*(a, b, α):
    - in trace θ, agent a enables the performance of action α by agent b,
    - example: $\theta \vdash$ *Enables*$(\texttt{Album}, \texttt{Pablo}, \mathrm{tag}_c(\ , \texttt{Daniel}))$,
      $\theta \vdash$ *Enables*$(\texttt{Daniel}, \texttt{Pablo}, \mathrm{tag}_c(\ , \texttt{Daniel}))$;
  - $\theta \vdash$ *Does*(a, b, α):
    - in trace θ, agent a performs action α on behalf of agent b,
    - example: $\theta \vdash$ *Does*$(\texttt{Album}, \texttt{Pablo}, \mathrm{tag}_c(\ , \texttt{Daniel}))$;
  - $\theta \vdash$ *Notifies*(a, b, c, α):
    - in trace θ, agent a notifies to agent b the performance of action α on behalf of agent c,
    - example: $\theta \vdash$ *Notifies*$(\texttt{Album}, \texttt{Daniel}, \texttt{Pablo}, \mathrm{tag}_c(\ , \texttt{Daniel}))$.

Trace consistency

- A trace θ is *consistent* if:
  - $\theta \vdash \mathit{Does}(c, a, \alpha) \implies \theta \vdash \mathit{Requests}(a, \alpha)$,
  - $\theta \vdash \mathit{Notifies}(a, b, c, \alpha) \implies \exists d, \theta \vdash \mathit{Does}(d, c, \alpha)$.
- Intuitively, a trace is inconsistent if it includes:
  - an action performed on behalf of an agent that has not requested it, or
  - the notification of an action that has not been performed.

Trace completeness

- A trace θ is *complete* wrt requirement R where $Can^R(a, \alpha, E, W)$ if:
  - $\theta \vdash \mathit{Requests}(a, \alpha) \land \forall b \in E, \theta \vdash \mathit{Enables}(b, a, \alpha) \implies \exists c \in \mathcal{A}, \theta \vdash \mathit{Does}(c, a, \alpha)$.
- Intuitively, a trace is complete if an action is always performed when:
  - it has been requested, and
  - it has been enabled by all necessary agents.

Trace compliance

- A trace θ is *compliant* with requirement R where $Can^R(a, \alpha, E, W)$ if:
  - $\forall d \in \mathcal{A}, \theta \vdash \mathit{Does}(d, a, \alpha) \implies \forall b \in E, \theta \vdash \mathit{Enables}(b, a, \alpha)$,
  - $\forall d \in \mathcal{A}, \theta \vdash \mathit{Does}(d, a, \alpha) \implies \forall b \in W, \exists c \in \mathcal{A}, \theta \vdash \mathit{Notifies}(c, b, a, \alpha)$.
- Intuitively, a trace is compliant if all Can^R constraint are met:
  - no action is performed unless it is enabled by all its enablers, and
  - all agents that have to be informed are notified.
- Compliance is noted $\theta \models R$.

- We introduce four independent *types of control*:
  - action control,
  - observability control,
  - authorization control,
  - notification control.
- Each type comes with three levels of control:
  - absolute control,
  - relative control,
  - lack of control.

Action control

- *Action control* describes an agent's control on actions that it initiates.
- With regard to a requirement R, an agent a has:
  - *absolute action control* over α if it does not depend on others to perform it:
    - $AA_R(a, \alpha) \iff Can^R(a, \alpha, \varnothing, W)$;
  - *relative action control* over α if it depends only trusted agents:
    - $RA_R(a, \alpha) \iff Can^R(a, \alpha, E, W) \wedge b \in E \implies Trust(a, b)$.
- Examples:
  - $Trust(\texttt{Pablo}, \texttt{Album}) \implies RA_R(\texttt{Pablo}, \text{upload}_c($$, album_{\texttt{Pablo}}))$,
  - $Trust(\texttt{Pablo}, \texttt{Album}) \implies RA_R(\texttt{Pablo}, \text{delete}_c($$, album_{\texttt{Pablo}}))$.

- *Observability control* describes an agent's capacity to perform actions that are not observable by others.
- With regard to a requirement R, an agent a has:
  - *absolute observability control* over α if it can perform α discreetly:
    - $AO_R(a, \alpha) \iff Can^R(a, \alpha, E, \varnothing)$;
  - *relative observability control* over α if only trusted agents can know about it:
    - $RO_R(a, \alpha) \iff Can^R(a, \alpha, E, W) \land b \in W \implies Trust(a, b)$.
- Examples:
  - $Trust(\texttt{Pablo}, \texttt{Album}) \implies RO_R(\texttt{Pablo}, \text{upload}_c($  $, album_{\texttt{Pablo}}))$,
  - $Trust(\texttt{Pablo}, \texttt{Album}) \land Trust(\texttt{Pablo}, \texttt{Daniel}) \implies RO_R(\texttt{Pablo}, \text{tag}_c($  $, \texttt{Daniel}))$.

- *Authorization control* describes an agent's control on actions initiated by others.
- With regard to a requirement R, an agent a has:
  - *absolute authorization control* over α if it is the only agent that can prevent it:
    - $AH_R(a, \alpha) \iff Can^R(b, \alpha, \{a\}, W)$;
  - *relative authorization control* over α if it is not the only agent having this capacity:
    - $RH_R(a, \alpha) \iff Can^R(b, \alpha, E, W) \implies a \in E$.
- Examples:
  - $AH_R(\texttt{Album}, \mathrm{upload}_c($$, \mathit{album}_{\texttt{Pablo}}))$,
  - $RH_R(\texttt{Daniel}, \mathrm{tag}_c($$, \texttt{Daniel}))$.

- *Notification control* describes an agent's capacity to be informed about actions performed by others.
- With regard to a requirement R, an agent a has:
  - *absolute notification control* over α if it is the only agent that has the capacity to be informed of it:
    - $AN_R(a, \alpha) \iff Can^R(b, \alpha, E, \{a\})$;
  - *relative notification control* over α if it is not the only agent having this capacity:
    - $RN_R(a, \alpha) \iff Can^R(b, \alpha, E, W) \implies a \in W$.
- Examples:
  - $AN_R(\texttt{Album}, \mathrm{upload}_c($$, album_{\texttt{Pablo}}))$,
  - $RN_R(\texttt{Daniel}, \mathrm{tag}_c($$, \texttt{Daniel}))$.

Extensions

- These types of control can be extended to resources and agents:
  - for resources, by generalizing to all actions that involves the resource:
    - e.g., $AA_R(a, r) \iff \forall \alpha \in \Delta, In(r, \alpha) \implies AA_R(a, \alpha)$;
  - for agents, by generalizing to all the personal data of the agent:
    - e.g., $AA_R(a) \iff \forall r \in \mathcal{R}, Pers(r, a) \implies AA_R(a, r)$.
- Control lattice:
  - it is easy to check that absolute control implies relative control ;
  - using the order defined by implication, we have a lattice made of 3^4 forms of control for each action, data, and agent.

- Concrete traces are sequences of concrete events which can be clearly identified:
  - HTTP requests and responses, SQL queries, file manipulations, etc.
- Modeling a concrete system in *Capacity* requires to:
  - identify the sets of agents, resources, actions, and contexts ;
  - define the conditions under which a concrete trace satisfies each abstract trace property.
- Given this model it is possible to:
  - compute the requirement that corresponds to the system,
  - verify if the system satisfies a specific requirement,
  - evaluate the types and levels of control of each agents.

- In *Album*, concrete traces are sequences of the following events:
  - `U-registers(`u`)`: user u creates an account on *Album* ;
  - `U-uploads-pic(`u, p`)`: user u uploads a photo to their album ;
  - `U-requests-album(`u_1, u_2`)`: user u_1 requests u_2's album ;
  - `U-submits-tag(`u_1, p, u_2`)`: user u_1 tags u_2 in photo p ;
  - `U-deletes-pic(`u_1, p`)`: user u_1 deletes photo p from their album ;
  - `U-requests-con(`u_1, u_2`)`: user u_1 requests to connect with u_2 ;
  - `U-accepts-con(`u_1, u_2`)`: user u_1 accepts to connect with u_2 ;
  - `U-rejects-con(`u_1, u_2`)`: user u_1 rejects to connect with u_2 ;
  - `U-disconnects(`u_1, u_2`)`: user u_1 disconnects from u_2 ;
  - `A-creates-account(`u`)`: *Album* creates u's account ;
  - `A-publishes-pic(`p, u`)`: *Album* publishes photo p in u's album ;
  - `A-serves-album(`u_1, u_2`)`: *Album* sends u_1 the album of u_2 ;
  - `A-connects(`u_1, u_2`)`: *Album* connects u_1 and u_2 ;
  - `A-disconnects(`u_1, u_2`)`: *Album* disconnects u_1 and u_2 ;
  - `A-tags-pic(`u_1, p`)`: *Album* tags u_1 in photo p ;
  - `A-notifies-req(`u_1, u_2`)`: *Album* notifies u_2 of u_1's request to connect ;
  - `A-notifies-con(`u_1, u_2`)`: *Album* notifies u_1 and u_2 that they are connected ;
  - `A-notifies-tag(`u_1, p, u_2`)`: *Album* notifies u_1 that they have been tagged in photo p by u_2.

Album: uploading a photo

- Let θ_n be the nth event in the concrete trace.
- We define our abstract properties as follows:
  - $\theta \vdash \mathit{Requests}(u, \mathrm{upload}_n(p, \mathit{album}_u))$
    $\iff \exists m < n, \theta_m = \texttt{U-uploads-pic}(u, p)$.
  - $\theta \vdash \mathit{Enables}(\texttt{Album}, u, \mathrm{upload}_n(p, \mathit{album}_u))$
    $\iff \exists m < n, \theta_m = \texttt{U-registers}(u)$.
  - $\theta \vdash \mathit{Does}(\texttt{Album}, u, \mathrm{upload}_n(p, \mathit{album}_u))$
    $\iff \theta_n = \texttt{A-publishes-pic}(p, u)$.

Control properties

- With these definitions we can prove that $\theta \models R$ such that:
  - $Can^R(u, \mathrm{upload}_n(p, album_u), \{\texttt{Album}\}, \{\texttt{Album}\})$.
- Which in terms of control means that we have:
  - $RA_R(u, \mathrm{upload}_n(p, album_u))$ if $Trust(u, \texttt{Album})$.
  - $RO_R(u, \mathrm{upload}_n(p, album_u))$ if $Trust(u, \texttt{Album})$.
  - $AH_R(\texttt{Album}, \mathrm{upload}_n(p, album_u))$.
  - $AN_R(\texttt{Album}, \mathrm{upload}_n(p, album_u))$.

Album: tagging a friend

- Let θ_n be the nth event in the concrete trace.
- We define our abstract properties as follows:
  - $\theta \vdash \mathit{Requests}(u_1, \mathrm{tag}_n(p, u_2))$
    $\iff \exists m < n, \theta_m = \texttt{U-submits-tag}(u_1, p, u_2)$.
  - $\theta \vdash \mathit{Enables}(u_2, u_1, \mathrm{tag}_n(p, u_2))$
    $\iff \exists m < n, \theta_m = \texttt{U-accepts-con}(u_2, u_1) \vee \theta_m = \texttt{U-accepts-con}(u_1, u_2)$
    $\wedge \nexists k, m < k < n, \theta_k = \texttt{U-disconnects}(u_2, u_1) \vee \theta_m = \texttt{U-disconnects}(u_1, u_2)$.
  - $\theta \vdash \mathit{Enables}(\texttt{Album}, u_1, \mathrm{tag}_n(p, u_2))$
    $\iff \theta_n = \texttt{A-tags-pic}(u_2, p)$.
  - $\theta \vdash \mathit{Does}(\texttt{Album}, u_1, \mathrm{tag}_n(p, u_2))$
    $\iff \theta_n = \texttt{A-tags-pic}(u_2, p)$.
  - $\theta \vdash \mathit{Notifies}(\texttt{Album}, u_2, u_1, \mathrm{tag}_n(p, u_2))$
    $\iff \theta_{n+1} = \texttt{A-notifies-tag}(u_2, p, u_1)$.

Control properties

- With these definitions we can prove that $\theta \models R$ such that:
  - $Can^R(u_1, \mathrm{tag}_n(p, u_2), \{u_2, \texttt{Album}\}, \{u_2, \texttt{Album}\})$.
- Which in terms of control means that we have:
  - $RA_R(u_1, \mathrm{tag}_n(p, u_2))$ if $Trust(u_1, \texttt{Album}) \wedge Trust(u_1, u_2)$.
  - $RO_R(u_1, \mathrm{tag}_n(p, u_2))$ if $Trust(u_1, \texttt{Album}) \wedge Trust(u_1, u_2)$.
  - $RH_R(u, \mathrm{tag}_n(p, u))$.
  - $RN_R(u, \mathrm{tag}_n(p, u))$.
  - $RH_R(\texttt{Album}, \mathrm{tag}_n(p, u))$.
  - $RN_R(\texttt{Album}, \mathrm{tag}_n(p, u))$.

Implementations comparison

- Types and levels of control allow to formally compare different systems.
- Studying alternative implementations of a given specification can be useful for *privacy by design*.

- *Capacity* provides a formal framework to reason about privacy in terms of control.
- The goal of this work is to serve as foundation for new privacy research and tools.
- Future work:
  - find a better way than contexts to formally capture the notion exposure ;
  - make a user-friendly interface to specify requirements ;
  - model control aspects of personal data related laws such as the GDPR ;
  - build model checking tools to automate requirement verification.

Upsilon : Université populaire de sécurité informatique libre et ouverte

- Recherche en sécurité *émancipatrice* :
  - auto-hébergement / décentralisation
  - transfert par logiciels libres
- Enseignement *technocritique* :
  - « code is law »
  - écologie
- Éducation *populaire* :
  - contrôler ses données
  - conférences gesticulées

29 mars @ Paris 8

`https://upsilon.sh/`