

Capacity: an Abstract Model of Control over Personal Data

Pablo Rauzy

pr@up8.edu

pablo.rauzy.name



2019-04-12 @ LIP6

Séminaire ComplexNetworks

Paper (with Daniel Le Métayer): hal-01638190

- ▶ Control over personal 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*.


- ▶ 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 encapsulate the intuition underlying the notion of control over personal data.


- ▶ 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.


- ▶ 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.


- ▶ 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 ($album_{pablo}$), and photos (.
 - Operations:
 - operations model what can be performed on resources,
 - the set of operations is \mathcal{O} ,
 - examples: 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.

- ▶ 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 (`albumpablo`), and photos (.
 - Operations:
 - operations model what can be performed on resources,
 - the set of operations is \mathcal{O} ,
 - examples: `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.

- ▶ 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 ($album_{pablo}$), and photos (.
 - Operations:
 - operations model what can be performed on resources,
 - the set of operations is \mathcal{O} ,
 - examples: 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.

► 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 ($album_{pablo}$), and photos (.
- Operations:
 - operations model what can be performed on resources,
 - the set of operations is \mathcal{O} ,
 - examples: 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.

- ▶ 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 ($album_{pablo}$), and photos (.
 - Operations:
 - operations model what can be performed on resources,
 - the set of operations is \mathcal{O} ,
 - examples: 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* model the application of an operation to a list of parameters in a context.
 - Action $\text{op}_c(x_1, \dots, x_n)$ is the application of operation op to x_1, \dots, x_n in context c .
 - Parameters x_i can be resources or agents.
- ▶ Examples:
 - $\text{connect}_c(\text{Daniel})$,
 - $\text{upload}_c(\text{🇪🇺}, \text{album}_{\text{pablo}})$,
 - $\text{tag}_c(\text{🇪🇺}, \text{Daniel})$.
- ▶ The set of actions is Δ .

- ▶ 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:
 - $Pers(\text{🇪🇸}, \text{Pablo})$,
 - $In(\text{🇪🇸}, \text{tag}_c(\text{🇪🇸}, \text{Pablo}))$,
 - $Trust(\text{Pablo}, \text{Daniel})$.

- ▶ 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 be informed of it.
- ▶ Examples:
 - $Can^R(\text{Pablo}, \text{upload}_c(\text{🇪🇺}, \text{album}_{\text{pablo}}), \{\text{Album}\}, \{\text{Album}\})$,
 - $Can^R(\text{Daniel}, \text{upload}_c(\text{🇪🇺}, \text{album}_{\text{pablo}}), \{\perp\}, \{\perp\})$,
 - $Can^R(\text{Pablo}, \text{tag}_c(\text{🇪🇺}, \text{Daniel}), \{\text{Daniel}, \text{Album}\}, \{\text{Daniel}, \text{Album}\})$.
- ▶ 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 be informed of it.
- ▶ Examples:
 - $Can^R(\text{Pablo}, \text{upload}_c(\text{🇪🇺}, \text{album}_{\text{pablo}}), \{\text{Album}\}, \{\text{Album}\})$,
 - $Can^R(\text{Daniel}, \text{upload}_c(\text{🇪🇺}, \text{album}_{\text{pablo}}), \{\perp\}, \{\perp\})$,
 - $Can^R(\text{Pablo}, \text{tag}_c(\text{🇪🇺}, \text{Daniel}), \{\text{Daniel}, \text{Album}\}, \{\text{Daniel}, \text{Album}\})$.
- ▶ 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*:
 - $\theta \vdash \text{Requests}(a, \alpha)$:
 - in trace θ , agent a attempts to perform action α ,
 - example: $\theta \vdash \text{Requests}(\text{Pablo}, \text{tag}_c(\text{🇪🇸}, \text{Daniel}))$;
 - $\theta \vdash \text{Enables}(a, b, \alpha)$:
 - in trace θ , agent a enables the performance of action α by agent b ,
 - example: $\theta \vdash \text{Enables}(\text{Album}, \text{Pablo}, \text{tag}_c(\text{🇪🇸}, \text{Daniel}))$,
 $\theta \vdash \text{Enables}(\text{Daniel}, \text{Pablo}, \text{tag}_c(\text{🇪🇸}, \text{Daniel}))$;
 - $\theta \vdash \text{Does}(a, b, \alpha)$:
 - in trace θ , agent a performs action α on behalf of agent b ,
 - example: $\theta \vdash \text{Does}(\text{Album}, \text{Pablo}, \text{tag}_c(\text{🇪🇸}, \text{Daniel}))$;
 - $\theta \vdash \text{Notifies}(a, b, c, \alpha)$:
 - in trace θ , agent a notifies to agent b the performance of action α on behalf of agent c ,
 - example: $\theta \vdash \text{Notifies}(\text{Album}, \text{Daniel}, \text{Pablo}, \text{tag}_c(\text{🇪🇸}, \text{Daniel}))$.

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

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

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

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

- ▶ A trace θ is *consistent* if:
 - $\theta \vdash \text{Does}(c, a, \alpha) \implies \theta \vdash \text{Requests}(a, \alpha)$,
 - $\theta \vdash \text{Notifies}(a, b, c, \alpha) \implies \exists d, \theta \vdash \text{Does}(d, c, \alpha)$.
- ▶ 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.

- ▶ A trace θ is *complete* wrt requirement R where $Can^R(a, \alpha, E, W)$ if:
 - $\theta \vdash Requests(a, \alpha) \wedge \forall b \in E, \theta \vdash Enables(b, a, \alpha) \implies \exists c \in \mathcal{A}, \theta \vdash Does(c, a, \alpha)$.
- ▶ 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.

- ▶ A trace θ is *compliant* with requirement R where $Can^R(a, \alpha, E, W)$ if:
 - $\forall d \in \mathcal{A}, \theta \vdash Does(d, a, \alpha) \implies \forall b \in E, \theta \vdash Enables(b, a, \alpha),$
 - $\forall d \in \mathcal{A}, \theta \vdash Does(d, a, \alpha) \implies \forall b \in W, \exists c \in \mathcal{A}, \theta \vdash Notifies(c, b, a, \alpha).$
- ▶ 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* 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:
 - $AA_R(a, \alpha) \iff Can^R(a, \alpha, \emptyset, W)$;
 - *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:
 - $Trust(\text{Pablo}, \text{Album}) \implies RA_R(\text{Pablo}, \text{upload}_c(\text{📷}, \text{album}_{\text{pablo}}))$,
 - $Trust(\text{Pablo}, \text{Album}) \implies RA_R(\text{Pablo}, \text{delete}_c(\text{📷}, \text{album}_{\text{pablo}}))$.

- ▶ *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:
 - $AO_R(a, \alpha) \iff Can^R(a, \alpha, E, \emptyset)$;
 - *relative observability control* over α if only trusted agents can know about it:
 - $RO_R(a, \alpha) \iff Can^R(a, \alpha, E, W) \wedge b \in W \implies Trust(a, b)$.
- ▶ Examples:
 - $Trust(\text{Pablo}, \text{Album}) \implies RO_R(\text{Pablo}, \text{upload}_c(\text{🇪🇺}, \text{album}_{\text{pablo}}))$,
 - $Trust(\text{Pablo}, \text{Album}) \wedge Trust(\text{Pablo}, \text{Daniel}) \implies RO_R(\text{Pablo}, \text{tag}_c(\text{🇪🇺}, \text{Daniel}))$.

- ▶ *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:
 - $AH_R(\text{Album}, \text{upload}_c(\text{🇪🇸}, \text{album}_{\text{Pablo}}))$,
 - $RH_R(\text{Daniel}, \text{tag}_c(\text{🇪🇸}, \text{Daniel}))$.

- ▶ *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:
 - $AN_R(\text{Album}, \text{upload}_c(\text{🇪🇸}, \text{album}_{\text{Pablo}}))$,
 - $RN_R(\text{Daniel}, \text{tag}_c(\text{🇪🇸}, \text{Daniel}))$.

- ▶ 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 .

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

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

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

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

- ▶ 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 (Tupac):
 - 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 tools to automate requirement verification.

Control over Personal Data

- Control
- Three dimensions of control

Modeling Control with *Capacity*

- Running example: *Album*
- Objects
- Actions
- Relations
- Requirements
- Abstract trace properties

Characterizing Control with *Capacity*

- Action control
- Observability control
- Authorization control
- Notification control
- Extensions

Evaluating Concrete Systems with *Capacity*

- An example with *Album*
- Album*: uploading a photo
- Album*: tagging a friend
- Implementations comparison

Conclusions and Perspectives