Type Checking Privacy Policies in the π-calculus

Dimitrios Kouzapas, Anna Philippou

Type checking privacy policies in the π-calculus

Dimitrios Kouzapas[1] and Anna Philippou[2]

[1] Department of Computing, Imperial College London and
Department of Computing Science, University of Glasgow
dk208@doc.ic.ac.uk
[2] Department of Computer Science, University of Cyprus
annap@cs.ucy.ac.cy

Abstract. In this paper we propose a formal framework for studying privacy. Our framework is based on the π-calculus with groups accompanied by a type system for capturing privacy requirements relating to information collection, information processing and information dissemination. The framework incorporates a privacy policy language. We show that a system respects a privacy policy if the typing of the system is compatible with the policy. We illustrate our methodology via analysis of privacy-aware schemes proposed for electronic traffic pricing.

1 Introduction

The notion of privacy is a fundamental notion for society and, as such, it has been an object of study within various scientific disciplines. Recently, its importance is becoming increasingly pronounced as the technological advances and the associated widespread accessibility of personal information is redefining the very essence of the term *privacy*.

A study of the diverse types of privacy, their interplay with technology, and the need for formal methodologies for understanding and protecting privacy is discussed in [19], where the authors follow in their arguments the analysis of David Solove, a legal scholar who has provided a discussion of privacy as a taxonomy of possible privacy violations [18]. According to Solove, privacy violations can be distinguished in four categories: *invasions*, *information collection*, *information processing*, and *information dissemination*. Invasion-related privacy violations are violations that occur on the physical sphere of an individual. The authors of [19] concentrate on the latter three categories and they identify a model for studying them consisting of the *data holder* possessing information about the *data subject* and responsible to protect this information against *unauthorized adversaries* within the environment.

The motivation of this paper stems from the need of developing formal frameworks for reasoning about privacy-related concepts. Such frameworks may provide solid foundations for understanding the notion of privacy and allow to rigorously model and study privacy-related situations. More specifically, our objective is to develop a static method for ensuring that a privacy policy is satisfied by an information system using the π-calculus as the underlying theory.

To achieve this objective, we develop a meta-theory for the π-calculus that captures privacy as policy. Following the model of [19], we create a policy language that enables us to describe privacy requirements for private data over data entities. For each type of private data we expect entities to follow different policy requirements. Thus, we define

policies as objects that describe a hierarchical nesting of entities where each node/entity of the hierarchy is associated with a set of privacy permissions. The choice of permissions encapsulated within a policy language is an important issue because identification of these permissions constitutes, in a sense, a characterization of the notion of privacy. In this work, we make a first attempt of identifying some such permissions, our choice emanating from the more obvious privacy violations of Solove's taxonomy which we refine by considering some common applications where privacy plays a central role.

As an example consider a medical system obligated to protect patient's data. Inside the system a nurse may access patient files to disseminate them to doctors. Doctors are able to process the data without any right to disseminate them. Overall, the data cannot be disclosed outside the hospital. We formalize this policy as follows:

$$\mathsf{t} \gg \mathsf{Hospital} : \{\mathsf{nondisclose}\}[\ \mathsf{Nurse} : \{\mathsf{access}, \mathsf{disclose\ Hospital\ 1}\},\ \mathsf{Doctor} : \{\mathsf{access}, \mathsf{read}, \mathsf{write}\}]$$

where t is the type of the patient's data. The policy describes the existence of the Hospital entity at the higher level of the hierarchy associated with the nondisclose permission signifying that patient data should not be disclosed outside the system. Within this structure, a nurse may access (but not read) a patient file and disseminate the file once (disclose Hospital 1). Similarly a doctor may be given access to a patient file but is also allowed to read and write data within the files (permissions access, read and write).

Moving on to the framework underlying our study, we employ the π-calculus with groups [5]. This calculus extends the π-calculus with the notion of *groups* and an associated type system in a way that controls how data is being disseminated inside a system. It turns out that groups give a natural abstraction for the representation of entities in a system. Thus, we build on the notion of a group of the calculus of [5], and we use the group memberships of processes to distinguish their roles within systems. Information processing issues can be analysed through the use of names of the calculus in input, output and object position to identify when a channel is reading or writing private data or when links to private data are being communicated between groups.

An implementation of the hospital scenario in the π-calculus with groups would be

$$(\nu\ \mathsf{Hospital})((\nu\ \mathsf{Nurse})(\overline{a}\langle l\rangle.\mathbf{0}) \mid (\nu\ \mathsf{Doctor})(a(x).x(y).\overline{x}\langle d\rangle.\mathbf{0}))$$

In this system, $(\nu\mathsf{Hospital})$ creates a new group that is known to the two processes of the subsequent parallel composition while (Nurse) and (Doctor) are groups nested within the Hospital group and available to processes $\overline{a}\langle l\rangle.\mathbf{0}$ and $a(x).x(y).\overline{x}\langle d\rangle.\mathbf{0}$, respectively. The group memberships of the two processes characterize their nature while reflecting the entity hierarchy expressed in the privacy policy defined above.

The types of the names in the above process are defined as $y : \mathsf{t}, d : \mathsf{t}$, that is y and d are values of sensitive data, while $l : \mathsf{Hospital}[\mathsf{t}]$ signifies that l is a channel that can be used only by processes which belong to group Hospital to carry data of type t. Similarly, $a : \mathsf{Hospital}[\mathsf{Hospital}[\mathsf{t}]]$ states that a is a channel that can be used by members of group Hospital, to carry objects of type $\mathsf{Hospital}[\mathsf{t}]$. Intuitively, we may see that this system conforms to the defined policy, both in terms of the group structure as well as the permissions exercised by the processes. Instead, if the nurse were able to engage in a $\overline{l}\langle d\rangle$ action then the defined policy would be violated as it would be the case if the

type of a was defined as $a : \mathsf{Other}[\mathsf{Hospital}[\mathsf{t}]]$ for some distinct group Other. Thus, the encompassing group is essential for capturing requirements of non-disclosure.

Using these building blocks, our methodology is applied as follows: Given a system and a typing we perform type checking to confirm that the system is well-typed while we infer a permission interface. This interface captures the permissions exercised by the system. To check that the system complies with a privacy policy we provide a correspondence between policies and permission interfaces the intention being that: a permission interface satisfies a policy if and only if the system exercises a subset of the allowed permissions of the policy. With this machinery at hand, we state and prove a safety theorem according to which, if a system *Sys* type-checks against a typing Γ and produces an interface Θ, and Θ satisfies a privacy policy $\mathscr{P}$, then *Sys* respects $\mathscr{P}$.

2 The Calculus

Our study of privacy is based on the π-calculus with groups proposed by Cardelli et al. [5]. This calculus is an extension of the π-calculus with the notion of a group and an operation of group creation where a group is a type for channels. In [5] the authors establish a close connection between group creation and secrecy as they show that a secret belonging to a certain group cannot be communicated outside the initial scope of the group. This is related to the fact that groups can never be communicated between processes. Our calculus is based on the π-calculus with groups with some modifications.

We assume the existence of two basic entities: $\mathscr{G}$, ranged over by $G, G_1, \ldots$ is the set of groups and $\mathscr{N}$, ranged over by $a, b, x, y, \ldots$, is the set of names. Furthermore, we assume a set of basic types D, ranged over by $\mathtt{t}_i$, which refer to the basic data of our calculus on which privacy requirements should be enforced. Specifically, we assign each name in $\mathscr{N}$ a type such that a name may either be of some base type $\mathtt{t}$ or of type $G[T]$, where G is the group of the name and T the type of value that can be carried on the name. Given the above, a type is constructed via the following BNF.

$$T \;::=\; \mathtt{t} \;\mid\; G[T]$$

Then the syntax of the calculus is defined at two levels. At the process level, P, we have the standard π-calculus syntax. At the system level, S, we include the group construct, applied both at the level of processes $(\nu\ G)P$, and at the level of systems, $(\nu\ G)S$, the name restriction construct as well as parallel composition for systems.

$$\begin{array}{rcl} P & ::= & x(y{:}T).P \;\mid\; \bar{x}\langle z\rangle.P \;\mid\; (\nu\ a{:}T)P \;\mid\; P_1 \mid P_2 \;\mid\; !P \;\mid\; \mathbf{0} \\ S & ::= & (\nu\ G)P \;\mid\; (\nu\ G)S \;\mid\; (\nu\ a{:}T)S \;\mid\; S_1 \mid S_2 \;\mid\; \mathbf{0} \end{array}$$

In $(\nu\ a{:}T)P$ and $(\nu\ a{:}T)S$, name a is bound in P and S, respectively, and in process $x(y{:}T).P$, name y is bound in P. In $(\nu\ G)P$ and $(\nu\ G)S$, the group G is bound in P and S. We write $\mathtt{fn}(P)$ and $\mathtt{fn}(S)$ for the sets of names free in a process P and a system S, and $\mathtt{fg}(S)$ and $\mathtt{fg}(T)$, for the free groups in a system S and a type T, respectively. Note that free occurrences of groups occur within the types T of a process/system.

We now turn to defining a labelled transition semantics for the calculus. We define a labelled transition semantics instead of a reduction semantics due to a characteristic of

the intended structural congruence in our calculus. In particular, the definition of such a congruence would omit the axiom $(\nu\ G)(S_1 \mid S_2) \equiv (\nu\ G)S_1 \mid S_2$ if $G \notin \mathtt{fg}(S_2)$ as it was used in [5]. This is due to our intended semantics of the group concept which is considered to assign capabilities to processes. Thus, nesting of a process P within some group G, as in $(\nu\ G)P$, cannot be lost even if $G \notin \mathtt{fg}(P)$, since the $(\nu\ G)$ construct has the additional meaning of group membership in our calculus and it instills P with privacy-related permissions as we will discuss in the sequel. The absence of this law renders a reduction semantics rule of parallel composition rather complex.

To define a labelled transition semantics we first define a set of labels:

$$\ell \ ::= \ \tau \ \mid \ x(y) \ \mid \ \overline{x}\langle y\rangle \ \mid \ (\nu\ y)\overline{x}\langle y\rangle$$

Label τ is the internal action whereas labels $x(y)$ and $\overline{x}\langle y\rangle$ are the input and output actions, respectively. Label $(\nu\ y)\overline{x}\langle y\rangle$ is the restricted output where the object y of the action is restricted. Functions $\mathtt{fn}(\ell)$ and $\mathtt{bn}(\ell)$ return the set of the free and bound names of ℓ, respectively. We also define the relation $\mathsf{dual}(\ell,\ell')$ which relates dual actions as

$$\mathsf{dual}(\ell,\ell') \text{ if and only if } \{\ell,\ell'\} = \{x(y),\overline{x}\langle y\rangle\} \text{ or } \{\ell,\ell'\} = \{x(y),(\nu\ y)\overline{x}\langle y\rangle\}.$$

We use the meta-notation ($F \ ::= \ P \ \mid \ S$) to define the labelled transition semantics.

$$x(y:T).P \xrightarrow{x(z)} P\{z/y\}\ (\mathsf{In}) \qquad \overline{x}\langle z\rangle.P \xrightarrow{\overline{x}\langle z\rangle} P\ (\mathsf{Out})$$

$$\frac{F_1 \xrightarrow{\ell} F_1' \quad \mathtt{bn}(\ell)\cap\mathtt{fn}(F_2)=\emptyset}{F_1 \mid F_2 \xrightarrow{\ell} F_1' \mid F_2}\ (\mathsf{ParL}) \qquad \frac{F_2 \xrightarrow{\ell} F_2' \quad \mathtt{bn}(\ell)\cap\mathtt{fn}(F_1)=\emptyset}{F_1 \mid F_2 \xrightarrow{\ell} F_1 \mid F_2'}\ (\mathsf{ParR})$$

$$\frac{F \xrightarrow{\ell} F' \quad x \notin \mathtt{fn}(\ell)}{(\nu\ x:T)F \xrightarrow{\ell} (\nu\ x:T)F'}\ (\mathsf{ResN}) \qquad \frac{F \xrightarrow{\overline{x}\langle y\rangle} F'}{(\nu\ y:T)F \xrightarrow{(\nu\ y)\overline{x}\langle y\rangle} F'}\ (\mathsf{Scope})$$

$$\frac{F \xrightarrow{\ell} F'}{(\nu\ G)F \xrightarrow{\ell} (\nu\ G)F'}\ (\mathsf{ResG}) \qquad \frac{F \equiv_\alpha F'' \quad F'' \xrightarrow{\ell} F'}{F \xrightarrow{\ell} F'}\ (\mathsf{Alpha})$$

$$\frac{P \xrightarrow{\ell} P'}{!P \xrightarrow{\ell} P' \mid !P}\ (\mathsf{Repl}) \qquad \frac{F_1 \xrightarrow{\ell_1} F_1' \quad F_2 \xrightarrow{\ell_2} F_2' \quad \mathsf{dual}(\ell_1,\ell_2)}{F_1 \mid F_2 \xrightarrow{\tau} (\nu\ \mathtt{bn}(\ell_1)\cup\mathtt{bn}(\ell_2))(F_1' \mid F_2')}\ (\mathsf{Com})$$

Fig. 1. The labelled transition system

The labelled transition semantics follows along the lines of standard π-calculus semantics where $\equiv_\alpha$ denotes α-equivalence and the rule for the group-creation construct, (ResG), captures that transitions are closed under group restriction.

3 Policies and Types

In this section we define a policy language and the appropriate type machinery to enforce policies over processes.

3.1 Policies

Typically, privacy policy languages express positive and negative norms that are expected to hold in a system. These norms distinguish what *may* happen, in the case of a positive norm, and what may not happen, in the case of a negative norm on *data attributes* which are types of sensitive data within a system, and, in particular, how the various agents, who are referred to by their *roles*, may/may not handle this data.

The notions of an attribute and a role are reflected in our framework via the notions of base types and groups, respectively. Thus, our policy language is defined in such a way as to specify the allowed and disallowed permissions associated with the various groups for each base type (sensitive data). This is achieved via the following entities:

$$\begin{array}{lrcl}
\text{(Linearities)} & \lambda & ::= & 1 \mid 2 \mid \ldots \mid * \\
\text{(Permissions)} & \mathsf{P} & ::= & \mathsf{read} \mid \mathsf{write} \mid \mathsf{access} \mid \mathsf{disclose}\, G\, \lambda \mid \mathsf{nondisclose} \\
\text{(Hierarchies)} & H & ::= & \varepsilon \mid G : \tilde{p}[H_j]_{j\in J} \\
\text{(Policies)} & \mathscr{P} & ::= & \mathsf{t} \gg H \mid \mathscr{P}; \mathscr{P}
\end{array}$$

Specifically, we define the set of *policy permissions* P: they express that data may be read (read) and written (write), that links to data may be accessed (access) or disclosed within some group G up to λ times ($\mathsf{disclose}\, G\, \lambda$). Notation λ is either a natural number or equal to $*$ which denotes an infinite number. While the above are positive permissions, permission $\mathsf{nondisclose}$ is a negative permission and, when associated with a group and a base type, expresses that the base type cannot be disclosed to any participant who is not a member of the group.

In turn, a policy has the form $\mathsf{t}_1 \gg H_1; \ldots; \mathsf{t}_n \gg H_n$ assigning a structure H_i to each type of sensitive data t_i. The components H_i, which we refer to as *permission hierarchies*, specify the group-permission associations for each base type. A permission hierarchy H has the form $G{:}\widetilde{p}\,[H_1, \ldots, H_m]$, and expresses that an entity belonging to group G has rights $\widetilde{p}$ to the data in question and if additionally it is a member of some group G_i where $H_i = G_i{:}\widetilde{p}_i\,[\ldots]$, then it also has the rights $\widetilde{p}_i$, and so on.

We define the auxiliary functions $\mathsf{groups}(H)$ and $\mathsf{perms}(H)$ so as to gather the sets of groups and the set of permissions, respectively, inside a hierarchy structure:

$$\mathsf{groups}(H) = \begin{cases} \{G\} \cup (\bigcup_{j\in J} \mathsf{groups}(H_j)) & \text{if } H = G : \tilde{p}[H_j]_{j\in J} \\ \emptyset & \text{if } H = \varepsilon \end{cases}$$

$$\mathsf{perms}(H) = \begin{cases} \tilde{p} \cup (\bigcup_{j\in J} \mathsf{perms}(H_j)) & \text{if } H = G : \tilde{p}[H_j]_{j\in J} \\ \emptyset & \text{if } H = \varepsilon \end{cases}$$

We say that a policy $\mathscr{P} = \mathsf{t}_1 \gg H_1; \ldots; \mathsf{t}_n \gg H_n$ is *well formed*, written $\mathscr{P} : \diamond$, if it satisfies the following:

1. The t_i are distinct.
2. If $H = G : \widetilde{p}[H_j]_{j\in J}$ occurs within some H_i then $G \notin \mathsf{groups}(H_j)$ for all $j \in J$, that is, the group hierarchy is acyclic.
3. If $H = G : \widetilde{p}[H_j]_{j\in J}$ occurs within some H_i, $\mathsf{nondisclose} \in \widetilde{p}$ and $\mathsf{disclose}\, G'\, \lambda \in \mathsf{perms}(H_j)$ for some $j \in J$, then $G' \in \mathsf{groups}(H)$. In words, no non-disclosure requirement imposed at some level of a hierarchy is in conflict with a disclosure requirement granted in its sub-hierarchy.

Hereafter, we assume that policies are well-formed policies. As a shorthand, we write $G : \tilde{p}$ for $G : \tilde{p}[\varepsilon]$ and we abbreviate G for $G : \emptyset$.

As an example, consider a hospital containing the departments of surgery (Surgery), cardiology (Cardiology), and psychotherapy (Psychotherapy), where cardiologists of group Cardio belong to the cardiology department, surgeons of group Surgeon belong to the surgery department, psychiatrists of group Psy belong to the psychotherapy department, and CarSurgeon refers to doctors who may have a joint appointment with the surgery and cardiology departments. Further, let us assume the existence of data of type MedFile which (1) should not be disclosed to any participant outside the Hospital group, (2) may be read, written, accessed and disclosed freely within both the surgery and the cardiology departments and (3) may be read, written and accessed but not disclosed outside the psychotherapy department. We capture these requirements via policy $\mathscr{P} = \mathsf{MedFile} \gg H$ where $p_{cd} = \{\mathsf{read}, \mathsf{access}, \mathsf{write}, \mathsf{disclose\ Cardiology} *\}$, $p_{sd} = \{\mathsf{read}, \mathsf{access}, \mathsf{write}, \mathsf{disclose\ Surgery} *\}$, $p_{pd} = \{\mathsf{nondisclose}, \mathsf{read}, \mathsf{access}, \mathsf{write}\}$,

$$\begin{aligned} H = \mathsf{Hospital}{:}\{\mathsf{nondisclose}\}\,[&\mathsf{Cardiology}{:}p_{cd}[\mathsf{Cardio}, \mathsf{CarSurgeon}],\\ &\mathsf{Surgery}{:}p_{sd}[\mathsf{Surgeon}, \mathsf{CarSurgeon}],\\ &\mathsf{Psychotherapy}{:}p_{pd}[\mathsf{Psy}]] \end{aligned}$$

At this point we note that, as illustrated in the above policy, hierarchies need not be tree-structured: group CarSurgeon may be reached via both the Hospital, Cardiology path as well as the Hospital, Surgery path. In effect this allows to define a process $(\nu\ \mathsf{Hospital})(\nu\ \mathsf{Cardiology})(\nu\ SD)(\nu\ \mathsf{CarSurgeon})P$ belonging to four groups and inheriting the permissions of each one of them.

3.2 The Type System

We proceed to define a typing system for the calculus.

Typing Judgements. The environment on which type checking is carried out consists of the component Γ. During type checking we infer the two additional structures of Δ-environments and Θ-interfaces as follows

$$\begin{array}{lcl} \Gamma & ::= & \emptyset \mid \Gamma \cdot x : T \mid \Gamma \cdot G \\ \Delta & ::= & \emptyset \mid \mathtt{t} : \tilde{p} \cdot \Delta \\ \Theta & ::= & \mathtt{t} \gg H^{\theta}; \Theta \mid \mathtt{t} \gg H^{\theta} \end{array}$$

with $H^{\theta} ::= G[H^{\theta}] \mid G[\tilde{p}]$. Note that H^{θ} captures a special type of hierarchies where the nesting of groups is linear. We refer to H^{θ} as interface hierarchies. The domain of environment Γ, $\mathtt{dom}(\Gamma)$, contains all groups and names recorded in Γ. Environment Δ assigns permissions to sensitive data types t. When associated with a base type t, permissions read and write express that it is possible to read/write data of type t along channels of type $G[\mathrm{t}]$ for any group G. Permission access, when associated with a type t, expresses that it is possible to receive a channel of type $G[\mathrm{t}]$ for any G and, finally, if permission disclose $G\lambda$ is associated with t then it is possible to send channels of type $G[\mathrm{t}]$ for up to λ times. Thus, while permissions read and write are related to manipulating sensitive data, permissions access and disclose are related to manipulating links

to sensitive data. Finally, interface Θ associates a sensitive type with a linear hierarchy of groups and a set of permissions, namely, an entity of the form $G_1[G_2[\ldots G_n[\widetilde{p}]\ldots]]$.

We define three typing judgements: $\Gamma \vdash x \triangleright T$, $\Gamma \vdash P \triangleright \Delta$ and $\Gamma \vdash S \triangleright \Theta$. Judgement $\Gamma \vdash x \triangleright T$ says that under typing environment Γ, name x has type T. Judgement $\Gamma \vdash P \triangleright \Delta$ stipulates that process P is well typed under the environment Γ and produces a permission environment Δ. In this judgement, Γ records the types of the names of P and Δ records the permissions exercised by the names in P for each base type. Finally, judgement $\Gamma \vdash S \triangleright \Theta$ defines that system S is well typed under the environment Γ and produces interface Θ which records the group memberships of all components of S as well as the permissions exercised by each component.

Typing System. We now move on to our typing system. We begin with some useful notation. We write:

$$\Delta_T^r = \begin{cases} \mathsf{t} : \mathsf{read} & \text{if } T = \mathsf{t} \\ \mathsf{t} : \mathsf{access} & \text{if } T = G[\mathsf{t}] \\ \emptyset & \text{otherwise} \end{cases} \qquad \Delta_T^w = \begin{cases} \mathsf{t} : \mathsf{write} & \text{if } T = \mathsf{t} \\ \mathsf{t} : \mathsf{disclose}\, G\, 1 & \text{if } T = G[\mathsf{t}] \\ \emptyset & \text{otherwise} \end{cases}$$

Furthermore, we define the $\uplus$ operator over permissions:

$$\begin{aligned} \tilde{p}_1 \uplus \tilde{p}_2 = \{ & p \mid p \in \tilde{p}_1 \cup \tilde{p}_2 \wedge p \neq \mathsf{disclose}\, G\, \lambda \wedge p \neq \mathsf{nondisclose} \} \\ & \cup \{ \mathsf{disclose}\, G\, (\lambda_1 + \lambda_2) \mid \mathsf{disclose}\, G\, \lambda_1 \in \tilde{p}_1 \wedge \mathsf{disclose}\, G\, \lambda_2 \in \tilde{p}_2 \} \\ & \cup \{ \mathsf{disclose}\, G\, \lambda \mid \mathsf{disclose}\, G\, \lambda \in \tilde{p}_1 \wedge \mathsf{disclose}\, G\, \lambda' \notin \tilde{p}_2 \} \\ & \cup \{ \mathsf{disclose}\, G\, \lambda \mid \mathsf{disclose}\, G\, \lambda' \notin \tilde{p}_1 \wedge \mathsf{disclose}\, G\, \lambda \in \tilde{p}_2 \} \end{aligned}$$

Operator $\uplus$ adds two permission sets by taking the union of the non nondisclose permissions modulo adding the linearities of the $\mathsf{disclose}\, G\, \lambda$ permissions. We extend the $\uplus$ operator for Δ-environments: assuming $\mathsf{t} : \emptyset \in \Delta$ if $\mathsf{t} : \tilde{p} \notin \Delta$, we define $\Delta_1 \uplus \Delta_2 = \{ \mathsf{t} : \tilde{p}_1 \uplus \tilde{p}_2 \mid \mathsf{t} : \tilde{p}_1 \in \Delta_1, \mathsf{t} : \tilde{p}_2 \in \Delta_2 \}$.

Finally, we define the $\oplus$ operator as:

$$\begin{aligned} G \oplus (\mathsf{t}_1 : \tilde{p}_1, \ldots, \mathsf{t}_m : \widetilde{p}_m) &= \mathsf{t}_1 \gg G[\tilde{p}_1], \ldots, \mathsf{t}_m \gg G[\tilde{p}_m] \\ G \oplus (\mathsf{t}_1 \gg H_1^\theta, \ldots, \mathsf{t}_m \gg H_m^\theta) &= (\mathsf{t}_1 \gg G[H_1^\theta], \ldots, \mathsf{t}_m \gg G[H_m^\theta]) \end{aligned}$$

Operator $\oplus$ when applied to a group G and an interface Δ attaches G to all permission sets of Δ, thus yielding a Θ interface, whereas, when applied to a group G and an interface Θ, it attaches group G to all interface hierarchies of Θ.

The typing system is defined in Fig. 2. Rule (Name) is used to type names: in name typing we require that all group names of the type are present in Γ. Process $\mathbf{0}$ can be typed under any typing environment (axiom (Nil)) to infer the empty Δ-interface.

Rule (In) types the input-prefixed process. If environment Γ extended with the type of y produces Δ as an interface of P, we conclude that the process $x(y).P$ produces an interface where the type of T is extended with the permissions Δ_T^r, where (i) if T is base type t then Δ is extended by $\mathsf{t} : \mathsf{read}$ since the process is reading an object of type t, (ii) if $T = T'[\mathsf{t}]$ then Δ is extended by $\mathsf{t} : \mathsf{access}$, since the process has obtained access to a link for base type t and (iii) Δ remains unaffected otherwise.

$$\text{(Name)}\ \frac{\mathtt{fg}(T) \subseteq \Gamma}{\Gamma \cdot x : T \vdash x \triangleright T} \qquad \text{(Nil)}\ \ \Gamma \vdash \mathbf{0} \triangleright \emptyset$$

$$\text{(In)}\ \frac{\Gamma \cdot y : T \vdash P \triangleright \Delta \quad \Gamma \vdash x \triangleright G[T]}{\Gamma \vdash x(y:T).P \triangleright \Delta \uplus \Delta_T^r} \qquad \text{(Out)}\ \frac{\Gamma \vdash P \triangleright \Delta \quad \Gamma \vdash x \triangleright G[T] \quad \Gamma \vdash y \triangleright T}{\Gamma \vdash \overline{x}\langle y \rangle.P \triangleright \Delta \uplus \Delta_T^w}$$

$$\text{(ParP)}\ \frac{\Gamma \vdash P_1 \triangleright \Delta_1 \qquad \Gamma \vdash P_2 \triangleright \Delta_2}{\Gamma \vdash P_1 \mid P_2 \triangleright \Delta_1 \uplus \Delta_2} \qquad \text{(ParS)}\ \frac{\Gamma \vdash S_1 \triangleright \Theta_1 \qquad \Gamma \vdash S_2 \triangleright \Theta_2}{\Gamma \vdash S_1 \mid S_2 \triangleright \Theta_1 \cdot \Theta_2}$$

$$\text{(ResNP)}\ \frac{\Gamma \cdot x : T \vdash P \triangleright \Delta}{\Gamma \vdash (\nu\, x : T)P \triangleright \Delta} \qquad \text{(ResNS)}\ \frac{\Gamma \cdot x : T \vdash S \triangleright \Theta}{\Gamma \vdash (\nu\, x : T)S \triangleright \Theta}$$

$$\text{(ResGP)}\ \frac{\Gamma \cdot G \vdash P \triangleright \Delta}{\Gamma \vdash (\nu\, G)P \triangleright G \oplus \Delta} \qquad \text{(ResGS)}\ \frac{\Gamma \cdot G \vdash S \triangleright \Theta}{\Gamma \vdash (\nu\, G)S \triangleright G \oplus \Theta}$$

$$\text{(Rep)}\ \frac{\Gamma \vdash P \triangleright \Delta}{\Gamma \vdash\, !P \triangleright \Delta^!}$$

Fig. 2. The Typing System

Rule (Out) is similar: If y is of type T, x of type $G[T]$ and Δ is the permission interface for P, then, $\overline{x}\langle y \rangle.P$ produces an interface which extends Δ with permissions Δ_T^w. These permissions are (i) $\{\mathsf{t} : \mathsf{write}\}$ if $T = \mathsf{t}$ since the process is writing data of type t, (ii) $\{\mathsf{disclose}\, G\, 1\}$ if $T = G[\mathrm{t}]$, since the process is disclosing once a link to private data via a channel of group G, and (iii) the empty set of permissions otherwise.

Rule (ParP) uses the $\uplus$ operator to compose the process interfaces of P_1 and P_2. Parallel composition of systems, rule (ParS), concatenates the system interfaces of S_1 and S_2. For name restriction, (ResNP) specifies that if P type checks within an environment $\Gamma \cdot x : T$, then $(\nu x)P$ type checks in environment Γ. (ResNS) is defined similarly. Moving on to group creation, for rule (ResGP) we have that, if P produces a typing Δ, then system $(\nu\, G)P$ produces the Θ-interface $G \oplus \Delta$whereas for rule (ResGS), we have that if S produces a typing interface Θ then process $(\nu\, G)S$ produces interface $G \oplus \Theta$ Thus, enclosing a system within an $(\nu\, G)$ operator results in adding G to the group memberships of each of the components.

Finally, for replication, axiom (Rep) states that if P produces an interface Δ then $!P$ produces an interface $\Delta^!$, where $\Delta^!$ is such that if a type is disclosed $\lambda > 1$ in Δ then it is disclosed for an unlimited number of times in $\Delta^!$. That is, $\Delta^! = \{\mathsf{t} : \tilde{p}^! \mid \mathsf{t} : \tilde{p} \in \Delta\}$, where $\tilde{p}^! = \{p \in \tilde{p} \mid p \neq \mathsf{disclose}\, G\, \lambda\} \cup \{\mathsf{disclose}\, G * \mid \mathsf{disclose}\, G\, \lambda \in \Delta\}$. Note that the type system never assigns the nondisclose permissions, thus interfaces are never inferred on the nondisclose permission. This is the reason the nondisclose permission is ignored in the definition of the $\uplus$ operator.

As an example consider $S = (\nu\ G_1)(\nu\ G_2)P$ where $P = !get(loc : T_l).\overline{put}\langle loc \rangle.\mathbf{0}$. Further, suppose the existence of a base type Loc and types $T_l = G_1[\mathsf{Loc}]$, $T_r = G_2[T_l]$ and $T_s = G_1[T_l]$. Let us write $\Gamma = get : T_r \cdot put : T_s \cdot loc : T_l$. Then we have:

$$\begin{array}{lll} \Gamma & \vdash & \mathbf{0} \triangleright \emptyset \qquad \text{by (Nil)} \\ \Gamma & \vdash & \overline{put}\langle loc \rangle.\mathbf{0} \triangleright \{\mathsf{Loc} : \{\mathsf{disclose}\, G_1\, 1\} \qquad \text{by (Out)} \end{array}$$

$\Gamma \vdash get(loc : T_l).\overline{put}\langle loc\rangle.\mathbf{0} \triangleright \{\mathsf{Loc} : \{\mathsf{disclose}\ G_1\, 1, \mathsf{access}\}$ by (In)
$\Gamma \vdash\ !get(loc : T_l).\overline{put}\langle loc\rangle.\mathbf{0} \triangleright \{\mathsf{Loc} : \{\mathsf{disclose}\ G_1\, *, \mathsf{access}\}$ by (Rep)
$\Gamma \vdash (\nu\ G_2)P \triangleright \mathsf{Loc} \gg G_2 : [\{\mathsf{disclose}\ G_1\, *, \mathsf{access}\}]$ by (ResNP)
$\Gamma \vdash S \triangleright \mathsf{Loc} \gg G_1 : [G_2 : [\{\mathsf{disclose}\ G_1\, *, \mathsf{access}\}]]$ by (ResNS)

4 Soundness and Safety

In this section we establish soundness and safety results for our framework. Missing proofs of results can be found in the appendix. First, we establish that typing is preserved under substitution.

Lemma 1 (Substitution). If $\Gamma \cdot x : T \vdash P \triangleright \Delta$ then $\Gamma \cdot y : T \vdash P\{y/x\} \triangleright \Delta$.

The next definition defines an operator that captures the changes on the interface environment when a process executes an action.

Definition 1 ($\Theta_1 \preceq \Theta_2$).

1. $\tilde{p}_1 \preceq \tilde{p}_2$ if (i) for all $p \in \tilde{p}_1$ and $p \neq \mathsf{disclose}\ G\lambda$ implies $p \in \tilde{p}_2$, and (ii) for all $\mathsf{disclose}\ G\lambda \in \tilde{p}_1$ implies $\mathsf{disclose}\ \lambda' G \in \tilde{p}_2$ and $\lambda' \geq \lambda$ or $\lambda' = *$.
2. $\Delta_1 \preceq \Delta_2$ if $\forall \mathsf{t}, \mathsf{t} : \tilde{p}_1 \in \Delta_1$ implies that $\mathsf{t} : \tilde{p}_2 \in \Delta_2$ and $\tilde{p}_1 \preceq \tilde{p}_2$.
3. (i) $G[\tilde{p}_1] \preceq G[\tilde{p}_2]$ if $\tilde{p}_1 \preceq \tilde{p}_2$, and (ii) $G[H_1] \preceq G[H_2]$ if $H_1 \preceq H_2$.
4. $\Theta_1 \preceq \Theta_2$ if (i) $\mathtt{dom}(\Theta_1) = \mathtt{dom}(\Theta_2)$, and (ii) for all t, $\mathsf{t} \gg H_1 \in \Theta_1$ implies that $\mathsf{t} \gg H_2 \in \Theta_2$ and $H_1 \succeq H_2$.

Specifically, when a process executes an action we expect a name to maintain or lose its interface capabilities that are expressed through the typing of the name.

We are now ready to define the notion of *satisfaction* of a policy $\mathscr{P}$ by a permission interface Θ thus connecting our type system with policy compliance.

Definition 2.

– Consider a policy hierarchy $H = G : \tilde{p}[H_j]_{j\in J}$ and an interface hierarchy H^θ. We say that H^θ *satisfies* H, written $H \Vdash H^\theta$, if:

$$\frac{\mathsf{groups}(H^\theta) = G \cup \bigcup_{j\in J} \mathsf{groups}(H^\theta_j) \qquad \forall j \in J, H_j \Vdash H^\theta_j \qquad \mathsf{perms}(H^\theta) \preceq (\uplus_{j\in J} \mathsf{perms}(H^\theta_j)) \uplus \tilde{p}}{G : \tilde{p}[H_j]_{j\in J} \Vdash H^\theta}$$

– Consider a policy $\mathscr{P}$ and an interface Θ. Θ *satisfies* $\mathscr{P}$, written $\mathscr{P} \Vdash \Theta$, if:

$$\frac{H \Vdash H^\theta}{\mathsf{t} \gg H; \mathscr{P} \Vdash \mathsf{t} \gg H^\theta} \qquad \frac{H \Vdash H^\theta \qquad \mathscr{P} \Vdash \Theta}{\mathsf{t} \gg H; \mathscr{P} \Vdash \mathsf{t} \gg H^\theta; \Theta}$$

According to the definition of $H \Vdash H^\theta$, an interface hierarchy H^θ satisfies a policy hierarchy H, if its groups can be decomposed into a partition $\{G\} \cup \bigcup_{j\in J} G_j$, such that there exist interface hierarchies H^θ_j referring to groups G_j, each satisfying hierarchy H_j and where the union of the assigned permissions H^θ_j with permissions $\tilde{p}$ is a superset

of the permissions of H^θ, that is, $\mathsf{perms}(H^\theta) \preceq (\uplus_{j\in J}\mathsf{perms}(H^\theta_j)) \uplus \tilde{p}$. Similarly, a Θ-interface satisfies a policy, $\mathscr{P} \Vdash \Theta$, if for each component $\mathsf{t} \gg H^\theta$ of Θ, there exists a component $\mathsf{t} \gg H$ of $\mathscr{P}$ such that H^θ satisfies H. A direct corollary of the definition is the preservation of the $\preceq$ operator over the satisfiability relation:

Corollary 1. If $\mathscr{P} \Vdash \Theta_1$ and $\Theta_2 \preceq \Theta_1$ then $\mathscr{P} \Vdash \Theta_2$.

The next definition formalises when a system satisfies a policy:

Definition 3 (Policy Satisfaction). Let $\mathscr{P} : \diamond$. We say that S satisfies $\mathscr{P}$, written $\mathscr{P} \vdash S$, if $\Gamma \vdash S \triangleright \Theta$ for some Γ and Θ such that $\mathscr{P} \Vdash \Theta$.

We may now state our result on type preservation by action execution of processes.

Theorem 1 (Type Preservation).

1. Let $\Gamma \vdash P \triangleright \Delta$ and $P \xrightarrow{\ell} P'$ then $\Gamma \vdash P' \triangleright \Delta'$ and $\Delta' \preceq \Delta$.
2. Let $\Gamma \vdash S \triangleright \Theta$ and $S \xrightarrow{\ell} S'$ then $\Gamma \vdash S' \triangleright \Theta'$ and $\Theta' \preceq \Theta$.

Corollary 2. Let $\mathscr{P} \vdash S$ and $S \xrightarrow{\ell} S'$ then $\mathscr{P} \vdash S'$.

Let $\mathsf{countLnk}(P,\Gamma,G[t])$ count the number of output prefixes of the form $\overline{x}\langle y\rangle$ in process P where $x : G[\mathsf{t}]$ for some base type t. (This can be defined inductively on the structure of P.) Moreover, given a policy hierarchy H and a set of groups $\widetilde{G}$, let us write $H_{\widetilde{G}}$ for the interface hierarchy such that (i) $\mathsf{groups}(H_{\widetilde{G}}) = \widetilde{G}$, (ii) $H \Vdash H_{\widetilde{G}}$ and, (iii) for all H^θ such that $\mathsf{groups}(H^\theta) = \widetilde{G}$ and $H \Vdash H^\theta$, then $\mathsf{perms}(H^\theta) \preceq \mathsf{perms}(H_{\widetilde{G}})$. Intuitively, $H_{\widetilde{G}}$ captures an interface hierarchy with the maximum possible permissions for groups $\widetilde{G}$ as determined by H. We may now define the notion of the *error process* which clarifies the satisfiability relation between the policies and processes.

Definition 4 (Error Process). Consider a policy $\mathscr{P}$, an environment Γ and a system

$$S \equiv (\nu\, G_1)(\nu\, \tilde{x}_1 : \tilde{T}_1)((\nu\, G_2)(\nu\, \tilde{x}_2 : \tilde{T}_2)(\ldots((\nu\, G_n)(\nu\, \tilde{x}_n : \tilde{T}_n)P \mid Q \mid S_n)\ldots) \mid S_1)$$

System S is an *error process* with respect to $\mathscr{P}$ and Γ, if there exists t such that $\mathscr{P} = \mathsf{t} \gg H; \mathscr{P}'$ and at least one of the following holds, where $\widetilde{G} = \langle G_1, \ldots, G_n\rangle$:

1. $\mathsf{read} \notin \mathsf{perms}(H_{\widetilde{G}})$ and $\exists x$ such that $\Gamma \vdash x \triangleright G[\mathsf{t}]$ and $P = x(y).P'$.
2. $\mathsf{write} \notin \mathsf{perms}(H_{\widetilde{G}})$ and $\exists x$ such that $\Gamma \vdash x \triangleright G[\mathsf{t}]$ and $P = \overline{x}\langle y\rangle.P'$.
3. $\mathsf{access} \notin \mathsf{perms}(H_{\widetilde{G}})$ and $\exists x$ such that $\Gamma \vdash x \triangleright G[\mathsf{t}]$ and $P = y(x).P'$.
4. $\mathsf{disclose}\ G'\lambda \notin \mathsf{perms}(H_{\widetilde{G}})$ and $\exists x,y$ such that $\Gamma \vdash x \triangleright G[\mathsf{t}]$, $\Gamma \vdash y \triangleright G'[G[\mathsf{t}]]$ and $P = \overline{y}\langle x\rangle.P'$.
5. $\mathsf{disclose}\ G\lambda \in \mathsf{perms}(H_{\widetilde{G}})$, $\lambda \neq *$ and $\mathsf{countLnk}(P,\Gamma,G[\mathsf{t}]) > \lambda$
6. there exists a sub-hierarchy of H, $H' = G_k : \tilde{p}[H_i]_{i\in I}, 1 \leq k \leq n$ with $\mathsf{nondisclose} \in \tilde{p}$ and $\exists, x, y$ such that $\Gamma \vdash x \triangleright G[t]$, $\Gamma \vdash y \triangleright G'[G[t]]$ and $P = \overline{y}\langle x\rangle.P'$ with $G' \notin \mathsf{groups}(H')$.

The first two error processes expect that a process with no read or write permissions on a certain level of the hierarchy should not have, respectively, a prefix receiving or sending an object typed with the private data. Similarly an error process with no access permission on a certain level of the hierarchy should not have an input-prefixed subject with object a link to private data. An output-prefixed process that send links through a channel of sort G' is an error process if it is found in a specific group hierarchy with no disclose $G'\lambda$ permission. In the fifth clause, a process is an error if the number of output prefixes to links in its definition (counted with the $\mathsf{countLnk}(P,\Gamma,t)$ function) are more than the λ in the disclose $G\lambda$ permission of the process's hierarchy. Finally, if a policy specifies that no data should be disclosed outside some group G, then a process should not be able to send private data links to groups that are not contained within the hierarchy of G.

As expected, if a process is an error with respect to a policy $\mathscr{P}$ and an environment Γ its Θ-interface does not satisfy $\mathscr{P}$:

Lemma 2. Let system S be an error process with respect to well formed policy $\mathscr{P}$ and sort Γ. If $\Gamma \vdash S \triangleright \Theta$ then $\mathscr{P} \not\Vdash \Theta$.

By Corollary 2 and Lemma 2 we conclude with our safety theorem which verifies that the satisfiability of a policy by a typed process is preserved by the semantics.

Theorem 2 (Safety). If $\Gamma \vdash S \triangleright \Theta$, $\mathscr{P} \Vdash \Theta$ and $S \xrightarrow{\ell}{}^{*} S'$ then S' is not an error with respect to policy $\mathscr{P}$.

5 Example

Electronic Traffic Pricing (ETP) is an electronic toll collection scheme in which the fee to be paid by drivers depends on the road usage of their vehicles where factors such as the type of roads used and the times of the usage determine the toll. To achieve this, for each vehicle detailed time and location information must be collected and processed and the due amount can be calculated with the help of a digital tariff and a road map. A number of possible implementation schemes may be considered for this system [8]. In the centralized approach, all location information is communicated to the pricing authority which computes the fee to be paid based on the received information. In the decentralized approach the fee is computed locally on the car via the use of a third trusted entity such as a smart card. In the following subsections we consider these approaches and their associated privacy characteristics.

5.1 The centralized approach

This approach makes use of on-board equipment (OBE) which computes regularly the geographical position of the car and forwards it to the Pricing Authority (PA). To avoid drivers tampering with their OBE and communicating false information, the authorities may perform checks on the spot to confirm that the OBE is reliable.

We may model this system with the aid of five groups: ETP corresponds to the entirety of the ETP system, Car refers to the car and is divided into the OBE and the

GPS subgroups, and PA refers to the pricing authority. As far as types are concerned, we assume the existence of two base types: Loc referring to the attribute of locations and Fee referring to the attribute of fees. We write $T_l = \mathsf{ETP}[\mathsf{Loc}]$, $T_r = \mathsf{Car}[T_l]$, $T_{pa} = \mathsf{ETP}[T_l]$, $T_x = \mathsf{ETP}[T_l]$ and $T_{sc} = \mathsf{ETP}[T_x]$.

$$
\begin{aligned}
O &= !\mathit{read}(\mathit{loc} : T_l).\overline{\mathit{topa}}\langle \mathit{loc}\rangle.\mathbf{0} \\
&\quad \mid !\mathit{spotcheck}(s : T_x).\mathit{read}(\mathit{ls} : T_l).\bar{s}\langle \mathit{ls}\rangle.\mathbf{0} \\
L &= !(\nu\ \mathit{newl} : T_l)\overline{\mathit{read}}\langle \mathit{newl}\rangle.\mathbf{0} \\
A &= !\mathit{topa}(z : T_l).z(l : \mathsf{Loc}).\mathbf{0} \\
&\quad \mid !\overline{\mathit{send}}\langle \mathit{fee}\rangle.\mathbf{0} \\
&\quad \mid !(\nu\ x : T_x)\overline{\mathit{spotcheck}}\langle x\rangle.x(y : T_l).y(l_s : \mathsf{Loc}).\mathbf{0} \\
\mathsf{System} &= (\nu\ \mathsf{ETP})(\nu\ \mathit{spotcheck} : T_{sc})(\nu\ \mathit{topa} : T_{pa}) \\
&\quad [\ (\nu\ \mathsf{PA})A \ \mid\ (\nu\ \mathsf{Car})((\nu\ \mathit{read} : T_r)((\nu\ \mathsf{OBE})O \mid (\nu\ \mathsf{GPS})L))\ ]
\end{aligned}
$$

In the above model we have the component of the OBE, O, belonging to group OBE, and the component responsible for computing the current location, L, belonging to group GPS. These two components are nested within the Car group and share the private name *read* on which it is possible for L to pass to O a name via which the current location may be read. The OBE O may spontaneously read on name *read* or it may enquire the current location for the purposes of a spot check. Such a check is initiated by the pricing authority A who may engage in three different activities: Firstly, it may receive a name z from the OBE via channel *topa* and then use z for reading the car location (action $z(l)$). Secondly, it may periodically compute the fee to be paid and communicate the link ($\mathit{fee} : \mathsf{ETP}[\mathsf{Fee}]$) via name $\mathit{send} : \mathsf{ETP}[\mathsf{ETP}[\mathsf{Fee}]]$. Thirdly, it may initiate a spot check, during which it creates and sends the OBE a new channel via which the OBE is expected to return the current location for a verification check.

By applying the rules of our type system we may show that $\Gamma \vdash \mathsf{System} \triangleright \Theta$, where $\Gamma = \{\mathit{fee} : \mathsf{ETP}[\mathsf{Fee}], \mathit{send} : \mathsf{ETP}[\mathsf{ETP}[\mathsf{Fee}]]\}$ and where

$$
\begin{aligned}
\Theta = {} & \mathsf{Fee} \gg \mathsf{ETP}[\mathsf{PA}[\{\mathsf{disclose\ ETP}*\}]];\ \ \mathsf{Loc} \gg \mathsf{ETP}[\mathsf{PA}[\{\mathsf{access}, \mathsf{read}\}]]; \\
& \mathsf{Loc} \gg \mathsf{ETP}[\mathsf{Car}[\mathsf{OBE}[\{\mathsf{access}, \mathsf{disclose\ ETP}*\}]]];\ \ \mathsf{Loc} \gg \mathsf{ETP}[\mathsf{Car}[\mathsf{GPS}[\{\mathsf{disclose\ Car}*\}]]]
\end{aligned}
$$

A possible privacy policy for the system might be one that states that locations may be freely forwarded by the OBE. We may define this by $\mathscr{P} = \mathsf{Loc} \gg H$ where

$$
\begin{aligned}
H = \mathsf{ETP} : \mathsf{nondisclose}\ [\mathsf{Car} : [\mathsf{OBE} : \{\mathsf{access}, \mathsf{disclose\ ETP}*\}, \\
\mathsf{GPS} : \{\mathsf{disclose\ Car}*\}], \\
\mathsf{PA} : \{\mathsf{access}, \mathsf{read}\}]
\end{aligned}
$$

We have that $\mathscr{P} \Vdash \mathsf{Loc} \gg \mathsf{ETP}[\mathsf{PA}[\{\mathsf{access}, \mathsf{read}\}]]$, since the permissions assigned to groups ETP and PA by the policy are equal to $\{\mathsf{access}, \mathsf{read}\} \preceq \{\mathsf{access}, \mathsf{read}\} = \mathsf{perms}(\mathsf{ETP}[\mathsf{PA}[\{\mathsf{access}, \mathsf{read}\}]])$. Similarly, $\mathscr{P} \Vdash \mathsf{Loc} \gg \mathsf{ETP}[\mathsf{Car}[\mathsf{OBE}[\{\mathsf{access}, \mathsf{disclose\ ETP}*\}]]]$ and $\mathscr{P} \Vdash \mathsf{Loc} \gg \mathsf{ETP}[\mathsf{Car}[\mathsf{GPS}[\{\mathsf{disclose\ Car}*\}]]]$. Thus, we conclude that System satisfies $\mathscr{P}$.

This architecture is simple but also very weak in protecting the privacy of individuals: the fact that the PA gets detailed travel information about every vehicle constitutes a privacy and security threat. An alternative implementation that limits the transmission of locations is presented in the second implementation proposal presented below.

5.2 The decentralized approach

To avoid the disclosure of the complete travel logs of a system this solution employs a third trusted entity (e.g. smart card) to make computations of the fee locally on the car and send its value to the authority which in turn may make spot checks to obtain evidence on the correctness of the calculation.

The policy here would require that locations can be communicated for at most a small fixed amount of times and that the OBE may read the fee computed by the smart card but not change its value. Precisely, the new privacy policy might be:

$$
\begin{array}{l}
\mathsf{Loc} \gg \mathsf{ETP} : \mathsf{nondisclose}\ [\\
\quad \mathsf{Car} : [\\
\quad\quad \mathsf{OBE} : \{\mathsf{access}, \mathsf{disclose}\ \mathsf{ETP}\, 2\}, \\
\quad\quad \mathsf{GPS} : \{\mathsf{disclose}\ \mathsf{Car}*\}, \\
\quad\quad \mathsf{SC} : \{\mathsf{access}, \mathsf{read}\}], \\
\quad \mathsf{PA} : \{\mathsf{access}, \mathsf{read}\}]
\end{array}
\qquad
\begin{array}{l}
\mathsf{Fee} \gg \mathsf{ETP} : \mathsf{nondisclose}\ [\\
\quad \mathsf{Car} : [\\
\quad\quad \mathsf{OBE} : \{\}, \\
\quad\quad \mathsf{GPS} : \{\}, \\
\quad\quad \mathsf{SC} : \{\mathsf{write}, \mathsf{disclose}\ \mathsf{ETP}*\}], \\
\quad \mathsf{PA} : \{\mathsf{access}, \mathsf{read}\}]
\end{array}
$$

The new system as described above may be modelled as follows, where we have a new group SC and a new component S, a smart card, belonging to this group:

$$
\begin{aligned}
S &= !\mathit{read}(\mathit{loc} : T_l).\mathit{loc}(l : \mathsf{Loc}).(\nu\ \mathit{newval} : \mathit{Fee})\overline{\mathit{fee}}\langle \mathit{newval}\rangle.\overline{\mathit{send}}\langle \mathit{fee}\rangle.\mathbf{0} \\
O &= \mathit{spotcheck}(s_1 : T_x).\mathit{read}(ls_1 : T_l).\overline{s_1}\langle ls_1\rangle.\mathit{spotcheck}(s_2 : T_x).\mathit{read}(ls_2 : T_l).\overline{s_2}\langle ls_2\rangle.\mathbf{0} \\
L &= !(\nu\ \mathit{newl} : T_l)\overline{\mathit{read}}\langle \mathit{newl}\rangle.\mathbf{0} \\
A &= !(\nu\ x : T_x)\overline{\mathit{spotcheck}}\langle x\rangle.x(y : T_l).y(l_s : \mathsf{Loc}).\mathbf{0} \\
&\quad \mid \mathit{send}(\mathit{fee}).\mathit{fee}(v : \mathsf{Fee}).\mathbf{0} \\
\mathsf{System} &= (\nu\ \mathsf{ETP})(\nu\ \mathit{spotcheck} : T_{sc})(\nu\ \mathit{topa} : T_{pa}) \\
&\quad [\ (\nu\ \mathsf{PA})A \ \mid\ (\nu\ \mathsf{Car})((\nu\ \mathit{read} : T_r)((\nu\ \mathsf{OBE})O \mid (\nu\ \mathsf{GPS})L) \mid (\nu\ \mathsf{SC})S)\ ]
\end{aligned}
$$

We may verify that $\Gamma \vdash \mathsf{System} \triangleright \Theta$, where $\Gamma = \{\mathit{fee} : \mathsf{ETP}[\mathsf{Fee}], \mathit{send} : \mathsf{ETP}[\mathsf{ETP}[\mathsf{Fee}]]\}$ and interface Θ satisfies the enunciated policy.

6 Related work

There exists a large body of literature concerned with formally reasoning about privacy. To begin with, a number of languages have been proposed to express privacy policies [16, 15] and can be used to verify the consistency of policies or to check whether a system complies with a certain policy via static techniques such as model checking [15, 13], on-the-fly using monitoring, or through audit procedures [7, 2].

Related to the privacy properties we are considering in this paper is the notion of Contextual Integrity [2]. Aspects of this notion have been formalized in a logical framework and were used for specifying privacy regulations while notions of compliance of policies by systems were considered. Also related to our work is [17] where a family of models named P-RBAC (Privacy-aware Role Based Access Control) is presented that extends the traditional role-based access control to support specification of complex privacy policies. In particular, the variation thereby introduced called Hierarchical P-RBAC introduces, amongst others, the notion of role hierarchies which is reminiscent

of our policy hierarchies. However, the methodology proposed is mostly geared towards expressing policies and checking for conflicts within policies as opposed to assessing the satisfaction of policies by systems, which is the goal of our work.

Also related to our work is the research line on typed-based security in process calculi. Among these works, numerous studies have focused on access control which is closely related to privacy. For instance the work on the Dπ calculus has introduced sophisticated type systems for controlling the access to distributed resources [11, 12]. Furthermore, discretionary access control has been considered in [4] which similarly to our work employs the π-calculus with groups, while role-based access control (RBAC) has been considered in [3, 9, 6]. In addition, authorization policies and their analysis via type checking has been considered in a number of papers including [10, 1]. While adopting a similar approach, our work departs from these works in the following respects: To begin with we note that role-based access control is insufficient for reasoning about certain privacy violations. While in RBAC it is possible to express that a doctor may read patient's data and send emails, it is not possible to detect the privacy violation breach executed when the doctor sends an email with the sensitive patient data. In our framework, we may control such information dissemination by distinguishing between different types of data and how these can be manipulated. Furthermore, a novelty of our approach is the concept of hierarchies within policies which allow to arrange the system into a hierarchical arrangement of disclosure zones while allowing the inheritance of permissions between groups within the hierarchy.

To conclude, we mention our previous work of [14]. In that work we employed the π-calculus with groups accompanied by a type system based on i/o and linear types for capturing privacy-related notions. In the present work, the type system is reconstructed and simplified using the notions of groups to distinguish between different entities and introducing permissions inference during type checking. Most importantly, a contribution of this work in comparison to [14] is that we introduce a policy language and prove a safety criterion that establishes policy satisfaction by typing.

7 Conclusions

In this paper we have presented a formal framework based on the π-calculus with groups for studying privacy. Our framework is accompanied by a type system for capturing privacy-related notions and a privacy language for expressing privacy policies. We have proved a type preservation theorem and a safety theorem which establishes sufficient conditions for a system to satisfy a policy.

The policy language we have proposed is a simple language that constructs a hierarchical structure of the entities composing a system and assigning permissions for accessing sensitive data to each of the entities while allowing to reason about some simple privacy violations. These permissions are certainly not intended to capture every possible privacy issue, but rather to demonstrate a method of how one might formalize privacy rights. Identifying an appropriate and complete set of permissions for providing foundations for the notion of privacy in the general context should be the result of intensive and probably interdisciplinary research that justifies each choice. To this ef-

fect, Solove's taxonomy of privacy violations forms a promising context in which these efforts can be based and it provides various directions for future work.

Other possible directions for future work can be inspired by privacy approaches such as contextual integrity and P-RBAC. We are currently extending our work to reason about more complex privacy policies that include *conditional* permissions and the concepts of *purpose* and *obligation* as in P-RBAC. Finally, it would be interesting to explore more dynamic settings where the roles evolve over time.

References

1. M. Backes, C. Hritcu, and M. Maffei. Type-checking zero-knowledge. In *Proceedings of CCS'08*, pages 357–370, 2008.
2. A. Barth, A. Datta, J. C. Mitchell, and H. Nissenbaum. Privacy and contextual integrity: Framework and applications. In *Proceedings of S&P'06*, pages 184–198, 2006.
3. C. Braghin, D. Gorla, and V. Sassone. Role-based access control for a distributed calculus. *Journal of Computer Security*, 14(2):113–155, 2006.
4. M. Bugliesi, D. Colazzo, S. Crafa, and D. Macedonio. A type system for discretionary access control. *Mathematical Structures in Computer Science*, 19(4):839–875, 2009.
5. L. Cardelli, G. Ghelli, and A. D. Gordon. Secrecy and group creation. *Information and Computation*, 196(2):127–155, 2005.
6. A. B. Compagnoni, E. L. Gunter, and P. Bidinger. Role-based access control for boxed ambients. *Theoretical Computer Science*, 398(1-3):203–216, 2008.
7. A. Datta, J. Blocki, N. Christin, H. DeYoung, D. Garg, L. Jia, D. K. Kaynar, and A. Sinha. Understanding and protecting privacy: Formal semantics and principled audit mechanisms. In *Proceedings of ICISS'11*, pages 1–27, 2011.
8. W. de Jonge and B. Jacobs. Privacy-friendly electronic traffic pricing via commits. In *Proceedings of FAST'08*, LNCS 5491, pages 143–161. Springer, 2009.
9. M. Dezani-Ciancaglini, S. Ghilezan, S. Jaksic, and J. Pantovic. Types for role-based access control of dynamic web data. In *Proceedings of WFLP'10*, LNCS 6559, pages 1–29. Springer, 2010.
10. C. Fournet, A. Gordon, and S. Maffeis. A type discipline for authorization in distributed systems. In *Proceedings of CSF'07*, pages 31–48, 2007.
11. M. Hennessy, J. Rathke, and N. Yoshida. safedpi: a language for controlling mobile code. *Acta Informatica*, 42(4-5):227–290, 2005.
12. M. Hennessy and J. Riely. Resource access control in systems of mobile agents. *Information and Computation*, 173(1):82–120, 2002.
13. M. Koleini, E. Ritter, and M. Ryan. Model checking agent knowledge in dynamic access control policies. In *TACAS'13*, LNCS 7795, pages 448–462. Springer, 2013.
14. D. Kouzapas and A. Philippou. A typing system for privacy. In *Proceedings of SEFM Workshops 2013*, LNCS 8368, pages 56–68. Springer, 2014.
15. Y. Liu, S. Müller, and K. Xu. A static compliance-checking framework for business process models. *IBM Systems Journal*, 46(2):335–362, 2007.
16. M. J. May, C. A. Gunter, and I. Lee. Privacy APIs: Access control techniques to analyze and verify legal privacy policies. In *Proceedings of CSFW-06*, pages 85–97, 2006.
17. Q. Ni, E. Bertino, J. Lobo, C. Brodie, C. Karat, J. Karat, and A. Trombetta. Privacy-aware role-based access control. *ACM Trans. on Information and System Security*, 13(3), 2010.
18. D. J. Solove. A Taxonomy of Privacy. *University of Pennsylvania Law Review*, 154(3):477–560, 2006.
19. M. C. Tschantz and J. M. Wing. Formal methods for privacy. In *Proceedings of FM'09*, LNCS 5850, pages 1–15. Springer, 2009.