Analysis of Privacy Policies to Enhance Informed Consent

Raúl Pardo, Daniel Le Métayer

Analysis of Privacy Policies to Enhance Informed Consent

Raúl Pardo[1] and Daniel Le Métayer[1]

Univ Lyon, Inria, INSA Lyon, CITI, F-69621 Villeurbanne, France

Abstract. In this paper, we present an approach to enhance informed consent for the processing of personal data. The approach relies on a privacy policy language used to express, compare and analyze privacy policies. We describe a tool that automatically reports the privacy risks associated with a given privacy policy in order to enhance data subjects' awareness and to allow them to make more informed choices. The risk analysis of privacy policies is illustrated with an IoT example.

1 Introduction

One of the most common argument to legitimize the collection of personal data is the fact that the persons concerned have provided their consent or have the possibility to object to the collection. Whether opt-out is considered as an acceptable form of consent (as in the recent California Consumer Privacy Act[1]) or opt-in is required (as in the European General Data Protection Regulation - GDPR[2]), a number of conditions have to be met to ensure that the collection respects the true will of the data subject. In fact, one may argue that this is seldom the case. In practice, internet users generally have to consent on the fly, when they want to use a service, which leads them to accept mechanically the conditions of the provider. Therefore, their consent is not really informed because they do not read the privacy policies of the service providers. In addition, these policies are often vague and ambiguous. This situation, which is already critical, will become even worse with the advent of the internet of things ("IoT") which has the potential to extend to the "real world" the tracking already in place on the internet.

A way forward to address this issue is to allow users to define their own privacy policies, with the time needed to reflect on them, possibly even with the help of experts or pairs. These policies could then be applied automatically to decide upon the disclosure of their personal data and the precise conditions of such disclosures. The main benefit of this approach is to reduce the imbalance of powers between individuals and the organizations collecting their personal data (hereafter, respectively data subjects, or DSs, and data controllers, or DCs, following the GDPR terminology): each party can define her own policy and these policies can then be compared to decide whether a given DC is authorized to collect the personal data of a DS. In practice, DSs can obviously not foresee all possibilities when they define their initial policies and they should

[1] https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375
[2] https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2016.119.01.0001.01.ENG&toc=OJ:L:2016:119:TOC

have the opportunity to update them when they face new types of DCs or new types of purposes for example. Nevertheless, their privacy policies should be able to cope with most situations and, as time passes, their coverage would become ever larger.

However, a language to define privacy policies must meet a number of requirements to be able to express the consent of the DSs. For example, under the GDPR, valid consent must be freely given, specific, informed and unambiguous. Therefore, the language must be endowed with a formal semantics in order to avoid any ambiguity about the meaning of a privacy policy. However, the mere existence of a semantics does not imply that DSs properly understand the meaning of a policy and its potential consequences. One way to enhance the understanding of the DSs is to provide them information about the potential risks related to a privacy policy. This is in line with Recital 39 of the GDPR which stipulates that data subjects should be "made aware of the risks, rules, safeguards and rights in relation to the processing of personal data and how to exercise their rights in relation to such processing". This approach can enhance the awareness of the DSs and allow them to adjust their privacy policies in a better informed way.

A number of languages and frameworks have been proposed in the literature to express privacy policies. However, as discussed in Section 6, none of them meets all the above requirements, especially the strong conditions for valid consent laid down by the GDPR. In this paper, we define a language, called PILOT, meeting these requirements and show its benefits to define precise privacy policies and to highlight the associated privacy risks. Even though PILOT is not restricted to the IoT, the design of the language takes into account the results of previous studies about the expectations and privacy preferences of IoT users [12].

We introduce the language in Section 2 and its abstract execution model in Section 3. Then we show in Section 4 how it can be used to help DSs defining their own privacy policies and understanding the associated privacy risks. Because the language relies on a well-defined execution model, it is possible to reason about privacy risks and to produce (and prove) automatically answers to questions raised by the DSs. In Section 5, we compare PILOT with existing privacy policy languages, and we conclude the paper with avenues for further research in Section 6.

2 The Privacy Policy Language PILOT

In this section we introduce, PILOT, a privacy policy language meeting the objectives set forth in Section 1. The language is designed so that it can be used both by DCs (to define certain aspects of their privacy rules or general terms regarding data protection) and DSs (to express their consent). DCs can also store the DSs policies that they have received for *accountability* purposes—i.e., to be able to demonstrate that data has been treated in accordance with the choices of DSs.

DCs devices must declare their privacy policies before they collect personal data. We refer to these policies as *DC policies*. Likewise, when a DS device sends data to a DC device, the DS device must always include a policy defining the restrictions imposed by the DS on the use of her data by the DC. We refer to these policies as *DS policies*.

In what follows, we formally define the language PILOT. We start with definitions of the most basic elements of PILOT (Section 2.1), which are later used to define the abstract syntax of the language (Section 2.2). This syntax is then illustrated with a working example (Section 2.3).

2.1 Basic definitions

Devices and Entities. We start with a set $\mathcal{D}$ of *devices*. Concretely, we consider devices such as smartphones, laptops or access points, that are able to store, process and communicate data.

Let $\mathcal{E}$ denote the set of *entities* such as Google or Alphabet and $\leq_{\mathcal{E}}$ the associated partial order—e.g., since Google belongs to Alphabet we have $Google \leq_{\mathcal{E}} Alphabet$. Entities include DCs and DSs. Every device is associated with an entity. However, entities may have many devices associated with them. The function `entity` $: \mathcal{D} \rightarrow \mathcal{E}$ defines the entity associated with a given device.

Data Items, datatypes and values. Let $\mathcal{I}$ be a set of *data items*. Data items correspond to the pieces of information that devices communicate. Each data item has a *datatype* associated with it. Let $\mathcal{T}$ be a set of datatypes and $\leq_{\mathcal{T}}$ the associated partial order. We use function `type` $: \mathcal{I} \rightarrow \mathcal{T}$ to define the datatype of each data item. Examples of datatypes[3] are: age, address, city and clinical records. Since $city$ is one of the elements that the datatype $address$ may be composed of, we have $city \leq_{\mathcal{T}} address$. We use $\mathcal{V}$ to the denote the set of all values of data items, $\mathcal{V} = (\bigcup_{t \in \mathcal{T}} \mathcal{V}_t)$ where $\mathcal{V}_t$ is the set of values for data items of type t. We use a special element $\perp \in \mathcal{V}$ to denote the undefined value. A data item may be undefined, for instance, if it has been deleted or it has not been collected. The device where a data item is created (its source) is called the *owner* device of the data item. We use a function `owner`$: \mathcal{I} \rightarrow \mathcal{D}$ to denote the owner device of a given data item.

Purposes. We denote by $\mathcal{P}$ the set of *purposes* and $\leq_{\mathcal{P}}$ the associated partial order. For instance, if newsletter is considered as a specific type of advertisement, then we have $newsletter \leq_{\mathcal{P}} advertisement$.

Conditions. Privacy policies are contextual: they may depend on *conditions* on the information stored on the devices on which they are evaluated. For example, (1) *"Only data from adults may be collected"* or (2) *"Only locations within the city of Lyon may be collected from my smartwatch"* are examples of policy conditions. In order to express conditions we use a simple logical language. Let $\mathcal{F}$ denote a set of functions and *terms* t be defined as follows: $t ::= i \mid c \mid f(\overrightarrow{t})$ where $i \in \mathcal{I}$ is data item, $c \in \mathcal{V}$ is a constant value, $f \in \mathcal{F}$ is a function, and $\overrightarrow{t}$ is a list of terms matching the arity of f. The syntax of the logical language is as follows: $\varphi ::= t_1 * t_2 \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid tt \mid ff$ where $*$ is an arbitrary binary predicate, t_1, t_2 are terms; tt and ff represent respectively true and false. For instance, $age \geq 18$ and $smartwatch\_location = Lyon$ model conditions (1) and (2), respectively. We denote the set of well-formed conditions as $\mathcal{C}$. In order to compare conditions, we use a relation, $\vdash : \mathcal{C} \times \mathcal{C}$. We write $\varphi_1 \vdash \varphi_2$ to denote that φ_2 is stronger than φ_1.

[3] Note that here we do not use the term "datatype" as traditionally in programming languages. We use datatype to refer to the semantic meaning of data items.

2.2 Abstract syntax of PILOT privacy policies

In this section we introduce the abstract syntax of PILOT *privacy policies*, or, simply, PILOT *policies*. We emphasize the fact that this abstract syntax is not the syntax used to communicate with DSs or DCs. This abstract syntax can be associated with a concrete syntax in a restricted form of natural language. We do not describe this mapping here due to space constraints, but we provide some illustrative examples in Section 2.3 and describe a user-friendly interface to define PILOT policies in Section 4.3. The goal of PILOT policies is to express the conditions under which data can be communicated. We consider two different types of data communications: *data collection* and *transfers*. Data collection corresponds to the collection by a DC of information directly from a DS. A transfer is the event of sending previously collected data to third parties.

Definition 1 (PILOT **Privacy Policies Syntax).** *Given* $Purposes \in 2^{\mathcal{P}}$*,* $retention\_time \in \mathbb{N}$*,* $condition \in \mathcal{C}$*,* $entity \in \mathcal{E}$ *and* $datatype \in \mathcal{T}$*, the syntax of* PILOT *policies is defined as follows:*

$$\begin{aligned} Pilot\ Privacy\ Policy &::= (datatype, dcr, TR) \\ Data\ Communication\ Rule\ (dcr) &::= \langle condition, entity, dur\rangle \\ Data\ Usage\ Rule\ (dur) &::= \langle Purposes, retention\_time\rangle \\ Transfer\ Rules\ (TR) &::= \{dcr_1, dcr_2, \ldots\} \end{aligned}$$

We use $\mathcal{DUR}$, $\mathcal{DCR}$, $\mathcal{PP}$ to denote the sets of data usage rules, data communication rules and PILOT privacy policies, respectively. The set of transfer rules is defined as the set of sets of data communication rules, $TR \in 2^{\mathcal{DCR}}$. In what follows, we provide some intuition about this syntax and an example of application.

Data Usage Rules. The purpose of these rules is to define the operations that may be performed on the data. $Purposes$ is the set of allowed purposes and $retention\_time$ the deadline for erasing the data. As an example, consider the following data usage rule,

$$dur_1 = \langle\{research\}, 26/04/2019\rangle.$$

This rule states that the data may be used only for the purpose of research and may be used until $26/04/2019$.

Data Communication Rules. A data communication rule defines the conditions that must be met for the data to be collected by or communicated to an entity. The outer layer of data communication rules — i.e., the condition and entity — should be checked by the sender whereas the data usage rule is to be enforced by the receiver. The first element, $condition$, imposes constraints on the data item and the context (state of the DS device); $entity$ indicates the entity allowed to receive the data; dur is a data usage rule stating how $entity$ may use the data. For example, $\langle age > 18, AdsCom, dur_1\rangle$. states that data may be communicated to the entity AdsCom which may use it according to dur_1 (defined above). It also requires that the data item age is greater than 18. This data item may be the data item to be sent or part of the contextual information of the sender device.

Transfer Rules. These rules form a set of data communication rules specifying the entities to whom the data may be transferred.

PILOT **Privacy Policies.** DSs and DCs use PILOT policies to describe how data may be used, collected and transferred. The first element, $datatype$, indicates the type of data the policy applies to; dcr defines the collection conditions and TR the transfer rules. In some cases, several PILOT policies are necessary to fully capture the privacy choices for a given datatype. For instance, a DS may allow only her employer to collect her data when she is at work but, when being in a museum, she may allow only the museum. In this example, the DS must define two policies, one for each location.

2.3 Example: Vehicle Tracking

In this section, we illustrate the syntax of PILOT with a concrete example that will be continued with the risk analysis in Section 4.

The use of Automatic Number Plate Recognition (ANPR) [10] is becoming very popular for applications such as parking billing or pay-per-use roads. These systems consist of a set of cameras that automatically recognize plate numbers when vehicles cross the range covered by the cameras. Using this information, it is possible to determine how long a car has been in a parking place or how many times it has traveled on a highway, for example.

ANPR systems may collect large amounts of mobility data, which raises privacy concerns [11]. When data is collected for the purpose of billing, the consent of the customer is not needed since the legal ground for the data processing can be the performance of a contract. However, certain privacy regulations, such as the GDPR, require prior consent for the use of the data for other purposes, such as commercial offers or advertisement.

Consider a DC, Parket, which owns parking areas equipped with ANPR in France. Parket is interested in offering discounts to frequent customers. To this end, Parket uses the number plates recorded by the ANPR system to send commercial offers to a selection of customers. Additionally, Parket transfers some data to its sister company, ParketWW, that operates worldwide with the goal of providing better offers to their customers. Using data for these purposes requires explicit consent from DSs. The PILOT policy below precisely captures the way in which Parket wants to collect and use number plates for these purposes.

$$\begin{aligned}(&number\_plate, \langle tt, Parket, \langle \{commercial\_offers\}, 21/03/2019\rangle\rangle,\\ &\qquad\{\langle tt, ParketWW, \langle \{commercial\_offers\}, 26/04/2019\rangle\rangle\})\end{aligned} \tag{1}$$

The condition (tt) in (1) means that Parket does not impose any condition on the number plates it collects or transfers to ParketWW. This policy can be mapped into the following natural language sentence:

> $Parket$ may collect data of type $number\_plate$ and use it for $commercial\_offers$ purposes until $21/03/2019$.
> This data may be transferred to $ParketWW$ which may use it for $commercial\_offers$ purposes until $26/04/2019$.

The parts of the policy in *italic font* correspond to the elements of PILOT's abstract syntax. These elements change based on the content of the policy. The remaining parts of the policy are common to all PILOT policies.

To obtain DSs consent, Parket uses a system which broadcasts the above PILOT policy to vehicles before they enter the ANPR area. The implementation of this broadcast process is outside the scope of this paper; several solutions are presented in [8]. DSs are therefore informed about Parket's policy before data is collected. However, DSs may disagree about the processing of their data for these purposes. They can express their own privacy policy in PILOT to define the conditions of their consent (or denial of consent).

Consider a DS, Alice, who often visits Parket parkings. Alice wants to benefit from the offers that Parket provides in her city (Lyon) but does not want her information to be transferred to third-parties. To this end, she uses the following PILOT policy:

$$\begin{aligned}(\mathit{number\_plate}, \langle \mathit{car\_location} = \mathit{Lyon}, \mathit{Parket}, \\ \langle \{\mathit{commercial\_offers}\}, \mathit{21/03/2019}\rangle\rangle, \emptyset)\end{aligned} \tag{2}$$

In practice, she would actually express this policy as follows:

> *Parket* may collect data of type *number_plate* if *car_location is Lyon* and use it for *commercial_offers* purposes until *21/03/2019*.

which is a natural language version of the above abstract syntax policy.

In contrast with Parket's policy, Alice's policy includes a condition using *car_location*, which is a data item containing the current location of Alice's car. In addition, the absence of transfer statement means that Alice does not allow Parket to transfer her data. It is easy to see that Alice's policy is more restrictive than Parket's policy. Thus, after Alice's device[4] receives Parket's policy, it can automatically send an answer to Parket indicating that Alice does not give her consent to the collection of her data in the conditions stated in *Parket*'s policy. In practice, Alice's policy can also be sent back so that Parket can possibly adjust her own policy to match Alice's requirements. Parket would then have the option to send a new DC policy consistent with Alice's policy and Alice would send her consent in return. The new policy sent by Parket can be computed as a join of Parket's original policy and Alice's policy (see Appendix C for an example of policy join which is proven to preserve the privacy preferences of the DS).

This example is continued in Section 4 which illustrates the use of PILOT to enhance Alice's awareness by providing her information about the risks related to her choices of privacy policy.

3 Abstract execution model

In this section, we describe the abstract execution model of PILOT. The purpose of this abstract model is twofold: it is useful to define a precise semantics of the language and therefore to avoid any ambiguity about the meaning of privacy policies; also, it is

[4] This device can be Alice's car on-board computer, which can itself be connected to the mobile phone used by Alice to manage her privacy policies [8].

used by the verification tool described in the next section to highlight privacy risks. The definition of the full semantics of the language, which is presented in a companion paper [19], is beyond the scope of this paper. In the following, we focus on the two main components of the abstract model: the system state (Section 3.1) and the events (Section 3.2).

3.1 System state

We first present an abstract model of a system composed of devices that communicate information and use PILOT policies to express the privacy requirements of DSs and DCs. Every device has a set of associated policies. A policy is associated with a device if it was defined in the device or the device received it. Additionally, DS devices have a set of data associated with them. These data may represent, for instance, the MAC address of the device or workouts recorded by the device. Finally, we keep track of the data collected by DC devices together with their corresponding PILOT policies. The system state is formally defined as follows.

Definition 2 (System state). *The system state is a triple $\langle \nu, \pi, \rho \rangle$ where:*

- $\nu : \mathcal{D} \times \mathcal{I} \rightharpoonup \mathcal{V}$ *is a mapping from the data items of a device to their corresponding value in that device.*
- $\pi : \mathcal{D} \to 2^{\mathcal{D} \times \mathcal{PP}}$ *is a function denoting the* policy base *of a device. The policy base contains the policies created by the owner of the device and the policies sent by other devices in order to state their collection requirements. A pair (d, p) means that* PILOT *policy p belongs to device d. We write π_d to denote $\pi(d)$.*
- $\rho : \mathcal{D} \to 2^{\mathcal{D} \times \mathcal{I} \times \mathcal{PP}}$ *returns a set of triples (s, i, p) indicating the data items and* PILOT *policies that a controller has received. If $(d', i, p) \in \rho(d)$, we say that device d has received or collected data item i from device d' and policy p describes how the data item must be used. We write ρ_d to denote $\rho(d)$.*

In Definition 2, ν returns the local value of a data item in the specified device. However, not all devices have values for all data items. When the value of a data item in a device is undefined, ν returns $\perp$. The policy base of a device d, π_d, contains the PILOT policies that the device has received or that have been defined locally. If $(d, p) \in \pi(d)$, the policy p corresponds to a policy that d has defined in the device itself. On the other hand, if $(d, p) \in \pi(e)$ where $d \neq e$, p is a policy sent from device e. Policies stored in the policy base are used to compare the privacy policies of two devices before the data is communicated. The information that a device has received is recorded in ρ. Also, ρ contains the PILOT policy describing how data must be used. The difference between policies in π and ρ is that policies in π are used to determine whether data can be communicated, and policies in ρ are used to describe how a data item must be used by the receiver.

Example 1. Fig. 1 shows a state composed of two devices: Alice's car, and Parket's ANPR system. The figure depicts the situation after Alice's car has entered the range covered by the ANPR camera and the collection of her data has already occurred.

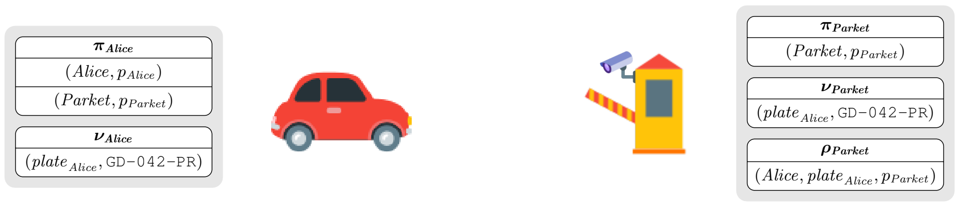


Fig. 1. Example System State

The database in Alice's state (ν_{Alice}) contains a data item of type number plate $plate_{Alice}$ whose value is `GD-042-PR`. The policy base in Alice's device (π_{Alice}) contains two policies: $(Alice, p_{Alice})$ representing a policy that Alice defined, and $(Parket, p_{Parket})$ which represents a policy p_{Parket} sent by Parket. We assume that p_{Alice} and p_{Parket} are the policies applying to data items of type number plate.

Parket's state contains the same components as Alice's state with, in addition, a set of received data (ρ_{Parket}). The latter contains the data item $plate_{Alice}$ collected from Alice and the PILOT policy p_{Parket} that must be applied in order to handle the data. Note that p_{Parket} was the PILOT policy originally defined by Parket. In order for Alice's privacy to be preserved, it must hold that p_{Parket} is more restrictive than Alice's PILOT policy p_{Parket}, which is denoted by $p_{Parket} \sqsubseteq p_{Alice}$.[5] This condition can easily be enforced by comparing the policies before data is collected. The first element in $(Alice, plate_{Alice}, p_{Parket})$ indicates that the data comes from Alice's device. Finally, Parket's policy base has one policy: its own policy p_{Parket}, which was communicated to Alice for data collection. □

3.2 System events

In this section we describe the set of events E in our abstract execution model. We focus on events that ensure that the exchange of data items is done according to the PILOT policies of DSs and DCs.

Events. The set of events E is composed by the following the events: $request$, $send$, $transfer$ and use. The events $request$, $send$ and $transfer$ model valid exchanges of policies and data among DCs and DSs. The event use models correct usage of the collected data by DCs. In what follow we explain each event in detail.

$request(sndr, rcv, t, p)$ models request of data from DCs to DSs or other DCs. Thus, $sndr$ is always a DC device, and rcv may be a DC or DS device. A request includes the type of the data that is being requested t and a PILOT policy p. As expected, the PILOT policy is required to refer to the datatype that is requested, i.e., $p = (t, \_, \_)$. As a result of executing $request$, the pair (rcv, p) is added to π_{rcv}. Thus, rcv is informed of the conditions under which $sndr$ will use the requested data.

[5] See Appendix A for the formal definition of $\sqsubseteq$.

$send(sndr, rcv, i)$ represents the collection by the DC rcv of a data item i from the DS $sndr$. In order for $send$ to be executed, the device $sndr$ must check that π_{sndr} contains: i) an active policy defined by $sndr$, p_{sndr}, indicating how $sndr$ allows DCs to use her data, and ii) an active policy sent by rcv, p_{rcv}, indicating how she plans to use the data. A policy is active if it applies to the data item to be sent, to rcv's entity, the retention time has not yet been reached, and its condition holds.[6] Data can only be sent if p_{rcv} is more restrictive than p_{sndr} (i.e., $p_{rcv} \sqsubseteq p_{sndr}$), which must be checked by $sndr$. We record the data exchange in ρ_{rcv} indicating: the sender, the data item and rcv's PILOT policy, $(sndr, i, p_{rcv})$. We also update rcv's database with the value of i in $sndr$'s state, $\nu(rcv, i) = \nu(sndr, i)$.

$transfer(sndr, rcv, i)$ is executed when a DC $(sndr)$ transfers a data item i to another DC (rcv). First, $sndr$ checks whether π_{sndr} contains an active policy, from rcv, p_{rcv}. Here we do not use a PILOT policy from $sndr$, instead we use the PILOT policy p sent along with the data—defined by the owner of i. Thus, $sndr$ must check whether there exists an active transfer rule (tr) in the set of transfers rules of the PILOT policy p. As before, $sndr$ must check that the policy sent by rcv is more restrictive than those originally sent by the owner of the data, i.e., $p_{rcv} \sqsubseteq p_{tr}$ where p_{tr} is a policy with the active transfer tr in the place of the data communication rule and with the same set of transfers as p. Note that data items can be transferred more than once to the entities in the set of transfers as long as the retention time has not been reached. This is not an issue in terms of privacy as data items are constant values. In the resulting state, we update ρ_{rcv} with the sender, the data item and rcv's PILOT policy, $(sndr, i, p_{rcv})$. Note that, in this case, the owner of the data item is not $sndr$ since transfers always correspond to exchanges of previously collected data, $\texttt{owner}(i) \neq sndr$. The database of rcv is updated with the current value of i in ν_{sndr}.

$use(dev, i, pur)$ models the use of a data item i by a DC device dev for purpose pur. Usage conditions are specified in the data usage rule of the policy attached to the data item, denoted as p_i, in the set of received data of dev, ρ_{dev}. Thus, in order to execute use we require that: i) the purpose pur is allowed by p_i, and ii) the retention time in p_i has not elapsed.

4 Risk Analysis

As described in the introduction, an effective way to enhance informed consent is to raise user awareness about the risks related to personal data collection. Privacy risks may result from different sorts of misbehavior such as the use of data beyond the allowed purpose or the transfer of data to unauthorized third parties [15].

In order to assess the risks related to a given privacy policy, we need to rely on assumptions about potential risk sources, such as:

- Entities e_i that may have a strong interest to use data of type t for a given purpose pur.

[6] See Appendix B for the formal definition of active policy and active transfer.

- Entities e_i that may have facilities and interest to transfer data of type t to other entities e_j.

In practice, some of these assumptions may be generic and could be obtained from databases populated by pairs or NGOs based on history of misconducts by companies or business sectors. Others risk assumptions can be specific to the DS (e.g., if she fears that a friend may be tempted to transfer certain information to another person). Based on these assumptions, a DS who is wondering whether she should add a policy p to her current set of policies can ask questions such as: "if I add this policy p:

- Is there a risk that my data of type t is used for purpose pur?
- Is there a risk that, at some stage, entity e gets my data of type t? "

In what follows, we first introduce our approach to answer the above questions (Section 4.1); then we illustrate it with the example introduced in Section 2.3 (Section 4.2) and we present a user-friendly interface to define and analyze privacy policies (Section 4.3).

4.1 Automatic Risk Analysis with SPIN

In order to automatically answer questions of the type described above, we use the verification tool SPIN [14]. SPIN belongs to the family of verification tools known as *model-checkers*. A model-checker takes as input a model of the system (i.e., an abstract description of the behavior of the system) and a set of properties (typically expressed in formal logic), and checks whether the model of the system satisfies the properties. In SPIN, the model is written in the modeling language PROMELA [14] and properties are encoded in *Linear Temporal Logic* (LTL) (e.g., [4]). We chose SPIN as it has successfully been used in a variety of contexts [18]. However, our methodology is not limited to SPIN and any other formal verification tool such as SMT solvers [5] or automated theorem provers [24] could be used instead.

Our approach consists in defining a PROMELA model for the PILOT events and privacy policies, and translating the risk analysis questions into LTL properties that can be automatically checked by SPIN. For example, the question *"Is there a risk that Alice's data is used for the purpose of profiling by ParketWW?"* is translated into the LTL property *"ParketWW never uses Alice's data for profiling"*. Devices are modeled as processes that randomly try to execute events defined as set forth in Section 3.2.

In order to encode the misbehavior expressed in the assumptions, we add "illegal" events to the set of events that devices can execute. For instance, consider the assumption *"use of data beyond the allowed purpose"*. To model this assumption, we introduce the event $illegal\_use$, which behaves as use, but disregards the purpose of the DS policy for the data.

SPIN explores all possible sequences of executions of events (including misbehavior events) trying to find a sequence that violates the LTL property. If no sequence is found, the property cannot be violated, which means that the risk corresponding to the property cannot occur. If a sequence is found, the risk corresponding to the property can occur, and SPIN returns the sequence of events that leads to the violation. This sequence of events can be used to further clarify the cause of the violation.

4.2 Case Study: Vehicle Tracking

We illustrate our risk analysis technique with the vehicle tracking example introduced in Section 2.3. We first define the PROMELA model and the assumptions on the entities involved in this example. The code of the complete model is available in [23].

Promela Model. We define a model involving the three entities identified in Section 2.3 with, in addition, the car insurance company $CarInsure$ which is identified as a potential source of risk related to $ParketWW$, i.e., $\mathcal{E} = \{Alice, Parket, ParketWW, CarInsure\}$. Each entity is associated with a single device: $\mathcal{D} = \mathcal{E}$ and $\texttt{entity}(x) = x$ for $x \in \{Alice, Parket, ParketWW, CarInsure\}$. We focus on one datatype $\mathcal{T} = \{number\_plate\}$ with its set of values defined as $\mathcal{V}_{number\_plate} = \{\texttt{GD-042-PR}\}$. We consider a data item $plate_{Alice}$ of type $number\_plate$ for which $Alice$ is the owner. Finally, we consider a set of purposes $\mathcal{P} = \{commercial\_offers, profiling\}$.

Risk assumptions on entities. In this case study, we consider two risk assumptions:

1. ParketWW may transfer personal data to CarInsure disregarding the associated DS privacy policies.
2. CarInsure has strong interest in using personal data for profiling.

In practice, these assumptions, which are not specific to Alice, may be obtained automatically from databases populated by pairs or NGOs for example.

Set of events. The set of events that we consider is derived from the risk assumptions on entities. On the one hand, we model events that behave correctly, i.e., as described in Section 3.2. In order to model the worst case scenario in terms of risk analysis, we consider that: the DCs in this case study (i.e., Parket, ParketWW and CarInsure) can request data to any entity (including Alice), the DCs can collect Alice's data, and the DCs can transfer data among them. On the other hand, the risk assumptions above are modeled as two events: ParketWW may transfer data to CarInsure disregarding Alice's policy, and CarInsure may use Alice's data for profiling even if it is not allowed by Alice's policy. Let $DC, DC' \in \{Parket, ParketWW, CarInsure\}$, the following events may occur:

– $request(DC, Alice, number\_plate, p)$ - A DC requests a number plate from Alice and p is the PILOT policy of the DC.
– $request(DC, DC', number\_plate, p)$ - A DC requests data items of type number plate from another DC and p is the PILOT policy of the requester DC.
– $send(Alice, DC, i)$ - Alice sends her item i to a DC.
– $transfer(DC, DC', i)$ - A DC transfers a previously received item i to another DC.
– $illegal\_transfer(ParketWW, CarInsure, i)$ - ParketWW transfers a previously received item i to CarInsure disregarding the associated PILOT policy defined by the owner of i.
– $illegal\_use(CarInsure, i, profiling)$ - CarInsure uses data item i for profiling disregarding the associated privacy policy defined by the owner of i.

Alice's policies. In order to illustrate the benefits of our risk analysis approach, we focus on the following two policies that Alice may consider.

$$\begin{aligned} p\_trans_{Alice} = (&number\_plate, \langle tt, Parket, \langle \{commercial\_offers\}, 21/03/2019\rangle\rangle, \\ &\{\langle tt, ParketWW, \langle \{commercial\_offers\}, 26/04/2019\rangle\rangle\}). \end{aligned}$$

$$p\_no\_trans_{Alice} = (number\_plate, \langle tt, Parket, \langle \{commercial\_offers\}, 21/03/2019\rangle\rangle, \emptyset).$$

The policy p_trans_{Alice} states that Parket can collect data of type number plate from Alice, use it for commercial offers and keep it until $21/03/2019$. It also allows Parket to transfer the data to ParketWW. ParketWW may use the data for commercial offers and keep it until $26/04/2019$. The policy $p\_no\_trans_{Alice}$ is similar to p_trans_{Alice} except that it does not allow Parket to transfer the data. We assume that Alice has not yet defined any other privacy policy concerning Parket and ParketWW.

Parket's policy. We set Parket's privacy policy equal to Alice's. By doing so, we consider the worst case scenario in terms of privacy risks because it allows Parket to collect Alice's data and use it in all conditions and for all purposes allowed by Alice.

ParketWW's policy. Similarly, ParketWW's policy is aligned with the transfer rule in p_trans_{Alice}:

$$p_{ParketWW} = (number\_plate, \langle tt, ParketWW, \langle \{commercial\_offers\}, 26/04/2019\rangle\rangle, \emptyset).$$

The above policy states that ParketWW may use data of type number plate for commercial offers and keep it until $26/04/2019$. It also represents the worst case scenario for risk analysis, as it matches the preferences in Alice's first policy.

Results of the Risk Analysis

Table 1 summarizes some of the results of the application of our SPIN risk analyzer on this example. The questions in the first column have been translated into LTL properties used by SPIN (see [23]). The output of SPIN appears in columns 2 to 5. The green boxes indicate that the output is in accordance with Alice's policy while red boxes correspond to violations of her policy.

Columns 2 and 3 correspond to executions of the system involving correct events, considering respectively p_trans_{Alice} and $p\_no\_trans_{Alice}$ as Alice's policy. As expected, all these executions respect Alice's policies.

Columns 3 and 4 consider executions involving $illegal\_transfer$ and $illegal\_use$. These columns show the privacy risks taken by Alice based on the above risk assumptions. Rows 3 and 6 show respectively that CarInsure may get Alice's data and use it for profiling. In addition, the counterexamples generated by SPIN, which are not pictured in the table, show that this can happen only after ParketWW executes $illegal\_transfer$.

From the results of this privacy risk analysis Alice may take a better informed decision about the policy to choose. In a nutshell, she has three options:

1. Disallow Parket to use her data for commercial offers, i.e., choose to add neither p_trans_{Alice} nor $p\_no\_trans_{Alice}$ to her set of policies (Parket will use the data only for billing purposes, based on contract).

Question	Normal behavior		Misbehavior Assumptions	
	p_trans_{Alice}	$p\_no\_trans_{Alice}$	p_trans_{Alice}	$p\_no\_trans_{Alice}$
Can Parket receive Alice's data?	🟩 Yes	🟩 Yes	🟩 Yes	🟩 Yes
Can ParketWW receive Alice's data?	🟩 Yes	🟩 No	🟩 Yes	🟩 No
Can CarInsure receive Alice's data?	🟩 No	🟩 No	🟥 Yes	🟩 No
Can Parket use Alice's data for other purpose than commercial offers?	🟩 No	🟩 No	🟩 No	🟩 No
Can ParketWW use Alice's data for other purpose than commercial offers?	🟩 No	🟩 No	🟩 No	🟩 No
Can CarInsure use Alice's data for profiling?	🟩 No	🟩 No	🟥 Yes	🟩 No

Table 1. Risk Analysis of Alice's policies p_trans_{Alice} and $p\_no\_trans_{Alice}$. Red boxes denote that Alice's policy is violated. Green boxes denote that Alice's policy is respected.

2. Allow Parket to use her data for commercial offers without transfers to ParketWW, i.e., choose $p\_no\_trans_{Alice}$.
3. Allow Parket to use her data for commercial offers and to transfer to ParketWW, i.e., choose p_trans_{Alice}.

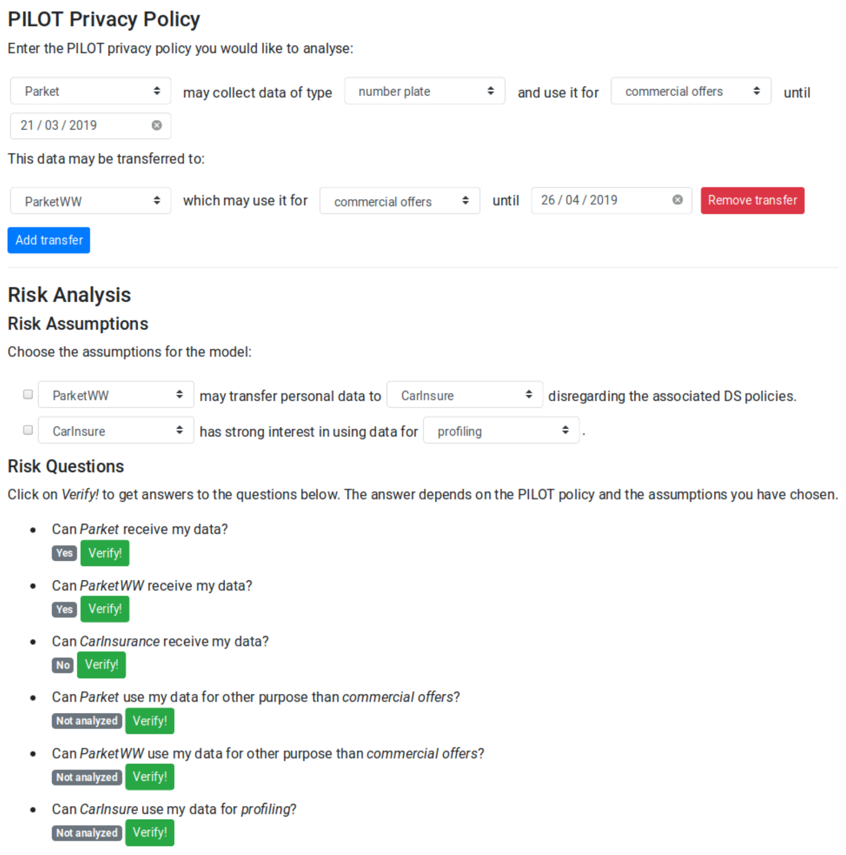
Therefore, if Alice wants to receive commercial offers but does not want to take the risk of being profiled by an insurance company, she should take option two.

4.3 Usability

In order to show the usability of the approach, we have developed a web application to make it possible for users with no technical background to perform risk analysis as outlined in Section 4.2 for the ANPR system.

Fig. 2 shows the input forms of the web application. First, DSs have access to a user-friendly form to input PILOT policies. In the figure we show an example for the policy p_trans_{Alice}. Then DSs can choose the appropriate risk assumptions from the list generated by the system. Finally, they can ask questions about the potential risks based on these assumptions. When clicking on "Verify!", the web application runs SPIN to verify the LTL property corresponding to the question. The text "Not Analyzed" in grey is updated with "Yes" or "No" depending on the result. The figure shows the results of the three first questions with p_trans_{Alice} and no risk assumption chosen (first column in Table 1).

The web application is tailored to the ANPR case study we use throughout the paper. The PROMELA model and the policies defined in Section 4.2 are implemented in the application. This prototype can be generalized in different directions, for example by allowing users to enter specific risk assumptions on third parties. The range of questions



PILOT Privacy Policy

Enter the PILOT privacy policy you would like to analyse:

Parket may collect data of type number plate and use it for commercial offers until 21 / 03 / 2019

This data may be transferred to:

ParketWW which may use it for commercial offers until 26 / 04 / 2019

Risk Analysis

Risk Assumptions

Choose the assumptions for the model:

ParketWW may transfer personal data to CarInsure disregarding the associated DS policies.

CarInsure has strong interest in using data for profiling .

Risk Questions

Click on *Verify!* to get answers to the questions below. The answer depends on the PILOT policy and the assumptions you have chosen.

- Can *Parket* receive my data? Yes Verify!
- Can *ParketWW* receive my data? Yes Verify!
- Can *CarInsurance* receive my data? No Verify!
- Can *Parket* use my data for other purpose than *commercial offers*? Not analyzed Verify!
- Can *ParketWW* use my data for other purpose than *commercial offers*? Not analyzed Verify!
- Can *CarInsure* use my data for *profiling*? Not analyzed Verify!

Fig. 2. Input Forms of Risk Analysis Web Application.

could also be extended to include questions such as “Can X use Y’s data for other purpose than pur? The code of the web application is available at [23].

5 Related Work

Several languages or frameworks dedicated to privacy policies have been proposed. A pioneer project in this area was the “Platform for Privacy Preferences” (P3P) [22]. P3P makes it possible to express notions such as purpose, retention time and conditions. However, P3P is not really well suited to the IoT as it was conceived as a policy language for websites. Also, P3P does not offer support for defining data transfers. Other languages close to P3P have been proposed, such as the “Enterprise Policy Authorization Language” (EPAL) [2] and “An Accountability Policy Language” (A-PPL) [3]. The lack of a precise execution model for these languages may also give rise to ambiguities and variations in their implementations.

Even if its first target was the interactions with service providers rather than IoT environments, the language that is the closest to the spirit of PILOT is the Data Handling Policy (DHP) language [1]. DHP also allows users to express the actions and purposes that are authorized for (specific or generic) recipients. DHP does not include explicitly transfer rules and retention time but, in contrast to PILOT, it makes it possible to specify obligations. Obligations can be used, for example, to require the deletion of data after a given period of time.

None of the above works include tools to help users understand the privacy risks associated with a given a policy, which is a major benefit of PILOT as discussed in Section 4. In the same spirit, Joyee De *et al.* [16] have proposed a methodology where DSs can visualize the privacy risks associated to their privacy settings. Here the authors use harm trees to determine the risks associated with privacy settings. The main difference with PILOT is that harm trees must be manually defined for a given application whereas we our analysis is fully automatic.[7]

Another line of work is that of formal privacy languages. Languages such as S4P [7] and SIMPL [17] define unambiguously the behavior of the system—and, consequently, the meaning of the policies—by means of trace semantics. The goal of this formal semantics is to be able to prove global correctness properties such as "DCs always use DS data according to their policies". While this semantics is well-suited for its intended purpose, it cannot be directly used to develop policy enforcement mechanisms. In contrast, we provide a PROMELA model in Section 4—capturing the execution model of PILOT (cf. Section 3)—that can be used as a reference to implement a system for the enforcement for PILOT policies. In addition, these languages, which were proposed before the adoption of the GDPR, were not conceived with its requirements in mind.

Other languages have been proposed to specify privacy regulations such as HIPAA, COPAA and GLBA. For instance, CI [6] is a dedicated linear temporal logic based on the notion of contextual integrity. CI has been used to model certain aspects of regulations such as HIPAA, COPPA and GLBA. Similarly, PrivacyAPI [18] is an extension of the access control matrix with operations such as notification and logging. The authors also use a PROMELA model of HIPAA to be able to verify the "correctness" and better understand the regulation. PrivacyLFP [9] uses first-order fixed point logic to increase the expressiveness of previous approaches. Using PrivacyLFP, the authors formalize HIPAA and GLBA with a higher degree of coverage than previous approaches. The main difference between PILOT and these languages is their focus. PILOT is focused on modeling DSs and DCs privacy policies and enhancing DSs awareness whereas these languages focus on modeling regulations.

Usage control (UCON) [20,21] appeared as an extension of access control to express how the data may be used after being accessed. To this end, it introduces *obligations*, which are actions such as "do not transfer data item i". The Obligation Specification Language (OSL) [13] is an example of enforcement mechanism through digital right management systems. However, UCON does not offer any support to compare policies and does not differentiate between DSs and DCs policies, which is a critical feature in the context of privacy policies. For DSs to provide an informed consent, they should know whether DCs policies comply with their own policies.

[7] Only risk assumptions must be defined, which is useful to answer different "what-if" questions.

Some work has also been done on privacy risk analysis [15], in particular to address the needs of the GDPR regarding Privacy Impact Assessments. We should emphasize that the notion of risk analysis used in this paper is different in the sense that it applies to potential risks related to privacy policies rather than systems or products. Hence, the risk assumptions considered here concern only the motivation, reputation and potential history of misbehavior of the parties (but not the vulnerabilities of the systems, which are out of reach and expertise of the data subjects).

6 Conclusion

In this paper, we have presented the privacy policy language PILOT, and a novel approach to analyzing privacy policies which is focused on enhancing informed consent. An advantage of a language like PILOT is the possibility to use it as a basis to implement "personal data managers", to enforce privacy policies automatically, or "personal data auditors", to check a posteriori that a DC has complied with the DS policies associated with all the personal data that it has processed. Another orthogonal challenge in the context of the IoT is to ensure that DSs are always informed about the data collection taking place in their environment and can effectively communicate their consent (or objection) to the surrounding sensors. Different solutions to this problem have been proposed in [8] relying on PILOT as a privacy policy language used by DCs to communicate their policies and DSs to provide their consent. These communications can either take place directly or indirectly (through registers in which privacy policies can be stored).

The work described in this paper can be extended in several directions. First, the risk analysis model used here is simple and could be enriched in different ways, for example by taking into account risks of inferences between different types of data. The evaluation of these risks could be based on past experience and research such as the study conducted by Privacy International.[8] The risk analysis could also involve the history of the DS (personal data already collected by DCs in the past). On the formal side, our objective is to use a formal theorem prover to prove global properties of the model. This formal framework could also be used to implement tools to verify that a given enforcement system complies with the PILOT policies.

Acknowledgments

This work has been partially funded by the ANR project CISC (Certification of IoT Secure Compilation) and by the Inria Project Lab SPAI.

References

1. Ardagna, C.A., De Capitani di Vimercati, S., Samarati, P.: Enhancing user privacy through data handling policies. In: Damiani, E., Liu, P. (eds.) Data and Applications Security XX. pp. 224–236. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)

[8] https://privacyinternational.org/sites/default/files/2018-04/data%20points%20used%20in%20tracking_0.pdf

2. Ashley, P., Hada, S., Karjoth, G., Powers, C., Schunter, M.: Enterprise Privacy Authorization Language (EPAL). IBM Research (2003)
3. Azraoui, M., Elkhiyaoui, K., Önen, M., Bernsmed, K., de Oliveira, A.S., Sendor, J.: A-PPL: An Accountability Policy Language. In: Data Privacy Management, Autonomous Spontaneous Security, and Security Assurance - 9th International Workshop, DPM 2014, 7th International Workshop, SETOP 2014, and 3rd International Workshop, QASA 2014, Revised Selected Papers. Lecture Notes in Computer Science, vol. 8872, pp. 319–326 (2014)
4. Baier, C., Katoen, J.: Principles of Model Checking. MIT Press (2008)
5. Barrett, C., Tinelli, C.: Satisfiability Modulo Theories, pp. 305–343. Springer International Publishing (2018)
6. Barth, A., Datta, A., Mitchell, J.C., Nissenbaum, H.: Privacy and Contextual Integrity: Framework and Applications. In: Proceedings of the 27th IEEE Symposium on Security and Privacy, S&P'06. pp. 184–198 (2006)
7. Becker, M., Malkis, A., Bussard, L.: S4P: A Generic Language for Specifying Privacy Preferences and Policies. Research report, Microsoft Research (2010)
8. Cunche, M., Le Métayer, D., Morel, V.: A Generic Information and Consent Framework for the IoT. Research Report RR-9234, Inria (2018), https://hal.inria.fr/hal-01953052
9. DeYoung, H., Garg, D., Jia, L., Kaynar, D.K., Datta, A.: Experiences in the Logical Specification of the HIPAA and GLBA Privacy Laws. In: Proceedings of the 2010 ACM Workshop on Privacy in the Electronic Society, WPES'10. pp. 73–82 (2010)
10. Du, S., Ibrahim, M., Shehata, M.S., Badawy, W.M.: Automatic License Plate Recognition (ALPR): A State-of-the-Art Review. IEEE Trans. Circuits Syst. Video Techn. **23**(2), 311–325 (2013)
11. Electronic Fountrier Foundatino (EFF): Automated License Plate Readers (ALPR) (2017), https://www.eff.org/cases/automated-license-plate-readers
12. Emami-Naeini, P., Bhagavatula, S., Habib, H., Degeling, M., Bauer, L., Cranor, L.F., Sadeh, N.M.: Privacy Expectations and Preferences in an IoT World. In: Proceedings of the 13th Symposium on Usable Privacy and Security, SOUPS'17. pp. 399–412 (2017)
13. Hilty, M., Pretschner, A., Basin, D.A., Schaefer, C., Walter, T.: A Policy Language for Distributed Usage Control. In: Proceedings of the 12th European Symposium On Research in Computer Security, ESORICS'07. Lecture Notes in Computer Science, vol. 4734, pp. 531–546. Springer (2007)
14. Holzmann, G.J.: The SPIN Model Checker - Primer and Reference Manual. Addison-Wesley (2004)
15. Joyee De, S., Le Métayer, D.: Privacy Risk Analysis. Morgan & Claypool Publishers (2016)
16. Joyee De, S., Le Métayer, D.: Privacy Risk Analysis to Enable Informed Privacy Settings. In: 2018 IEEE European Symposium on Security and Privacy, Workshops, EuroS&P Workshops. pp. 95–102 (2018)
17. Le Métayer, D.: A formal privacy management framework. In: Proceedings of the 5th International Workshop on Formal Aspects in Security and Trust, FAST'08, Revised Selected Papers. LNCS, vol. 5491, pp. 162–176. Springer (2008)
18. May, M.J., Gunter, C.A., Lee, I.: Privacy APIs: Access Control Techniques to Analyze and Verify Legal Privacy Policies. In: Proceedings of the 19th IEEE Computer Security Foundations Workshop, CSFW'06. pp. 85–97. IEEE Computer Society (2006)
19. Pardo, R., Le Métayer, D.: Formal Verification of Legal Privacy Requirements (Submitted for publication.)
20. Park, J., Sandhu, R.S.: The UCON_{ABC} Usage Control Model. ACM Trans. Inf. Syst. Secur. **7**(1), 128–174 (2004)
21. Pretschner, A., Hilty, M., Basin, D.A.: Distributed Usage Control. Commun. ACM **49**(9), 39–44 (2006)

22. Reagle, J., Cranor, L.F.: The Platform for Privacy Preferences. Commun. ACM **42**(2), 48–55 (1999)
23. PILOT Risk Analysis Model: https://github.com/raulpardo/pilot-risk-analysis-model.
24. Robinson, A.J., Voronkov, A.: Handbook of Automated Reasoning, vol. 1 & 2. Elsevier (2001)

Appendix

A Policy subsumption

We formalize the notion of *policy subsumption* as a relation over PILOT policies. We start by defining subsumption of data usage and data communication rules, which is used to define PILOT policy subsumption.

Definition 3 (Data Usage Rule Subsumption). *Given two data usage rules* $dur_1 = \langle P_1, rt_1 \rangle$ *and* $dur_2 = \langle P_2, rt_2 \rangle$*, we say that* dur_1 subsumes dur_2*, denoted as* $dur_1 \preceq_{\mathcal{DUR}} dur_2$*, iff i)* $\forall p_1 \in P_1 \cdot \exists p_2 \in P_2$ *such that* $p_1 \leq_{\mathcal{P}} p_2$*; and ii)* $rt_1 \leq rt_2$*.*

Definition 4 (Data Communication Rule Subsumption). *Given two data communication rules* $dcr_1 = \langle c_1, e_1, dur_1 \rangle$ *and* $dcr_2 = \langle c_2, e_2, dur_2 \rangle$*, we say that* dcr_1 subsumes dcr_2*, denoted as* $dcr_1 \preceq_{\mathcal{DCR}} dcr_2$*, iff i)* $c_1 \vdash c_2$*; ii)* $e_1 \leq_{\mathcal{E}} e_2$*; and iii)* $dur_1 \preceq_{\mathcal{DUR}} dur_2$*.*

Definition 5 (PILOT Privacy Policy Subsumption). *Given two* PILOT *privacy policies* $\pi_1 = \langle t_1, dcr_1, TR_1 \rangle$ *and* $\pi_2 = \langle t_2, dcr_2, TR_2 \rangle$*, we say that* π_1 subsumes π_2*, denoted as* $\pi_1 \sqsubseteq \pi_2$ *iff i)* $t_1 \leq_{\mathcal{T}} t_2$*; ii)* $dcr_1 \preceq_{\mathcal{DCR}} dcr_2$*; and iii)* $\forall tr_1 \in TR_1 \cdot \exists tr_2 \in TR_2$ *such that* $tr_1 \preceq_{\mathcal{DCR}} tr_2$*.*

B Active Policies and Transfer rules

Here we formally define when PILOT policies and transfer rules are active. Let $\texttt{eval}(\nu, d, \varphi)$ denote an *evaluation function* for conditions. $\texttt{eval}(\nu, d, \varphi)$ is defined as described in Table 2. We use a function $\texttt{time}(e) : E \rightarrow \mathbb{N}$ to assign a timestamp—represented as a natural number $\mathbb{N}$—to each event of a trace.

Active policy. Formally, $\texttt{activePolicy}(p, \mathit{send}(\mathit{sndr}, \mathit{rcv}, i), \mathit{st}) = \texttt{type}(i) \leq_{\mathcal{T}} t \land \texttt{eval}(\nu, \mathit{sndr}, \varphi) \land \texttt{time}(\mathit{st}, \mathit{send}(\mathit{sndr}, \mathit{rcv}, i)) < \mathit{rt} \land \texttt{entity}(\mathit{rcv}) \leq_{\mathcal{E}} e$ where $p = (t, \langle \varphi, e, \langle \_, \mathit{rt} \rangle \rangle, \_)$ and $\mathit{st} = \langle \nu, \_, \_ \rangle$. Intuitively, given $p = (t, \langle \varphi, e, \langle P, \mathit{rt} \rangle \rangle, \mathit{TR})$, we check that: i) the type of the data to be sent corresponds to the type of data the policy is defined for ($\texttt{type}(i) \leq_{\mathcal{T}} t$); ii) the condition of the policy evaluates to true ($\texttt{eval}(\nu, \mathit{sndr}, \varphi)$); iii) the retention time for the receiver has not expired ($\texttt{time}(\mathit{send}(\mathit{sndr}, \mathit{rcv}, i)) < \mathit{rt}$); and iv) the entity associated with the receiver device is allowed by the policy ($\texttt{entity}(\mathit{rcv}) \leq_{\mathcal{E}} e$).

Active transfer rule. In order for a transfer rule to be active, the above checks are performed on the transfer rule tr, and, additionally, it is required that the retention time for the sender has not elapsed ($\texttt{time}(\mathit{transfer}(\mathit{sndr}, \mathit{rcv}, i)) < \mathit{rt}$).

$$
\begin{aligned}
\mathtt{eval}(\nu, d, tt) &= \mathtt{true}\\
\mathtt{eval}(\nu, d, f\!f) &= \mathtt{false}\\
\mathtt{eval}(\nu, d, i) &= \nu(d, i)\\
\mathtt{eval}(\nu, d, c) &= \hat{c}\\
\mathtt{eval}(\nu, d, f(t_1, t_2, \ldots)) &= \hat{f}(\mathtt{eval}(\nu, d, t_1), \mathtt{eval}(\nu, d, t_2), \ldots)\\
\mathtt{eval}(\nu, d, t_1 * t_2) &= \begin{cases} \mathtt{eval}(\nu, d, t_1) \mathrel{\hat{*}} \mathtt{eval}(\nu, d, t_2) \text{ if } \mathtt{eval}(\nu, d, t_i) \neq \bot \\ \bot \text{ otherwise} \end{cases}\\
\mathtt{eval}(\nu, d, \varphi_1 \wedge \varphi_2) &= \begin{cases} \mathtt{eval}(\nu, d, \varphi_1) \text{ and } \mathtt{eval}(\nu, d, \varphi_2) \text{ if } \mathtt{eval}(\nu, d, \varphi_i) \neq \bot \\ \bot \text{ otherwise} \end{cases}\\
\mathtt{eval}(\nu, d, \neg\varphi) &= \begin{cases} \text{not } \mathtt{eval}(\nu, d, \varphi) \text{ if } \mathtt{eval}(\nu, d, \varphi) \neq \bot \\ \bot \text{ otherwise} \end{cases}
\end{aligned}
$$

Table 2. Definition of $\mathtt{eval}(\nu, d, \varphi)$. We use $\hat{c}$, $\hat{f}$ and $\hat{*}$ to denote the interpretation of constants, functions and binary predicates, respectively.

C Policy join

We present a *join operator* for PILOT policies and prove that the resulting policy is more restrictive than the policies used to compute the join. We first define join operators for data usage rules and data communication rules, and use them to the join operator for PILOT policies. Let $min(e, e')$ be a function that, given two elements $e, e' \in \mathcal{X}$ returns the minimum in the corresponding partial order $\leq_{\mathcal{X}}$. Let $\Cap$ denote the intersection keeping the minimum of comparable elements in the partial order of purposes. Formally, given $P, P' \in \mathcal{P}$, $P \Cap P' \triangleq (P \cap P') \cup P''$ where $P'' = \{p \in P \mid \exists p' \in P' \text{ s.t. } p < p'\}$.

Definition 6 (Data Usage Rule Join). *Given two data usage rules $dur_1 = \langle P_1, rt_1\rangle$ and $dur_2 = \langle P_2, rt_2\rangle$, the data usage rule join operator is defined as: $dur_1 \sqcup_{\mathcal{DUR}} dur_2 = \langle P_1 \Cap P_2, min(rt_1, rt_2)\rangle$.*

Definition 7 (Data Communication Rule Join). *Given two data communication rules $dcr_1 = \langle c_1, e_1, dur_1\rangle$ and $dcr_2 = \langle c_2, e_2, dur_2\rangle$, the data communication rule join operator is defined as: $dur_1 \sqcup_{\mathcal{DCR}} dur_2 = \langle c_1 \wedge c_2, min(e_1, e_2), dur_1 \sqcup_{\mathcal{DUR}} dur_2\rangle$.*

Definition 8 (PILOT Policy Join $\sqcup$). *Given two PILOT policies $p = (t_1, dcr_1, TR_1)$ and $q = (t_2, dcr_2, TR_2)$, the policy join operator is defined as: $dur_1 \sqcup_{\mathcal{DCR}} dur_2 = (min(t_1, t_2), dcr_1 \sqcup_{\mathcal{DCR}} dcr_2, \{t \sqcup_{\mathcal{DCR}} t' \mid t \in TR_1 \ \wedge\ t' \in TR_2 \ \wedge\ t \preceq_{\mathcal{DCR}} t' \})$.*

We say that an join operation is privacy preserving if the resulting policy is more restrictive than both operands. Formally,

Definition 9 (Privacy Preserving Join). *We say that $\sqcup$ is* privacy preserving *iff $\forall p, q \in \mathcal{PP}.(p \sqcup q) \sqsubseteq p \wedge (p \sqcup q) \sqsubseteq q$.*

In what follows we prove that the operation $\sqcup$ is privacy preserving, Lemma 3.

Lemma 1. *Given two data usage rules $dur_1, dur_2 \in \mathcal{DUR}$ it holds that $dur_1 \sqcup_{\mathcal{DUR}} dur_2 \preceq_{\mathcal{DUR}} dur_1$ and $dur_1 \sqcup_{\mathcal{DUR}} dur_2 \preceq_{\mathcal{DUR}} dur_2$.*

Proof. We split the proof into the two conjuncts of Lemma 1.

$\underline{dur_1 \sqcup_{\mathcal{DUR}} dur_2 \preceq_{\mathcal{DUR}} dur_1}$ - We split the proof into the elements of data usage rules, i.e., purposes and retention time.

- We show that $\forall p \in dur_1.P \Cap dur_2.P \cdot \exists p' \in dur_1.P$ such that $p \leq_{\mathcal{P}} p'$.

- $\forall p \in [(dur_1.P \cap dur_2.P) \cup \{p_1 \in dur_1.P \mid \exists p_2 \in dur_2.P$ s.t. $p_1 \leq_{\mathcal{P}} p_2\}] \cdot \exists p' \in dur_1.P$ such that $p \leq_{\mathcal{P}} p'$ [By Def. $\Cap$]
- We split the proof for each operand in the union.
  - ∗ We show that $\forall p \in (dur_1.P \cap dur_2.P) \cdot \exists p' \in dur_1.P$ such that $p \leq_{\mathcal{P}} p'$. Assume $p \in (dur_1.P \cap dur_2.P)$. Then $\exists p' \in dur_1.P$ s.t. $p = p'$ [By Def. $\cap$]. Therefore, $p \leq_{\mathcal{P}} p'$.
  - ∗ We show that $\forall p \in \{p_1 \in dur_1.P \mid \exists p_2 \in dur_2.P$ s.t. $p_1 \leq_{\mathcal{P}} p_2\} \cdot \exists p' \in dur_1.P$ such that $p \leq_{\mathcal{P}} p'$. Assume $p \in \{p_1 \in dur_1.P \mid \exists p_2 \in dur_2.P$ s.t. $p_1 \leq_{\mathcal{P}} p_2\}$. Then, $p \in dur_1.P$, and, consequently, $\exists p' \in dur_1.P$ s.t. $p \leq_{\mathcal{P}} p'$.

– $min(dur_1.rt, dur_2.rt) \leq dur_1.rt$ [By Def. min]

$\underline{dur_1 \sqcup_{\mathcal{DUR}} dur_2 \preceq_{\mathcal{DUR}} dur_2}$ - Retention time is symmetric to the previous case, therefore we only show purposes.

– We show that $\forall p \in dur_1.P \Cap dur_2.P \cdot \exists p' \in dur_2.P$ such that $p \leq_{\mathcal{P}} p'$.
- $\forall p \in [(dur_1.P \cap dur_2.P) \cup \{p_1 \in dur_1.P \mid \exists p_2 \in dur_2.P$ s.t. $p_1 \leq_{\mathcal{P}} p_2\}] \cdot \exists p' \in dur_2.P$ such that $p \leq_{\mathcal{P}} p'$ [By Def. $\Cap$]
- We split the proof for each operand in the union.
  - ∗ The case $\forall p \in (dur_1.P \cap dur_2.P) \cdot \exists p' \in dur_2.P$ such that $p \leq_{\mathcal{P}} p'$ is symmetric to the case above.
  - ∗ We show that $\forall p \in \{p_1 \in dur_1.P \mid \exists p_2 \in dur_2.P$ s.t. $p_1 \leq_{\mathcal{P}} p_2\} \cdot \exists p' \in dur_2.P$ such that $p \leq_{\mathcal{P}} p'$. Assume $p \in \{p_1 \in dur_1.P \mid \exists p_2 \in dur_2.P$ s.t. $p_1 \leq_{\mathcal{P}} p_2\}$. Then $\exists p_2 \in dur_2.P$ s.t. $p \leq_{\mathcal{P}} p_2$, and, consequently, $\exists p' \in dur_2.P$ s.t. $p \leq_{\mathcal{P}} p'$. □

Lemma 2. *Given two data communication rules $dcr_1, dcr_2 \in \mathcal{DCR}$ it holds that $dcr_1 \sqcup_{\mathcal{DUR}} dcr_2 \preceq_{\mathcal{DUR}} dcr_1$ and $dcr_1 \sqcup_{\mathcal{DCR}} dcr_2 \preceq_{\mathcal{DCR}} dcr_2$.*

Proof. We split the proof into the two conjuncts of Lemma 2.

$\underline{dcr_1 \sqcup_{\mathcal{DUR}} dcr_2 \preceq_{\mathcal{DUR}} dcr_1}$ - We split the proof into the elements of data communication rules, i.e., conditions and entities and data usage rules.

– $dcr_1.c \wedge dcr_2.c \vdash dcr_1.c$. [By $\wedge$-elimination]

– $min(dcr_1.e, dcr_2.e) \leq_{\mathcal{E}} dcr_1.e$ [By Def. min]

– $dcr_1.dur \sqcup_{\mathcal{DUR}} dcr_2.dur \preceq_{\mathcal{DCR}} dcr_1.dur$. [By Lemma 1]

$\underline{dcr_1 \sqcup_{\mathcal{DUR}} dcr_2 \preceq_{\mathcal{DUR}} dcr_2}$ - The proof is symmetric to the previous case. □

Lemma 3. *The operation $\sqcup$ in Definition 8 is privacy preserving.*

Proof. Let p_1 and p_2 be two PILOT privacy policies. We show that $p_1 \sqcup p_2 \sqsubseteq p_1$ and $p_1 \sqcup p_2 \sqsubseteq p_2$. We proof each conjunct separately.

$\underline{p_1 \sqcup p_2 \sqsubseteq p_1}$ - We split the proof in cases based on the structure of PILOT policies, i.e., datatype ($p_1.t$,$p_2.t$), data communication rules ($p_1.dcr, p_2.dcr$) and transfers ($p_1.TR, p_2.TR$).

– $min(p_1.t, p_2.t) \leq_{\mathcal{T}} p_1.t$. [By Def. of min]

– $p_1.dcr \sqcup_{\mathcal{DCR}} p_2.dcr \preceq_{\mathcal{DCR}} p_1.dcr$. [By Lemma 2]

– $\forall tr_{\sqcup} \in \{tr_1 \sqcup_{\mathcal{DCR}} tr_2 \mid tr_1 \in p_1.TR \wedge tr_2 \in p_2.TR \wedge tr_1 \preceq_{\mathcal{DCR}} tr_2\} \cdot \exists tr \in TR_1$ s.t. $tr_{\sqcup} \preceq_{\mathcal{DCR}} tr$. Assume $tr_{\sqcup} \in \{tr_1 \sqcup_{\mathcal{DCR}} tr_2 \mid tr_1 \in p_1.TR \wedge tr_2 \in p_2.TR \wedge tr_1 \preceq_{\mathcal{DCR}} tr_2\}$. Then $tr_1 \in p_1.TR$ and $tr_{\sqcup} \preceq_{\mathcal{DCR}} tr_1$. [By Lemma 2]

$\underline{p_1 \sqcup p_2 \sqsubseteq p_2}$ - The proof is symmetric to the previous case. □